



CONFORME
ALLE NUOVE
INDICAZIONI 2025



DIDATTICA
INCLUSIVA

G. Bissoli • A. Del Zozzo
M. Lodi • A. Montresor



Sulle orme di Milù



**GUIDA
DI
INFORMATICA**

INTRODUZIONE AI CONCETTI
CHIAVE DELL'INFORMATICA

LEZIONI GUIDATE PASSO PASSO

© 2026 by Mondadori Education S.p.A., Milano

Tutti i diritti riservati

Mondadori Education S.p.A.
Via Privata Mondadori, 1 – 20054 Segrate (MI)
servizioclienti.edu@mondadorieducation.it
www.mondadorieducation.it

Fotocopie e permessi di riproduzione. Le fotocopie per uso personale del lettore possono essere effettuate nei limiti del 15% di ciascun volume/fascicolo di periodico dietro pagamento alla SIAE del compenso previsto dall'art. 68, commi 4 e 5, della legge 22 aprile 1941 n. 633.

Le fotocopie effettuate per finalità di carattere professionale, economico o commerciale o comunque per uso diverso da quello personale possono essere effettuate a seguito di specifica autorizzazione rilasciata da CLEARedi, Centro Licenze e Autorizzazioni per le Riproduzioni Editoriali, Corso di Porta Romana 108, 20122 Milano, e-mail autorizzazioni@clearedi.org e sito web www.clearedi.org.

Riferimenti a marchi commerciali e siti di terze parti. Occasionalmente, possono essere visibili in questo testo nomi, confezioni e marchi commerciali di prodotti o società. L'autore e l'editore non intendono sostenere che i prodotti fotografati o citati siano migliori o peggiori di altri, né indirettamente consigliarne o sconsigliarne l'acquisto: non esiste alcun rapporto di nessun genere con i relativi produttori. I riferimenti a nomi e marchi commerciali sono da intendersi come marchi registrati dalle rispettive società anche se, per semplicità di grafia, si è omessa la relativa indicazione.

L'editore fornisce – per il tramite dei testi scolastici da esso pubblicati e attraverso i relativi supporti – link a siti di terze parti esclusivamente per fini didattici o perché indicati e consigliati da altri siti istituzionali. Pertanto, l'editore non è responsabile, neppure indirettamente, del contenuto e delle immagini riprodotte su tali siti in data successiva a quella della pubblicazione, distribuzione e/o ristampa del presente testo scolastico.

Obiettivi di sviluppo sostenibile. Il contenuto di questo libro non è stato approvato dalle Nazioni Unite e quindi potrebbe non riflettere la posizione ufficiale di questa organizzazione, www.un.org/sustainabledevelopment.

Testi in formato accessibile. L'Editore promuove l'accessibilità e collabora con AID – Associazione Italiana Dislessia, Seleggio e con la Biblioteca Italiana per i Ciechi “Regina Margherita”. La conformità dei prodotti e dei servizi resta disciplinata dalla normativa vigente.

Segnalazione di errori e/o imprecisioni. La realizzazione di un libro scolastico è un'attività complessa che comporta controlli di varia natura. Essi riguardano sia la correttezza dei contenuti sia la coerenza tra testo, immagini, strumenti di esercitazione e applicazioni digitali. È pertanto possibile che, dopo la pubblicazione, siano riscontrabili errori e imprecisioni.

Mondadori Education ringrazia fin da ora chi vorrà segnalarli a:

Servizio Clienti Mondadori Education

e-mail servizioclienti.edu@mondadorieducation.it – numero verde 800 123 931

Per eventuali e comunque non volute omissioni e per gli aventi diritto tutelati dalla legge, l'editore dichiara la piena disponibilità.

Realizzazione editoriale

Redazione Curvilinee

Progetto grafico e impaginazione Curvilinee

Copertina Ilaria Raboni

Disegno di copertina Manuela Leporesi

I marchi e i nomi registrati sono a tutti gli effetti proprietà delle società che ne detengono i diritti, anche se non viene fatto riferimento specifico a tale circostanza nel testo.

Contenuti digitali

Progettazione Fabio Ferri, Nicola Barzagli

Realizzazione Silvia Sferruzza, EICON S.r.l., IMMAGINA S.r.l., Lumina Datamatics, Sidecar Studio

Il Sistema Qualità di Mondadori Education S.p.A. è certificato da Bureau Veritas Italia S.p.A. secondo la Norma UNI EN ISO 9001.

Questo ebook contiene materiale protetto da copyright e non può essere copiato, riprodotto, trasferito, distribuito, noleggiato, licenziato, trasmesso in pubblico, o utilizzato in alcun altro modo ad eccezione di quanto è stato specificamente autorizzato dall'editore in base ai termini e alle condizioni alle quali è stato acquistato o di quanto esplicitamente previsto dalla legge applicabile. Qualsiasi utilizzo non autorizzato di questo testo, inclusi quelli connessi a tecnologie di intelligenza artificiale o all'estrazione di testo e dati, così come l'alterazione delle informazioni elettroniche sul regime dei diritti costituisce una violazione dei diritti dell'editore e dell'autore e comporta l'applicazione di sanzioni civili e penali secondo quanto previsto dalla Legge 633/1941 e successive integrazioni e modifiche.

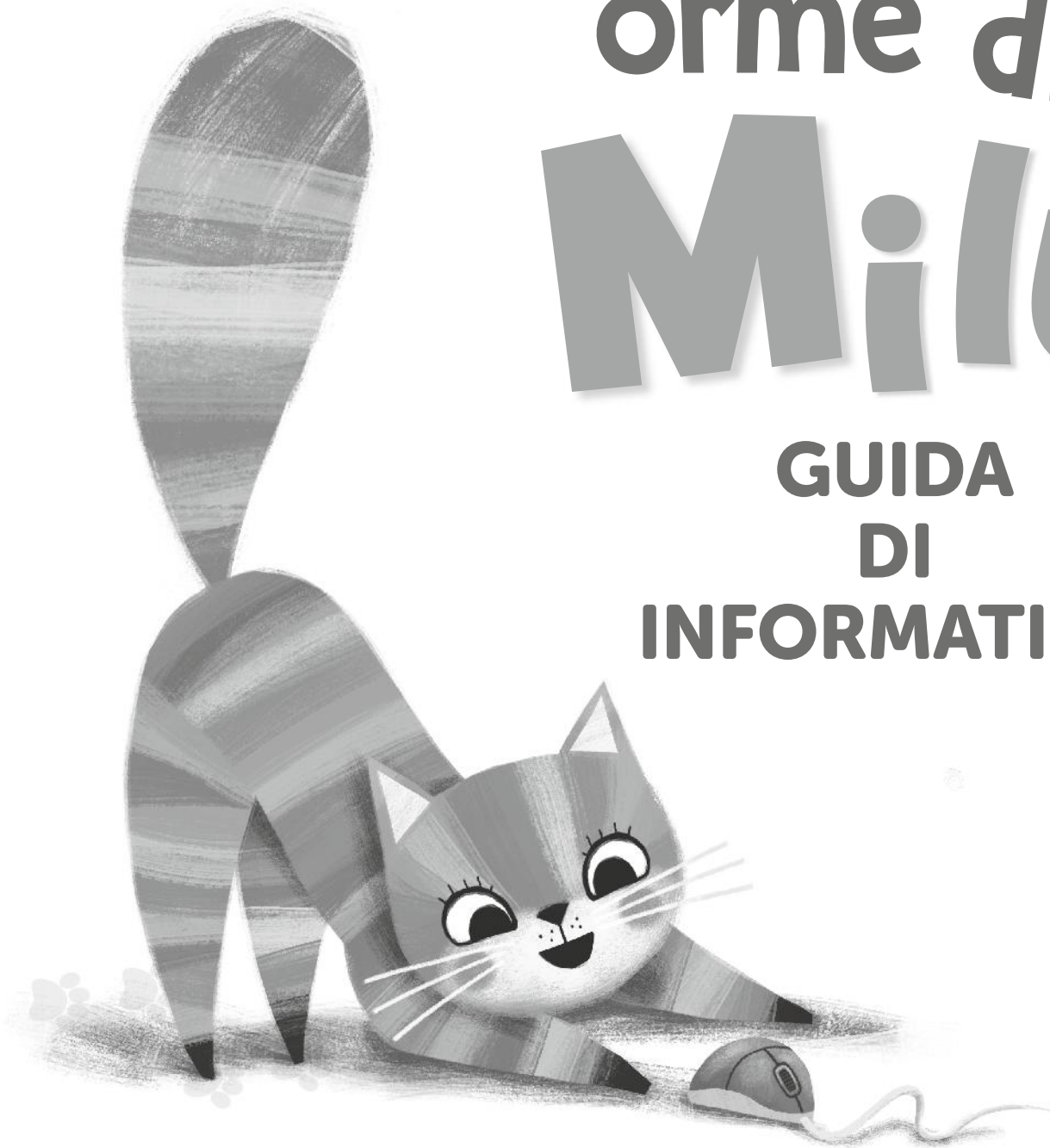
Questo ebook non potrà in alcun modo essere oggetto di scambio, commercio, prestito, sublicenza o altrimenti diffuso o messo a disposizione di terzi senza il preventivo consenso scritto dell'editore.

G. Bissoli • A. Del Zozzo
M. Lodi • A. Montresor



Sulle orme di **Milù**

**GUIDA
DI
INFORMATICA**



INDICE

CAPITOLO 1: INTRODUZIONE 5

Perché l'informatica fin dalla scuola primaria.....	5
Perché Questo Manuale.....	6
Dalla teoria alla pratica - Leggi qui!.....	7
Visione didattica.....	10

CAPITOLO 2: CONCETTI FONDAMENTALI DI INFORMATICA 14

Informatica: la scienza dell'elaborazione automatica	14
Informazioni e dati.....	14
Elaborare dati.....	16
Competenze digitali o competenze informatiche?.....	17
Istruzioni ed esecutori.....	17
E gli algoritmi?	21
Riferimenti.....	22

CAPITOLO 3: ALGORITMI QUOTIDIANI 23

Traguardi e obiettivi generali.....	23
Prerequisiti	23
Contenuti, in breve.....	24
Modalità di lavoro	24
• ATTIVITÀ 1: Sequenze e ordinamenti.....	25
• ATTIVITÀ 2: Istruzioni e programmi.....	26
• ATTIVITÀ 3: Descrivere configurazioni geometriche.....	29
Possibili estensioni, variazioni e personalizzazioni	33
Aspetti informatici	33
Relazione con l'Informatica nelle Indicazioni Nazionali.....	34
Valore costruzionista.....	35
Suggerimenti per ulteriori attività e riferimenti.....	35

CAPITOLO 4: MOVIMENTI SU UNA GRIGLIA 37

Traguardi e obiettivi generali.....	37
Prerequisiti	37
Contenuti, in breve.....	37
Movimenti su una griglia e sistemi di riferimento	38
Modalità di lavoro	40
• ATTIVITÀ 1: Be my robot	41
• ATTIVITÀ 2: Be my more intelligent robot.....	47
• ATTIVITÀ 3: Bee my robot.....	49

Possibili estensioni, variazioni e personalizzazioni	50
Aspetti informatici	50
Relazione con l'Informatica nelle Indicazioni Nazionali	51
Valore costruzionista	52
Suggerimenti per ulteriori attività e riferimenti	52

CAPITOLO 5: RAPPRESENTAZIONE DELL'INFORMAZIONE 54

Traguardi e obiettivi generali	54
Prerequisiti	55
Contenuti, in breve	55
Modalità di lavoro	55
• ATTIVITÀ 1: Capire il sistema di codifica	57
• ATTIVITÀ 2: Pixel art	58
• ATTIVITÀ 3: Luci di Natale	61
Possibili estensioni, variazioni e personalizzazioni	62
Aspetti informatici	62
Relazione con l'Informatica nelle Indicazioni Nazionali	64
Valore costruzionista	64
Suggerimenti per ulteriori attività e riferimenti	64

CAPITOLO 6: STRATEGIE DI SOLUZIONE DI PROBLEMI 65

Traguardi e obiettivi generali	65
Prerequisiti	65
Contenuti, in breve	65
Problema, algoritmo, strategia	65
• ATTIVITÀ 1: Trova il sacchetto più pesante (gruccia-bilancia)	66
• ATTIVITÀ 2: Il barista ordinato	69
Relazione con l'Informatica nelle Indicazioni Nazionali	73
Valore costruzionista	74
Suggerimenti per ulteriori attività e riferimenti	74

CAPITOLO 7: CODE.ORG 76

Traguardi e obiettivi generali	76
Prerequisiti	76
Preparazione	77
• ATTIVITÀ 1: Introduzione a Code.org	78
• ATTIVITÀ 2: Labirinti (Corsi A e B)	81
• ATTIVITÀ 3: Artista (Corsi A e B)	83
• ATTIVITÀ 3: Labirinti (Corsi C)	86
Altre attività	87
Modalità di lavoro	88
Valore costruzionista	91

Aspetti informatici	92
Relazione con l'Informatica nelle Indicazioni Nazionali	93
Suggerimenti per ulteriori attività e riferimenti	94

CAPITOLO 8: PROGRAMMAZIONE CREATIVA CON SCRATCH 95

Prerequisiti	96
Traguardi e obiettivi generali	96
Preparazione	96
• ATTIVITÀ 1: Benvenuti in Scratch!	97
• ATTIVITÀ 2: Diventa un regista con Scratch!	98
• ATTIVITÀ 3: Dialogo tra due personaggi	100
• ATTIVITÀ 4: Movimenti precisi	102
• ATTIVITÀ 5: Posizionare un personaggio	104
• ATTIVITÀ 6A: Sperimentare programmi	107
• ATTIVITÀ 6B: Solo 8 blocchi!	107
Bonus: Costumi e suoni	108
Modalità di lavoro	109
Aspetti informatici	110
Relazione con l'Informatica nelle Indicazioni Nazionali	110
Valore costruzionista	111
Suggerimenti per ulteriori attività e riferimenti	111

CAPITOLO 9: CONCLUSIONI 113

SCHEDE Alunno/a 114

CAPITOLO 1: INTRODUZIONE

Insegnante che ti appresti a leggere questa guida con curiosità, scetticismo, speranza, timore, entusiasmo o un mix di tutte queste cose: un sorriso a te! 😊

Questa guida propone attività di **informatica** per il primo triennio della scuola primaria. Leggendola, a volte ti verrà spontaneo pensare: “ma non è così diverso da quello che faccio già!”: hai perfettamente ragione! Ciò che faremo in questo percorso insieme è guardare a queste attività anche con una nuova lente, quella dell’informatica, che potremmo chiamare *computazionale*.

Molte attività si basano infatti su esperienze motorie, sull’esplorazione progressiva dello spazio ambientale e personale, sull’uso intenzionale del linguaggio, sulla collaborazione tra pari e sulla riflessione sugli errori: tutte pratiche che accompagnano e sostengono il naturale percorso evolutivo della scuola primaria. La novità sta nella prospettiva con cui tali pratiche vengono affrontate, ossia attraverso la lente dell’informatica. Inoltre, le stesse attività diventano occasioni per rafforzare le competenze di risoluzione dei problemi, sviluppare la capacità di pianificazione e, al tempo stesso, potenziare la creatività dei nostri bambini e bambine.

Questo è il punto di partenza di un percorso da affrontare per tutto il primo ciclo. Prima, attività senza computer, semplici e giocose; poi, a partire dalla terza primaria, anche con l’introduzione graduale di strumenti digitali facili da usare e gratuitamente disponibili in rete.

PERCHÉ L’INFORMATICA FIN DALLA SCUOLA PRIMARIA

Società digitale

Oggi viviamo in un mondo in cui le tecnologie digitali plasmano il modo in cui ci informiamo, ci intratteniamo, comunichiamo con gli altri e prendiamo decisioni nella vita quotidiana. Per orientarsi in questa realtà in modo consapevole non basta saper usare tablet, computer o applicazioni: è necessario capirne i meccanismi sottostanti, così da poterli dominare e valutare criticamente.

La diffusione dell’intelligenza artificiale rende questa esigenza ancora più urgente: sempre più spesso affideremo alle macchine compiti complessi, da svolgere seguendo le nostre indicazioni. Per questo motivo è fondamentale imparare a interagire con la tecnologia digitale non solo come consumatori passivi, ma come protagonisti attivi e consapevoli.

Qui entra in gioco l’informatica. È bene chiarire subito un equivoco diffuso: l’informatica non coincide con la tecnologia. Fare informatica non significa semplicemente usare dispositivi digitali, ma studiare i principi che li rendono possibili. È una disciplina scientifica a tutti gli effetti, che offre ai nostri alunni e alunne un nuovo modo di pensare e di interpretare il mondo. Così come la lettura apre le porte al linguaggio dei testi e la matematica permette di comprendere quantità e relazioni, l’informatica consente di padroneggiare i principi e i linguaggi alla base delle tecnologie che ci circondano.

In quest’ottica, le attività proposte mirano a sviluppare competenze per:

- **Affrontare problemi** scomponendoli in passi chiari e risolvibili;
- **Descrivere soluzioni** che possano essere eseguite da altri;
- **Riflettere sul proprio ragionamento**, allenando metacognizione e pensiero critico;
- **Usare la creatività** per immaginare soluzioni nuove e originali.

Avviare l'alfabetizzazione informatica fin dalla scuola primaria significa piantare le fondamenta di un curriculum verticale che accompagni gli alunni per tutto il loro percorso scolastico. La scuola primaria è infatti il luogo in cui tutti, indipendentemente dal contesto familiare o dalle opportunità offerte dall'esterno, possono accedere alle stesse esperienze di apprendimento. Introdurre l'informatica in questo contesto significa iniziare un percorso di equità, offrendo a ciascuno le basi per un rapporto attivo e critico con la tecnologia, accrescendo la consapevolezza dei meccanismi che regolano gli strumenti quotidiani e contribuendo a ridurre il divario digitale.

Informatica nelle Indicazioni Nazionali 2025

Le Indicazioni Nazionali 2025 introducono, per il primo ciclo, competenze, obiettivi e conoscenze di Informatica nelle discipline Matematica e Tecnologia.

Per la prima volta, le Indicazioni riconoscono l'importanza di avvicinare fin da subito bambine e bambini ai concetti fondamentali della disciplina.

Nella parte di matematica, troviamo obiettivi relativi a tre aree fondamentali: informazioni, dati e loro rappresentazione; algoritmi; programmazione. Queste tre aree sono coperte da questa guida. Verranno discusse in dettaglio e proposte attività didattiche relative alle competenze, agli obiettivi e alle conoscenze presenti nelle Indicazioni, specificati alla fine di ogni capitolo.

Nella parte di tecnologia, troviamo obiettivi relativi a varie aree fondamentali: aspetti tecnologici (componenti hardware e software di un sistema informatico, rete Internet e i suoi servizi); aspetti di sicurezza (protezione dei dati personali, crittografia) e uso consapevole delle tecnologie; creatività digitale, specialmente tramite la realizzazione di semplici applicazioni informatiche. In questa guida ci concentreremo in particolare sull'ultima (tramite la programmazione creativa con Scratch) e faremo accenni alle altre, lasciando gli approfondimenti alle classi quarta e quinta della scuola primaria.

PERCHÉ QUESTO MANUALE

Portare l'informatica in classe significa introdurre un linguaggio e un modo di pensare che non fanno parte dell'esperienza scolastica o universitaria di molti insegnanti. È un territorio ancora poco esplorato nella scuola primaria, in cui può capitare di sentirsi senza mappe né punti di riferimento chiari.

Questo manuale nasce per colmare proprio quel vuoto. Non solo per offrire spunti per attività da far svolgere in classe e a casa, ma anche per dare un filo conduttore, una visione unitaria e coerente di cosa significhi fare informatica alla scuola primaria.

Troverai:

- proposte di attività da svolgere in classe, adatte all'età e integrate nella vita di classe, e strategie didattiche per metterle in pratica al meglio;
- consigli pratici, soluzioni, estensioni delle schede presenti nel Quaderno di Informatica allegato al progetto di primo ciclo *Sulle orme di Milù*;
- la spiegazione, rigorosa ma semplice e accessibile, dei concetti fondamentali di informatica necessari per capire e lavorare su ciò che è richiesto nelle Indicazioni Nazionali 2025.

La nostra ambizione è duplice: fornire strumenti affidabili con cui lavorare e, allo stesso tempo, incoraggiare la sperimentazione e l'adattamento. Perché l'informatica alla scuola primaria non è solo un nuovo argomento da introdurre: è un'opportunità per aprire spazi di scoperta, di creatività e di ragionamento condiviso, in cui anche l'insegnante può imparare insieme ad alunni e alunne.

DALLA TEORIA ALLA PRATICA - LEGGI QUI!

Fermiamoci un attimo! Sfogliando le prossime pagine, potresti avere la sensazione di trovarti davanti a tanto, tantissimo testo. 😓 😓

Scusaci: in questi primi due capitoli ci prendiamo un po' di spazio per costruire un linguaggio comune, chiarire i concetti fondamentali e collocarli nel quadro educativo della scuola primaria. Non temere: non sarà così per tutto il manuale.

A partire dal capitolo 3, proponiamo attività da svolgere in classe attraverso schede operative. Se vuoi, puoi cominciare a dare un'occhiata direttamente alle schede: troverai tanti termini, alcuni familiari, altri nuovi, altri ancora usati anche in altre discipline, ma che in informatica assumono una sfumatura diversa. Dare un'occhiata alle schede ti spingerà a tornare indietro e a leggere con maggiore attenzione e consapevolezza queste pagine introduttive.

Ogni scheda contiene sempre:

- 🔍 attenzioni particolari per valorizzare l'esperienza e guidare l'osservazione;
- ❓ domande stimolo, per trasformare ogni gioco in un'occasione di riflessione informatica.

Il nostro obiettivo è chiaro: offrirti una raccolta di strumenti da usare. La parte scritta in questo manuale serve a darti sicurezza e una cornice di senso; il cuore del percorso sono le schede operative, in cui potrai sperimentare insieme ad alunni e alunne l'informatica come disciplina, mettendo in pratica con essa un uso del linguaggio creativo, rigoroso e divertente.

Informatica, pensiero computazionale e coding: facciamo chiarezza

Sei ancora qui! Molto bene 😊

Per dare coerenza a questo percorso serve prima di tutto una bussola. A scuola, infatti, raramente si parla di informatica, mentre capita spesso di sentire parlare di pensiero computazionale, programmazione e coding come se fossero sinonimi. Chiarire il significato di questi termini è il primo passo per dare all'informatica il giusto spazio e valore, evitando confusioni e fraintendimenti.

Questo manuale parla di informatica: la disciplina scientifica che studia l'**elaborazione automatica dell'informazione** - una disciplina autonoma con idee, metodi e un linguaggio propri. Come dici? Pensavi che l'informatica studiasse i computer? Questa definizione ti fa pensare a un'intelligenza artificiale che sceglie autonomamente quali informazioni dare al telegiornale? Non è proprio così. Pensa all'informazione come a qualunque cosa utile (per esempio, la foto dell'ultimo tuo viaggio è utile per fare invidia agli amici) che vuoi far elaborare in modo automatico (per esempio applicando un filtro del tuo smartphone che fa sembrare il cielo azzurro anche se in realtà era nuvoloso). Nel capitolo successivo, proveremo a dare definizioni più chiare e precise.

Così come in matematica si parla di "pensiero matematico" - caratterizzato da modelli, astrazioni e strategie tipiche del "fare matematica" - anche l'informatica permette di sviluppare un modo di ragionare disciplinare, che potremmo chiamare "pensiero informatico". Questo modo di pensare è caratterizzato dalla capacità di affrontare una classe di problemi, individuare una soluzione generale e descriverla in modo così chiaro e preciso che questa può essere eseguita da un agente esterno, spesso con un insieme molto preciso e limitato di azioni disponibili. In queste pagine useremo però il termine più diffuso, **pensiero computazionale**: non si tratta di "pensare come un computer", ma di adottare un metodo di ragionamento tipico dell'informatica, da applicare alla soluzione di problemi che ci troviamo davanti - a scuola e nella vita.

Una delle attività fondamentali dell'informatica è la **programmazione**: comprendere un problema, progettare una soluzione e scrivere, passo dopo passo, una sequenza di istruzioni che

possono essere eseguite da un computer. La parte più pratica e operativa (inserire materialmente queste istruzioni nel computer) è detta dagli informatici **coding**. È una parola molto usata a scuola negli ultimi anni (a volte - con un repentino cambio di significato - addirittura per indicare una nuova disciplina), ma agli informatici non piace, perché riduce l'informatica al solo "scrivere comandi", perdendo di vista la comprensione del problema e la progettazione della soluzione. È come paragonare la scrittura creativa all'atto di trascrivere un testo su carta!

Quindi, come sottolineato anche dall'Accademia della Crusca, coding, pensiero computazionale e informatica non sono assolutamente sinonimi [Lodi e Martini, 2024]: l'informatica è una disciplina, mentre il pensiero computazionale è il modo di pensare (e di approcciare i problemi) tipico di chi se ne occupa. La programmazione è un modo con cui gli informatici esprimono le loro soluzioni affinché queste vengano eseguite da un computer.

Già ti immaginiamo obiettare: "voi la fate facile, ma come pensa un informatico? 🤖".

Pensiero computazionale

Negli anni, sono state proposte innumerevoli definizioni di pensiero computazionale; riportiamo qui quella che ci sembra più completa:

Il pensiero computazionale è il processo mentale coinvolto nella **formulazione di un problema** e nell'**esprimere le sue soluzioni** in una forma che può essere **effettivamente eseguita** da un **agente di elaborazione delle informazioni** (umano o macchina).

Troppo astratta? Per comprendere meglio di che si tratta, ti consigliamo di guardare il video Exact Instruction Challenge¹, di Josh Darnit, su YouTube. Il video è in inglese, ma è facile da comprendere. In alternativa, puoi guardare un vecchio (ma incredibilmente efficace) video del maestro Manzi, in italiano².

Nel primo video, Josh propone ai suoi figli, Johanna ed Evan, una sfida: scrivere una lista di istruzioni per preparare un sandwich al burro di arachidi e marmellata! Una volta che i figli le avranno scritte, Josh le eseguirà alla lettera. I bambini iniziano con una serie di istruzioni incomplete, che non permettono di raggiungere l'obiettivo, e man mano le perfezionano. Dopo alcuni tentativi, Johanna riesce nell'impresa, mentre Evan, più piccolo, si arrabbia.

Particolarmente interessanti sono alcuni momenti del video (viene indicato il minutaggio del momento del video a cui si fa riferimento):

- Evan ha scritto: "Prendi una fetta di pane e spalmala in giro con il coltello." In queste istruzioni, però, non viene menzionato il burro di arachidi. Quando Josh glielo fa notare, Evan esclama "Hold on" (aspetta!), e sul suo volto emerge la consapevolezza che le istruzioni non sono complete. **1:06**
- Johanna ha scritto: "Con il coltello, prendi un po' di burro di arachidi," e, quando sente il padre leggere la frase ad alta voce, commenta: "Un po' significa un sacco." Alla domanda sorpresa di Josh, "Un po' significa un sacco?", lei risponde: "Nel mio mondo." **1:55**
- Più avanti, Josh legge l'istruzione: "Inserisci il coltello nel barattolo" e lo inserisce con la lama rivolta verso l'alto. Lo stupore e il disappunto si leggono sul volto di Johanna: dopo vari tentativi, pensava di aver formulato la soluzione in modo preciso, ma si rende conto che c'era ancora un passaggio ambiguo. **3:18**

¹ <https://q3.hubscuola.it/f00>

² <https://q3.hubscuola.it/v0o> (DAL MINUTO 8:44)

Partendo dal video, proviamo a spiegare meglio alcuni aspetti chiave del pensiero computazionale.

■ **Riflettere sul procedimento di risoluzione dei problemi**

Se il problema (l'obiettivo) dato ai figli di Josh fosse stato quello di preparare un panino al burro di arachidi e marmellata, il risultato sarebbe stato, appunto... lo specifico panino da loro preparato! Qui la loro sfida è invece descrivere un modo (una "ricetta") per preparare questo tipo di panini. A Johanna ed Evan non basta ottenere un risultato (anzi, non serve: a quello penserà il padre preparando il panino), ma devono pensare al procedimento per ottenerlo.

Questo è un particolare approccio alla soluzione dei problemi tipico dell'informatica: astrarre la soluzione in modo che sia applicabile a tutti i problemi "di quel tipo" (Josh potrà seguire la ricetta ogni volta che vuole prepararsi un panino al burro di arachidi e marmellata).

■ **Esprimere una soluzione**

In questa sfida, descrivere una soluzione (una ricetta) richiede di mettersi nei panni dell'altro. È necessario esprimere la soluzione come una sequenza di istruzioni elementari che possano essere comprese, in modo non ambiguo, dall'esecutore. Quando si descrive un procedimento (o qualsiasi argomento) è fuorviante fare riferimento al "proprio mondo", come fa Johanna; bisogna invece avere un'idea chiara delle istruzioni e dei concetti che l'esecutore è in grado di comprendere. Il padre "fa il finto tonto", cercando deliberatamente le ambiguità nella descrizione. Non avendo compreso pienamente le regole del gioco, Evan si arrabbia perché sa che l'esecutore è intelligente e conosce già la soluzione; perciò si aspetta che il padre interpreti le sue istruzioni, anche se ambigue.

Il computer è l'esecutore più "stupido" di tutti: conosce solo pochissime istruzioni elementari: non aggiunge sue interpretazioni, ma ha il vantaggio di eseguirle sempre allo stesso modo, in modo totalmente prevedibile. Spiegare un procedimento a un computer richiede quindi una comprensione profonda del problema e della soluzione, tenendo sempre conto del livello di comprensione del proprio interlocutore.

Qui il padre "fa il computer": esegue alla lettera ciò che legge, senza aggiungere o interpretare.

■ **Scomporre il problema e le sue soluzioni**

Nel video si nota che Johanna individua subito la necessità di scomporre il problema in due parti: imburrare una fetta e spalmare la marmellata sull'altra, per poi ricomporre le soluzioni parziali nel panino completo. Evan, più giovane, all'inizio fa più fatica; nella sua seconda soluzione, infatti, viene fuori un pasticcio, poiché il problema non è stato scomposto in modo efficace.

È interessante notare che la necessità di scomporre problemi in sottoproblemi non è un'esigenza del computer, ma serve a noi per affrontare una complessità che, se non domata, andrebbe oltre le nostre capacità cognitive.

■ **Ottenere un feedback dai propri errori**

Ogni volta che la ricetta viene "eseguita", Johanna ed Evan ricevono un feedback sulla loro soluzione: osservano che alcune parti funzionano, mentre altre richiedono ulteriori aggiustamenti. La possibilità di testare e migliorare le proprie soluzioni è un aspetto cruciale di questa sfida, poiché consente ai ragazzi di imparare dai propri errori. Senza l'opportunità di imparare dagli errori, infatti, non può esserci un vero apprendimento.

In informatica, questa attività è nota come **debugging**; ancora una volta, la capacità di analizzare i propri errori è una competenza fondamentale in ogni campo del sapere, dalla mate-

matica alla letteratura. In informatica, tuttavia, il feedback arriva “prima”. Non serve aspettare che l’insegnante corregga, non c’è in fondo al libro un ermetico numero come risultato. Una volta scritto il programma, si può eseguirlo, “vedere l’effetto che fa” e spesso anche osservare qual è l’effetto di ogni singolo passaggio, per verificare se il programma funziona come previsto e, in caso, apportare correzioni.

■ Una riflessione conclusiva sul pensiero computazionale

Ti riproponiamo la definizione di pensiero computazionale:

Il **pensiero computazionale** è il processo mentale coinvolto nella **formulazione di un problema** e nell’**esprimere le sue soluzioni** in una forma che può essere **effettivamente eseguita** da un **agente di elaborazione delle informazioni** (umano o macchina).

Questa definizione evidenzia subito un punto cruciale: il pensiero computazionale non riguarda l’esecuzione di procedure, ma si concentra sulla capacità di formulare problemi, progettare soluzioni e rappresentarle in modo chiaro e condivisibile. Si tratta di un particolare modo di risolvere problemi, che potremmo definire *problem solving computazionale*.

Nella scuola, a volte, gran parte del lavoro si concentra sul rafforzamento delle competenze procedurali: imparare tecniche, ripetere esercizi, applicare regole. I bambini e le bambine assumono il ruolo di **esecutori**. Tutti aspetti importanti, ma parziali. Il pensiero computazionale invita a fare un passo oltre: non solo “seguire procedure”, ma anche discutere i problemi, cercare soluzioni generali, analizzare gli errori e trasformarli in occasioni di apprendimento.

Forse già promuovi questo approccio nelle tue attività quotidiane, quando chiedi ad alunni e alunne di spiegare un procedimento, di affrontare un compito complesso a piccoli passi, o di riflettere insieme sugli errori. L’obiettivo di questo testo è proprio aiutarti a riconoscere, in queste pratiche, l’essenza dell’informatica e a rafforzarla sempre di più, per offrire ai bambini e alle bambine competenze che, nel mondo di oggi – fatto di dati, algoritmi e automazione – sono fondamentali per sviluppare un rapporto critico e consapevole con la tecnologia.

VISIONE DIDATTICA

Informatica senza e con computer

Nei capitoli seguenti, proponiamo attività in due modalità.

- **Unplugged:** attività **senza computer**, incentrate sull’uso del corpo, sul linguaggio, sul gioco e sulla collaborazione. Si realizzano con materiali semplici (griglie, carte, pennarelli) e materializzano i concetti dell’informatica in forme concrete e accessibili. Queste attività permettono anche di evitare la visione errata dell’informatica come “la scienza dei computer”.
- **Con strumenti digitali:** attività che introducono gradualmente **ambienti di programmazione** visuale (cioè con istruzioni grafiche, non testuali) e **robot educativi**. In queste attività, i bambini e le bambine hanno un riferimento tangibile e concreto delle idee sperimentate nelle attività unplugged: concetti astratti come sequenze, condizioni, ripetizioni si materializzano in comportamenti di personaggi che si muovono sullo schermo o di robot che seguono comandi. Queste attività permettono di sperimentare concretamente cosa significhi dare istruzioni a un esecutore automatico “stupido e intransigente”.

Le proposte didattiche seguono il naturale percorso evolutivo degli alunni e delle alunne: dalle esperienze corporee e motorie, dal gioco simbolico e dalla narrazione, fino a raggiungere gradualmente forme sempre più astratte di rappresentazione.

Il costruzionismo

Quello che oggi chiamiamo “pensiero computazionale” affonda le sue radici negli studi che Seymour Papert fece già negli anni Settanta, quando i computer iniziarono a entrare nelle scuole. Papert ebbe un ruolo duplice e originale: da matematico e informatico, seppe riconoscere il valore dell'informatica non solo come disciplina da apprendere, ma anche come ambiente privilegiato per pensare, esplorare e costruire conoscenza; da pedagogo, tradusse i principi astratti del costruttivismo di Jean Piaget in pratiche educative concrete, capaci di dare forma all'apprendimento quotidiano degli alunni e alunne.

Da questo intreccio nasce il costruzionismo, la sua teoria dell'apprendimento. Come il costruttivismo, anche il costruzionismo sostiene che la conoscenza non si riceve passivamente, ma si costruisce attivamente attraverso l'esperienza. La novità introdotta da Papert è l'idea che l'apprendimento diventa particolarmente efficace quando prende la forma di un'attività creativa orientata alla produzione di artefatti significativi: prodotti che hanno valore per chi li realizza o per la comunità di cui fanno parte.

Papert chiama questi artefatti “oggetti con cui pensare”: strumenti che rendono le idee visibili, manipolabili, discutibili e migliorabili, facilitando la costruzione di modelli mentali. Possono essere concreti o astratti – un disegno, un racconto, una costruzione con i mattoncini, una sequenza di istruzioni, un programma al computer – ma ciò che conta è che offrano a chi apprende un terreno fertile su cui sperimentare, riflettere e crescere.

■ Costruzionismo e informatica

Nella prefazione del suo libro più famoso, *Mindstorms*, Papert [1980] afferma che da bambino era affascinato dagli ingranaggi – smontava qualunque oggetto meccanico per capirne il funzionamento. Il suo modello mentale sugli ingranaggi gli ha permesso di comprendere concetti matematici più complessi, come le proporzioni, le frazioni, fino alle equazioni differenziali:

“Credo che l'aver tanto giocato con i differenziali sia stato più efficace per la mia comprensione della matematica, di tutto quello che mi è stato insegnato alla scuola elementare. Gli ingranaggi, servendomi da modelli, hanno fatto entrare nella mia mente idee che altrimenti sarebbero restate astratte.”

Nella scelta di questi “oggetti con cui pensare”, però, non c'è solo un aspetto conoscitivo: è sempre in gioco una fondamentale componente affettiva. La passione di Papert per gli ingranaggi è personale e non riproducibile.

“Una Maria Montessori dei nostri giorni potrebbe proporre, se la mia storia l'avesse convinta, di creare una confezione di ingranaggi per bambini. Così ogni bambino potrebbe fare l'esperienza che ho fatto io. Ma sperarlo sarebbe trascurare l'essenza della storia: io mi ero innamorato degli ingranaggi. È dunque qualcosa che non può essere ridotto in termini soltanto cognitivi. Era avvenuto qualcosa di molto personale, e non si può presumere che ciò si ripeterebbe per altri bambini esattamente nella stessa forma.”

Il computer, per Papert, diventa il più flessibile fra questi “oggetti con cui pensare”, una sorta di oggetto universale, capace di rappresentare e simulare qualsiasi sistema o concetto. Come una volta gli ingranaggi hanno offerto a lui stesso una comprensione intuitiva di concetti complessi, ora l'informatica può fornire a ogni studente un insieme flessibile di strumenti con cui esplorare e comprendere idee astratte. Il computer non è quindi solo un mezzo per eseguire calcoli o simulazioni, ma uno spazio di esplorazione in cui gli alunni e le alunne possono creare e modificare in prima persona.

“La mia tesi potrebbe essere sintetizzata nel modo seguente: quello che gli ingranaggi non possono fare, il computer lo potrebbe. Il computer è il Proteo delle macchine. La sua essenza è la sua universalità, il suo potere di simulare. Poiché può assumere migliaia di forme e può essere utilizzato per migliaia di funzioni, può soddisfare migliaia di gusti.”

Il computer ha anche un altro ruolo nell'apprendimento: consente agli alunni e alunne di vedere immediatamente il risultato delle proprie azioni - ogni comando o modifica genera una risposta visibile e immediata, che stimola ulteriori tentativi e incoraggia l'auto-correzione. Questo ciclo di azione-feedback-miglioramento rende l'apprendimento più efficace, consentendo agli alunni e alle alunne di sperimentare. In questo senso, il feedback immediato del computer diventa un potente “oggetto con cui pensare”, uno strumento che aiuta a interiorizzare e applicare concetti in un processo di scoperta continua. Questo aspetto rende l'uso del computer particolarmente adatto alla pedagogia costruzionista, favorendo un apprendimento in cui il discente può iterare, riflettere e progredire autonomamente nel proprio percorso di scoperta.

Come si collega tutto questo alle discipline studiate alla scuola primaria? Papert utilizza il termine “Mathland” per indicare un paese immaginario in cui la matematica si impara con la stessa naturalezza con cui si impara a parlare, vivendo immersi in una lingua. L'informatica può diventare questo ambiente speciale: non un insieme di esercizi astratti, ma un luogo in cui i concetti matematici vengono costruiti tramite l'interazione con artefatti significativi. Per esempio, quando i bambini e le bambine programmano un piccolo robot, esplorano la geometria muovendosi sul piano e facendo uso di nozioni come angoli (a tale proposito, si vedano le attività nelle guide di classe terza del progetto PerContare <https://q3.hubscuola.it/1et>). In questo modo, la matematica non resta “fuori” dalle loro esperienze, ma diventa parte viva dei loro giochi, dei loro progetti e delle loro invenzioni.

Papert sottolinea che lo stesso approccio può estendersi anche al linguaggio. Così come la matematica può diventare parte viva dei giochi e delle esplorazioni, l'informatica può sostenere le capacità narrative ed espressive. Costruire storie interattive, far dialogare personaggi o programmare piccoli robot che recitano un racconto significa unire informatica e storytelling: gli alunni e le alunne imparano a dare ordine agli eventi, a strutturare sequenze, a esprimere le proprie idee in forma chiara e condivisibile.

■ Costruzionismo e apprendimento

Nel costruzionismo di Papert, l'obiettivo è ribaltare il processo tradizionale di insegnamento, passando da un approccio in cui il docente trasferisce conoscenze a uno in cui studenti e studentesse diventano protagonisti del proprio apprendimento. Invece di fornire informazioni già pronte, l'insegnante ha il compito di creare condizioni che favoriscano la scoperta. Per il costruzionismo, quindi, l'apprendimento è un autentico processo esplorativo, in cui il ruolo dell'insegnante è quello di offrire agli alunni e alunne esperienze e strumenti che consentano loro di approfondire concetti e costruire attivamente la propria conoscenza.

Tradurre questa prospettiva in pratica significa organizzare attività che non siano solo esercizi da svolgere, ma vere esperienze di costruzione e riflessione. Alcuni principi chiave:

- **Ribaltare il processo di apprendimento**

Bisogna porsi sempre la domanda: come posso fare un passo indietro e rendere gli alunni e le alunne protagonisti? L'obiettivo non è fornire risposte, ma stimolare la ricerca e la costruzione di soluzioni personali.

- **Personalizzazione e motivazione**

Le attività hanno più valore se lasciano spazio agli interessi personali. Per esempio, pro-

grammare con Scratch un biglietto di auguri per la propria migliore amica non è solo un modo per imparare comandi, strutture di ripetizione o variabili, ma è anche (soprattutto) un modo per esprimere qualcosa di personale e rilevante. Questo rende la costruzione significativa e motivante, proprio come per Papert lo erano stati gli ingranaggi.

- **Valorizzare l'errore come motore creativo**

In un approccio costruzionista l'errore non è un fallimento, ma una scoperta. Ogni volta che un programma non funziona o che il risultato non è quello atteso, si apre uno spazio di ricerca: cos'è successo? Cosa posso cambiare? Spesso gli errori portano a soluzioni nuove e inattese. In questo senso, l'errore diventa un processo naturale di esplorazione e non un motivo di frustrazione.

- **Iterazione e miglioramento continuo**

Non esiste “la soluzione giusta e basta”: ogni costruzione può essere raffinata, estesa, modificata. Ogni processo può essere migliorato. Abituare gli alunni e le alunne a ripensare al proprio lavoro, cercando di renderlo più chiaro o più efficace, rafforza l'idea che il sapere è un processo in evoluzione.

- **Discussione e riflessione collettiva**

Il momento finale dell'attività è spesso il più importante: non fermarsi al “compito svolto”, ma dedicare tempo a discutere insieme. Che cosa ha funzionato? Dove ci siamo confusi? Come avremmo potuto fare meglio? Questo passaggio trasforma l'esperienza in apprendimento riflessivo: non solo “abbiamo fatto”, ma “abbiamo capito cosa significa fare”.

- **Artefatti visibili e condivisi**

Ogni attività dovrebbe concludersi con un prodotto tangibile (un disegno, una sequenza scritta, una mappa, un programma). Questo oggetto diventa la traccia del ragionamento fatto, può essere mostrato, discusso, confrontato. Così idee, scelte e strategie diventano visibili e condivisibili.

- **Uso creativo degli strumenti digitali**

Non proporre gli strumenti solo per eseguire esercizi guidati, ma lasciare che diventino mezzi espressivi: con Scratch si può raccontare una fiaba interattiva, con un robot mettere in scena una coreografia, con un editor di testi costruire un giornalino di classe.

- **Ruolo dell'insegnante come facilitatore**

Nel costruzionismo, l'insegnante non fornisce soluzioni già pronte, ma crea condizioni per la scoperta. In pratica: non correggere subito un errore, ma porre domande (“Cosa ti aspettavi che accadesse?”), rilancia le idee degli alunni e delle alunne, favorisce il confronto tra pari. È un ruolo più simile a quello di un allenatore che stimola e incoraggia, più che a quello di un esaminatore che giudica.

RIFERIMENTI

- Michael Lodi e Violetta Lonati. Informatica nelle Indicazioni Nazionali - Raccolta di materiali. <https://q3.hubscuola.it/21p>
- Michael Lodi e Simone Martini. Coding, pensiero computazionale e informatica: quale evoluzione possibile? ITADINFO 2024. <https://q3.hubscuola.it/gqh>
- Ministero dell'Istruzione e del Merito. Indicazioni Nazionali 2025 (versione del 7/7/2025). <https://q3.hubscuola.it/9b8>
- Seymour Papert, Prefazione a “Mindstorms: Children, computers, and powerful ideas”. Basic Books, Inc., 1980. (pag. VI-VIII). Traduzione di Augusto Chiocciariello. <https://q3.hubscuola.it/3fx>
- Giorgia Bissoli e Alberto Montresor. Insegnare il pensiero computazionale - strumenti, metodologie e riflessioni per la scuola. <https://q3.hubscuola.it/1hz>

CAPITOLO 2: CONCETTI FONDAMENTALI DI INFORMATICA

In questo capitolo introduciamo alcuni concetti chiave dell'informatica. Per far sì che i tuoi alunni e alunne inizino a sviluppare la capacità di approcciare il mondo e risolvere i problemi anche con una lente computazionale, è importante conoscere questi “ingredienti” di base: idee semplici ma molto potenti alla base di ogni programma o sistema informatico. Alcuni termini ti suoneranno familiari, ma in informatica assumono un significato specifico che è importante chiarire.

INFORMATICA: LA SCIENZA DELL'ELABORAZIONE AUTOMATICA

Così come l'oggetto di studio principale dell'astronomia non sono i telescopi, ma gli astri e l'universo, e così come l'oggetto di studio principale della biologia sono gli esseri viventi, e non i microscopi, così l'oggetto principale di studio dell'informatica **non** sono i computer. Al contrario, l'informatica studia l'**elaborazione automatica dell'informazione**: nelle prossime sezioni proviamo a capire, per ciascun concetto, che cosa significa.

INFORMAZIONI E DATI

L'**informazione** è una **conoscenza utile** per le persone, rispetto ad un determinato obiettivo. Ad esempio, può essere utile conoscere l'età di un bambino perché alcune attività sportive della scuola sono riservate ai bambini “dai 10 anni in su”.

Per **esprimere questa stessa informazione, possiamo usare dati diversi** (Lonati e Monga, 2025): l'anno di nascita del bambino, la sua data di nascita completa, oppure il numero di anni compiuti. Alcuni dati sono più precisi di altri: soprattutto se l'attività è all'inizio dell'anno, può fare la differenza sapere anche il giorno e il mese di nascita per capire se, a quella data, il bambino avrà già compiuto 10 anni oppure no. I dati, da soli, non hanno significato (per esempio: “10”, “2015”, “07/03/2015”). Diventano significativi solo all'interno di una convenzione e di un contesto d'uso: se decidiamo che “14/03/2015” rappresenta la data di nascita di una persona, allora possiamo interpretare quel dato come utile per esprimere una certa informazione (ad esempio per stabilirne l'età).

Sebbene quindi sarebbe più preciso dire che l'informatica si occupa dell'elaborazione automatica delle rappresentazioni dei dati che esprimono una certa informazione, per brevità nel seguito parleremo di elaborazione dei dati o di elaborazione dell'informazione.

Elaboratori

Ora immaginiamo di avere l'elenco di tutte le date di nascita degli alunni e delle alunne della scuola, con centinaia o migliaia di nomi, e di voler calcolare, per ciascuno, l'età precisa in un certo giorno. Farlo a mano richiederebbe moltissimo tempo. È qui che l'Informatica ci viene in aiuto: ci consente di **elaborare automaticamente questi dati**.

Oggi abbiamo a disposizione dei “macchinari” (potrebbe capitare di sentire gli informatici chiamarli “macchine”: non stanno parlando di automobili): i **computer**¹.

Il “macchinario computer” avrà al suo interno meccanismi specifici² (pensiamoli, per il momento, come interruttori e ingranaggi). La disposizione interna degli interruttori (acceso/spento) in un dato momento, detta **stato**, può rappresentare determinati dati. Gli ingranaggi, opportunamente progettati, potranno compiere fisicamente l’elaborazione (accendere/spegnere alcuni interruttori) e quindi portare la macchina a uno stato diverso, che sarà il risultato dell’elaborazione.

Un computer moderno potrebbe avere decine o centinaia di miliardi di interruttori (che, in questo contesto, si chiamano **transistor** e sono componenti elettronici).

Rappresentare numeri con “interruttori”

La prima sfida è quindi rappresentare qualsiasi informazione (o, più precisamente, qualsiasi dato che usiamo per esprimerla) esclusivamente mediante interruttori.

Dobbiamo quindi riuscire a rappresentare numeri, lettere, colori, immagini e suoni utilizzando solo interruttori.

Per i numeri è relativamente semplice. Il nostro sistema numerico è posizionale, cioè il valore di ciascuna cifra è stabilito dalla sua posizione, ed è decimale, cioè in base 10. Ad esempio, 2025 in base 10 ha, leggendolo da destra a sinistra, la cifra 5 nella posizione delle unità, 2 in quella delle decine, 0 in quella delle centinaia e 2 in quella delle migliaia) e quindi, in notazione polinomiale, $2025 = 2 \cdot 10^3 + 0 \cdot 10^2 + 2 \cdot 10^1 + 5 \cdot 10^0$.

Se invece di usare un sistema in base 10, ne usiamo uno in base 2 che ha due cifre soltanto (es. 0 e 1), possiamo dire che l’anno in cui sono state rilasciate le Indicazioni Nazionali è il 1111101001, cioè

$$1 \cdot 2^{10} + 1 \cdot 2^9 + 1 \cdot 2^8 + 1 \cdot 2^7 + 1 \cdot 2^6 + 1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$$

$$1024 + 512 + 256 + 128 + 64 + 32 + 8 + 1 = 2025$$

Possiamo allora rappresentare gli 1 con interruttori accesi e gli 0 con interruttori spenti (ma potremmo anche fare il contrario: l’importante è mettersi d’accordo).

In questo modo possiamo rappresentare tutti i numeri che vogliamo (con un numero finito di cifre binarie, in inglese *binary digit*, abbreviato in **bit**³). Ovviamente, più grande è il numero, più bit ci servono.

Rappresentare lettere e simboli

Per non complicarci la vita, possiamo sempre passare dai numeri, e quindi decidere che ogni **carattere** (lettera, cifra, simbolo, emoji, ...) viene rappresentato da un numero, e quel numero, abbiamo già visto come rappresentarlo sotto forma di interruttori.

1 Molti traducono *computer* con *calcolatore*, ma qui sceglieremo invece **elaboratore**, per porre maggiore attenzione sul loro scopo principale, che è appunto l’elaborazione automatica dei dati.

2 Solitamente con il termine **architettura** si indicano le parti principali di un computer (ad esempio processore, memoria e dispositivi di input/output) e come “collaborano” tra loro per farlo funzionare.

3 Per ragioni storiche e tecnologiche, chiamiamo **byte** una sequenza di **8 bit**.

L'associazione carattere-numero è detta **codice** o **sistema di codifica**, e, di nuovo, l'importante è mettersi d'accordo. Per esempio, il codice Unicode è uno standard internazionale, e associa ogni carattere a un numero. Ad esempio:

Carattere	Numero in base 10	Numero in base 2
!	33	100001
A	65	1000001
a	97	1100001
b	98	1100010
Ω	937	1110101001
爱	29233	111001000110001
😊	128522	11110110000001010

Rappresentare colori, immagini, video

Se ti avvicini molto ad uno schermo di un computer (meglio se vecchio), vedrai dei quadratini. Semplificando, dietro a ciascun quadratino che forma il pezzettino dell'immagine che vedi sullo schermo (un "elemento di un'immagine", in inglese picture element, cioè **pixel**) ci sono tre lampadine (led) colorate: una rossa, una verde e una blu. Cambiando l'intensità di ciascuna lampadina, possiamo ottenere tutti i colori (es. se le accendiamo tutte e tre avremmo tantissima luce: il bianco; se le spegniamo tutte e tre, avremo il nero; se accendiamo totalmente solo quella rossa avremo il rosso; se accendiamo totalmente quella rossa e per metà quella verde, avremo un bell'arancione). Nota che non stiamo lavorando con le tempere su un foglio: qui più colore (più luce) aggiungiamo, più andiamo verso il bianco: ad esempio, tanto verde+tanto blu = azzurro.

L'intensità di Rosso, Verde, Blu (Red, Green, Blue - **RGB**) può essere espressa, ad esempio (di nuovo, per convenzione e per ragioni tecnologiche), con un numero compreso tra 0 (lampadina spenta) e 255 (lampadina completamente accesa), e dunque ad esempio un bel blu potrebbe essere 51, 102, 255.

Con tanti pixel in griglia si forma un'immagine; con tante immagini mostrate velocemente, una di seguito all'altra, si forma un video.

ELABORARE DATI

Speriamo di averti convinto, almeno un po', che, tramite milioni di interruttori, è possibile rappresentare tutti i dati che vogliamo. Ma quello che vogliamo fare è anche elaborare questi dati - anzi, farlo fare ai computer!

I computer (oggi tramite circuiti elettronici) possono modificare quali interruttori sono accesi e quali spenti, cioè eseguire **operazioni sugli interruttori**⁴: operatori logici (leggendo l'interruttore acceso come "Vero" e spento come "Falso", ad esempio accendendo un bit solo se altri due sono già accesi), semplici operazioni aritmetiche sui bit, trasferimento di bit da/verso la memoria del computer, confronti e salti verso un'altra istruzione.

La potenza sta nel fatto che un computer moderno può eseguire **decine di miliardi di queste operazioni ogni secondo**.

⁴ Anche queste operazioni, essendo informazioni/dati, vengono rappresentate all'interno del computer tramite interruttori.

Il computer non ha alcuna idea del significato che noi esseri umani attribuiamo a queste operazioni: si tratta semplicemente di ingranaggi (circuiti elettronici) che accendono o spengono interruttori secondo la logica da noi definita. Il computer, inoltre, non può compiere altre operazioni se non quelle poche e semplici che abbiamo elencato, ma combinandone migliaia in modo sapiente, possiamo compiere elaborazioni molto avanzate.

COMPETENZE DIGITALI O COMPETENZE INFORMATICHE?

I computer (inclusi, ad esempio, smartphone, tablet, elettrodomestici smart) vengono quindi chiamati **dispositivi digitali** proprio perché rappresentano ed elaborano dati rappresentati in modo discreto (tramite cifre binarie, “binary digit” o bit).

Ma quando si parla di **competenze digitali**, si fa riferimento solo alla “*padronanza di uso efficace, sicuro e consapevole di dispositivi, strumenti e tecnologie digitali*” [MIM, 2025]. Questo manuale offre invece la possibilità di acquisire **competenze informatiche**, cioè capire come questi dispositivi possano rappresentare ed elaborare dati in modo automatico, pur seguendo procedimenti ideati dall'uomo.

ISTRUZIONI ED ESECUTORI

Il computer può eseguire solo operazioni molto semplici su dati rappresentati come bit. Per poterlo usare in modo efficace, abbiamo bisogno di descrivere queste operazioni in una forma che sia comprensibile agli esseri umani, ma che possa comunque essere tradotta automaticamente in tali operazioni. Per questo sono stati inventati diversi **linguaggi formali**, con precise regole sintattiche da rispettare: i **linguaggi di programmazione**. Essi svolgono proprio questo ruolo: permettono di esprimere in modo chiaro e strutturato le elaborazioni che vogliamo far eseguire al computer, affidando al computer stesso il compito di tradurle nelle corrispondenti operazioni sui bit.

Possiamo quindi “dimenticarci” degli ingranaggi e degli interruttori. Dato un linguaggio di programmazione, possiamo supporre che esista un **esecutore** in grado di “comprendere” le istruzioni scritte in quel linguaggio, cioè di eseguirle senza ambiguità. L'esecutore “non ha dubbi” su come interpretare l'istruzione: ha modo di eseguirla direttamente o tramite passaggi automatici di traduzione nascosti al programmatore. L'esecutore compirà sempre le stesse operazioni in modo meccanico e totalmente prevedibile in risposta a una stessa istruzione.

Se gli esecutori che abbiamo in mente sono chiaramente i computer, didatticamente può essere utile lavorare con esecutori diversi (ad esempio robot, che sono comunque computer, ma anche esseri umani che “fanno finta di essere un computer”, come il papà che seguiva passo passo e alla lettera le istruzioni per preparare il panino nel video del Capitolo 1).

Ogni volta che vogliamo far fare qualcosa a un esecutore, dobbiamo esprimerlo tramite un **linguaggio** per lui “comprensibile”. In tutte le attività che proponiamo, cerchiamo sempre di chiarire (a meno che il compito non sia proprio quello di inventarle o scoprirle) qual è **l'insieme di istruzioni che quel particolare esecutore sa eseguire**.

Per descrivere questo insieme, ci serve definire la **sintassi**, cioè le regole del linguaggio (come si scrivono le istruzioni? Ad esempio: mediante specifiche parole o simboli, indispensabili perché l'esecutore possa interpretarle ed eseguirle) e la **semantica**, cioè il significato delle istruzioni in termini di comportamento del programma, un comportamento a cui l'esecutore non assegna un significato come farebbe un essere umano, ma che segue in modo rigoroso e meccanico durante l'esecuzione.



SUL QUADERNO DI INFORMATICA

Nel Quaderno di Informatica, raccogliamo sintassi e semantica nel *Manuale delle Istruzioni*⁵, e lo rappresentiamo sempre con una tabella blu. Ad esempio a pagina 8:

Il panda-robot Cesare è capace di capire queste istruzioni:

MANUALE DELLE ISTRUZIONI DI PANDA-ROBOT CESARE			
VAI AVANTI DI UNA CASELLA			
GIRA A SINISTRA			
GIRA A DESTRA			

In questo manuale non adottiamo fin da subito una definizione rigorosa delle istruzioni, in particolare quando l'esecutore è un essere umano. Questa scelta è intenzionale. Da un lato, il linguaggio naturale è intrinsecamente ambiguo e dipendente dal contesto; dall'altro, lavorando con alunne e alunni molto piccoli, non è realistico né pedagogicamente opportuno richiedere da subito un livello elevato di precisione formale.

La progressione proposta (dagli algoritmi quotidiani fino a esecutori veri e propri come Code.org e Scratch, passando per attività unplugged sulla griglia) ha proprio lo scopo di accompagnare gradualmente gli alunni e le alunne nello sviluppo della precisione espressiva e procedurale. Il passaggio a esecutori automatici rende progressivamente esplicite le ambiguità del linguaggio informale e introduce, in modo naturale e motivato, la necessità di istruzioni non ambigue, complete e verificabili.

Una volta definito il *Manuale delle istruzioni*, è possibile combinarle per ottenere un **programma**, cioè una sequenza finita di istruzioni che formalizzano l'elaborazione che vogliamo descrivere e far eseguire all'esecutore.

In particolare, useremo la parola programma anche quando le istruzioni non sono ancora espresse con piena precisione formale, soprattutto nelle attività in cui l'esecutore è un essere umano. In questi casi, il programma non è ancora un oggetto completamente privo di ambiguità, ma rappresenta comunque un tentativo intenzionale di descrivere un processo in modo sufficientemente chiaro da poter essere seguito ed eseguito da un esecutore esterno.

Elementi comuni

Sebbene ogni esecutore abbia un diverso insieme di istruzioni (espresse in un certo linguaggio) che sa eseguire, ci sono alcuni elementi che accomunano la maggior parte dei linguaggi di programmazione pensati per la didattica⁶.

Sequenza

Le istruzioni date a un esecutore vengono eseguite nella sequenza in cui sono state scritte o fornite a esso. Se ci immaginiamo di dare un'istruzione alla volta (es. dire "fai 10 passi" e poi aspettare che il robot li compia davvero prima di dare l'istruzione successiva), questo appare ovvio. Ma solitamente quando si programma si scrivono tante istruzioni, e solo in seguito si chiede al computer di eseguirle tutte, dalla prima all'ultima, per poi osservare e valutare il risultato dell'elaborazione.

⁵ Sappiamo che da un punto di vista grammaticale l'uso di "delle" può suonare un po' strano. Tuttavia, il manuale spiega proprio come funzionano le istruzioni e non come svolgere l'attività. Per questa ragione, abbiamo scelto l'espressione *Manuale delle istruzioni*.

⁶ Solitamente sono linguaggi di programmazione *imperativi*, dove appunto si "impera", cioè si impartisce all'elaboratore una sequenza di istruzioni (di "ordini") da seguire ("fai questo, fai quello!").

Quello della sequenza può apparire un concetto semplice, e infatti è uno dei primi che si apprendono - ma è importante riflettere sul fatto che spesso cambiare l'ordine delle istruzioni produce un risultato diverso.

Consideriamo, per esempio, la sequenza di istruzioni:

1. METTI IL LIBRO DI MATEMATICA NELLO ZAINO
2. METTI IL QUADERNO DI ITALIANO NELLO ZAINO
3. METTI L'ASTUCCIO NELLO ZAINO
4. CHIUDI LA CERNIERA DELLO ZAINO
5. INDOSSA LO ZAINO

Alcune istruzioni (1-3) si possono scambiare senza problemi, mentre altre (4 e 5) devono rimanere in un certo ordine, altrimenti l'effetto sarà diverso.

■ Selezione

A volte vogliamo che il computer esegua certe istruzioni solo se si verifica una certa condizione (ecco perché il nome *selezione*: il computer “sceglie” cosa fare in base alla condizione).

La **condizione** è una domanda a cui si può rispondere con “vero” o “falso”. Ovviamente, la condizione non può lasciare spazio a interpretazioni; deve essere insindacabilmente vera o falsa: una condizione del tipo “se c'è molta gente” non è una condizione precisa.

Una selezione ci permette quindi di dare sequenze di istruzioni del tipo:

SE PIOVE:

PRENDI L' OMBRELLO

La condizione “piove” viene valutata: se è vera, si esegue l'istruzione **PRENDI L' OMBRELLO**, altrimenti non si esegue.

Potremmo voler indicare un'istruzione alternativa nel caso in cui la condizione sia falsa, come

SE HAI FINITO I COMPITI:

ESCI A GIOCARE

ALTRIMENTI:

FINISCI I COMPITI

L'istruzione **ESCI A GIOCARE** verrà eseguita solo se la condizione **HAI FINITO I COMPITI** è vera; l'istruzione **FINISCI I COMPITI** verrà eseguita solo se la condizione è falsa.

■ Ripetizione

Spesso usiamo i computer per automatizzare compiti ripetitivi. Capiterà spesso di dover far eseguire a un computer la stessa istruzione più e più volte.

Per esempio, potremmo dare queste istruzioni

RIPETI 2 VOLTE:

FAI UN SALTO

RIPETI 2 VOLTE:

FAI LA GIRAVOLTA

GUARDA IN SU

GUARDA IN GIÙ

DAI UN BACIO A CHI VUOI TU

In questo caso abbiamo dovuto specificare

- quali istruzioni vanno ripetute, utilizzando il **rientro** (detto anche **indentazione**): le istruzioni scritte un po' più a destra rispetto a **RIPETI** sono quelle che il computer deve ripe-

tere; quando il rientro finisce, finisce anche la ripetizione e quindi le istruzioni successive non vengono ripetute.

- quante volte vanno ripetute: prima si ripete **FAI UN SALTO** due volte, poi si ripete **FAI LA GIRAVOLTA** due volte.

Potrebbe capitare di non sapere prima quante volte vogliamo che il computer ripeta un'istruzione. Ecco che possiamo usare una seconda, più potente, forma di ripetizione: **l'iterazione con condizione**.

Potremmo per esempio dire:

```
VAI NEL CAMPO
FINCHÉ IL CESTO NON È PIENO:
    RACCOGLI UNA MELA
    METTI LA MELA NEL CESTO
TORNA A CASA
```

In questo caso abbiamo dovuto indicare:

- quali istruzioni vanno ripetute (è una sequenza: prima **RACCOGLI UNA MELA** poi **METTI LA MELA NEL CESTO**);
- la condizione (**IL CESTO NON È PIENO**) che, finché è vera, ci dice di non fermarci: quando sarà falsa, smetteremo di ripetere le due istruzioni e passeremo alla successiva in sequenza.

Scrivere tutti i programmi possibili

Sequenza, selezione e ripetizione sono dette **strutture di controllo** perché ci permettono di **controllare il flusso di esecuzione**, cioè di decidere quali istruzioni vengono eseguite e in che ordine, eventualmente modificandolo durante l'esecuzione.

Non sono solo strumenti utili, ma **i mattoni fondamentali di ogni programma informatico**: è stato dimostrato che, combinando queste tre strutture (insieme a una memoria “grande a piacere” e a operazioni di base per leggere, modificare e confrontare i dati in essa contenuti), è possibile scrivere qualsiasi programma che ci venga in mente. È un risultato affascinante, perché ci mostra come un'idea così semplice, di cui è possibile fare esperienza, ad esempio, con semplici giochi su una griglia, sia in realtà la base di tutta l'elaborazione automatica che ci circonda.

Debug

Perché nel paragrafo precedente abbiamo scritto “programmi possibili”? Perché gli informatici hanno dimostrato che esistono elaborazioni che non possono essere espresse come programmi. Questo è un risultato fondamentale, ma è fuori dalla portata degli alunni e delle alunne. Però ha una conseguenza che invece è molto rilevante per chiunque si approccia alla programmazione: una delle cose che i computer non potranno mai fare automaticamente è verificare se, in generale, un programma è corretto o meno.

Sta quindi al programmatore **“provare” (“testare”) un programma per verificare se, in diverse situazioni, si comporta come dovrebbe**. Fa parte della pratica di un programmatore **trovare e correggere gli errori, i “bug”** (quindi fare **“debugging”**) e procedere iterativamente, attraverso prove ed errori e raffinamenti successivi, per migliorare il proprio prodotto.

I computer eseguono sempre fedelmente tutte e sole le istruzioni che ricevono, quindi, anche se a volte ci piace pensare il contrario, i bug derivano sempre da un errore umano

(a meno che i circuiti elettronici non siano rotti, ovviamente). Chi impara a programmare si abitua a non abbattersi e a dare la caccia ai *bug* per migliorare sempre.

E GLI ALGORITMI?

Oggi, quando negli organi di comunicazione si parla di “algoritmi”, si fa spesso riferimento ad algoritmi di intelligenza artificiale, che prendono decisioni o influenzano le nostre scelte (per esempio, decidendo quale contenuto mostrarci sui social). In realtà, in quei casi gli algoritmi sono più che altro quelli che hanno permesso di addestrare tali intelligenze artificiali su un’ampia quantità di dati.

Qui offriamo una prospettiva molto più generale - in linea con le Indicazioni Nazionali 2025 [MIM, 2025], che definiscono, in ambito informatico, gli algoritmi come “descrizioni precise e non ambigue (in riferimento a uno specifico esecutore) di procedure (per raggiungere [...] obiettivi) che si prestano ad essere automatizzate.”

Spesso questi obiettivi da raggiungere sono anche chiamati **problemi**. A scuola può capitare che si chiamino “problemi” esercizi del tipo:

*Oggi alla mensa della scuola sono stati portati 60 yogurt alla banana, 45 yogurt alla fragola e 38 yogurt all’ananas. Quanti yogurt sono stati portati complessivamente alla mensa?*⁷

Questo, da un punto di vista informatico, è un **problema specifico**, perché i dati sono fissati [Crippa et al., 2025]. La **soluzione** è un numero: 143. Per trovarla, abbiamo sommato 60, 45 e 38.

Gli informatici, però, non vogliono risolvere il problema in prima persona: vogliono farlo risolvere a un computer. Si tratta di un modo specifico di affrontare i problemi, chiamato “problem solving computazionale”. E, se possibile, vogliono risolverlo “una volta per tutte”, per tutti i possibili dati di partenza. Per esempio, potrebbero voler affrontare il problema seguente:

*Ogni giorno, alla mensa della scuola vengono portati **un certo numero di** yogurt alla banana, **un certo numero** alla fragola e **un certo numero** all’ananas. Il responsabile della mensa vuole calcolare **ogni giorno** quanti yogurt ha ricevuto in totale.*

Questo è un **problema generale**, perché i dati possono assumere un qualsiasi valore (tra quelli accettabili nella data situazione) [Crippa et al., 2025].

La **soluzione** a questo problema non può essere un numero (dipende da quanti yogurt vengono portati) ma **un procedimento** da svolgere: un **algoritmo**. Per esempio:

1. CHIEDI QUANTI YOGURT ALLA BANANA SONO STATI RICEVUTI OGGI
2. CHIEDI QUANTI YOGURT ALLA FRAGOLA SONO STATI RICEVUTI OGGI
3. CHIEDI QUANTI YOGURT ALL’ANANAS SONO STATI RICEVUTI OGGI
4. FAI L’ADDIZIONE TRA IL NUMERO DI YOGURT ALLA BANANA, QUELLO DEGLI YOGURT ALLA FRAGOLA E QUELLO DEGLI YOGURT ALL’ANANAS
5. COMUNICA IL RISULTATO

⁷ Laura Valdiserra. Qua la ZAMPA! Matematica 3. Pagina 37.

Questo algoritmo è espresso in modo da poter essere tradotto in un **programma** scritto in un linguaggio di programmazione. Ad esempio, scritto in Scratch:



Ogni giorno, il responsabile potrà eseguire sul proprio computer questo programma senza più pensare a quale sia il procedimento: il computer eseguirà fedelmente le istruzioni al posto suo, gli chiederà in input i dati, e gli mostrerà il risultato.

Nella pratica scolastica può capitare di utilizzare il termine **algoritmi** per indicare solo le procedure che le alunne e gli alunni eseguono a volte meccanicamente (es. l'algoritmo per eseguire la divisione in colonna), cosa che può veicolare una visione procedurale della conoscenza lontana dalla comprensione concettuale e del pensiero critico [Crippa et al., 2025]. Questo è molto lontano dal senso informatico: infatti, gli informatici non hanno interesse a eseguire procedure sempre uguali e ripetitive - esattamente per questo ci sono i computer! Gli informatici sono molto più interessati a **inventare algoritmi** che risolvano problemi generali che si prestano poi ad essere automatizzati.

Programmatore o esecutore?

Se in generale gli informatici non amano essere "esecutori", per imparare l'informatica può essere utile, a volte, "fare finta di essere un computer", cioè eseguire alla lettera (secondo una precisa semantica) solo un insieme definito e limitato di istruzioni. Questo aiuta, quando si assumerà invece il ruolo di programmatori, a descrivere le elaborazioni che ci interessano in modo preciso e tenendo conto di ogni aspetto, senza lasciare nulla di implicito.

In alcune attività didattiche per l'introduzione dell'Informatica a scuola, però, viene posto l'accento solo sul ruolo di esecutore. È importante che le alunne e gli alunni riescano, a un certo punto, a costruire i propri programmi.



SUL QUADERNO DI INFORMATICA

Nel Quaderno di Informatica, esplicitiamo tramite due icone diverse il ruolo che assumeranno le alunne e gli alunni nelle diverse attività proposte.



l'alunno/a ha ruolo di esecutore



l'alunno/a ha ruolo di programmatore

RIFERIMENTI

- Violetta Lonati e Mattia Monga. Rendere concreta l'informazione per elaborarla. Nuova Secondaria. 43:3(2025), pp. 74-79.
- Ministero dell'Istruzione e del Merito (2025). Indicazioni Nazionali per il curricolo. Scuola dell'infanzia e Scuole del Primo ciclo di istruzione.
- Marta Crippa, Violetta Lonati, Luca Lamanna, Laura Branchetti. Educare al pensiero algoritmico: riflessioni e proposte per il primo ciclo. ITADINFO 2025. pp. 19-27. Liberamente scaricabile da <https://q3.hubscuola.it/x80>

CAPITOLO 3:

ALGORITMI QUOTIDIANI

In questo capitolo proponiamo attività didattiche per una lettura algoritmica di molte delle attività che facciamo ogni giorno e che seguono una sequenza di passi. Infatti, per vestirci, cucinare o realizzare un modello con i mattoncini LEGO®, compiamo azioni che, se osservate dall'esterno, assomigliano molto a un algoritmo. Esploreremo come descrivere algoritmi relativi ad attività quotidiane in modo chiaro, preciso e non ambiguo, mettendo in luce come ciò che per chi descrive sembra "ovvio" possa essere interpretato in modi diversi da chi ascolta.

Dal punto di vista informatico, quando un algoritmo viene espresso in un linguaggio che il computer può "capire", viene chiamato **programma**. Il computer esegue il programma solo quando glielo "chiediamo". Il computer eseguirà solo ciò che è stato scritto, alla lettera e passo dopo passo, senza intuire o indovinare ciò che non è stato esplicitato e senza fare domande intermedie. Se durante o al termine dell'esecuzione qualcosa non ha funzionato, il programmatore, attraverso la pratica del debugging (si veda Capitolo 1, pag. 20), potrà esaminare il comportamento dell'esecutore, individuare l'errore e modificare la sequenza.

Le attività di questa sezione propongono alcuni esempi di pratiche didattiche che incarnano le idee mostrate nei video del Capitolo 1 (sezione Pensiero computazionale), cioè la sfida dell'*Exact Instructions Challenge* per il panino con il burro di arachidi e il video del maestro Manzi sul lavarsi i denti. Potrete poi sbizzarrirvi nel progettare molte altre attività simili declinandole nel modo più opportuno alle vostre classi e scegliendo ambientazioni che diano significato e motivazione (ad esempio le regole di un gioco motorio, un laboratorio artistico, una storia inventata da alunni e alunne).

TRAGUARDI E OBIETTIVI GENERALI

- Scomporre un compito complesso in una sequenza di istruzioni elementari e ben definite.
- Ordinare le istruzioni in modo coerente.
- Utilizzare un linguaggio chiaro, preciso e non ambiguo per descrivere istruzioni.
- Immedesimarsi nel punto di vista dell'esecutore, anticipando possibili fraintendimenti.
- Riconoscere e correggere ambiguità nelle istruzioni.
- Attuare pratiche di debugging.
- Progettare e utilizzare un linguaggio condiviso per descrivere azioni.

PREREQUISITI

Le attività proposte non richiedono particolari conoscenze informatiche pregresse: è sufficiente che alunni e alunne sappiano seguire e dare istruzioni in linguaggio naturale e ordinare nel tempo le fasi di una storia o di un'azione quotidiana. Per l'Attività 3 è utile che abbiano già incontrato le principali forme geometriche e abbiano avuto qualche esperienza di composizione con figure usando costruzioni di vario tipo.

Alcune attività sono progettate per essere proposte a chi non ha ancora raggiunto un livello di apprendimento della letto-scrittura adeguato ma, una volta che tali abilità strumentali saranno acquisite, sarà possibile proporre richieste più articolate e sfidanti.

CONTENUTI, IN BREVE

Questo capitolo contiene attività pensate per capire cosa significhi sviluppare una lettura algoritmica di un compito: scomporlo in istruzioni essenziali, metterle in ordine e formularle con precisione. Riprendendo quanto visto nel Capitolo 2 sulle istruzioni, vedremo come:

- definire le **istruzioni** elementari per raggiungere un certo obiettivo;
- stabilire la **sequenza** corretta di istruzioni, cioè come ordinarle per ottenere un certo effetto;
- **esprimere le istruzioni** in modo chiaro e non ambiguo.

Attività 1 – Sequenze e ordinamenti: alunni e alunne scompongono azioni o storie in istruzioni elementari, le mettono in sequenza e sperimentano cosa cambia (o non cambia) quando si altera l'ordine, per capire perché in un algoritmo l'ordine delle istruzioni è fondamentale, e talvolta sequenze diverse possono produrre lo stesso risultato.

Attività 2 – Istruzioni e programmi: attraverso la richiesta di eseguire programmi dati e poi di inventarne di nuovi si introduce l'idea di *Manuale delle istruzioni* che definisce “quali istruzioni esistono e che cosa fanno”.

Attività 3 – Descrivere configurazioni geometriche: far ricostruire una configurazione geometrica a un esecutore umano che segue le istruzioni alla lettera (senza interpretare né fare domande) e migliorare iterativamente le descrizioni tramite debugging, per arrivare a un linguaggio condiviso e rigoroso.

MODALITÀ DI LAVORO

Tutte le attività che seguono possono essere svolte in classe.

- Si alterneranno fasi di lavoro a coppie con momenti di confronto collettivo o a piccoli gruppi.
- Lo spazio deve permettere di lavorare a coppie in modo che i membri di ciascuna coppia possano comunicare tra loro agevolmente.
- Sarà importante osservare lo svolgimento delle varie attività tenendo traccia delle parole e delle produzioni di alunni e alunne (ad esempio, fornendo schede strutturate per scrivere o disegnare, ma anche scattando foto alle loro produzioni) da usare a supporto di discussioni collettive.

In tutte le proposte è cruciale mantenere netta la distinzione tra i ruoli di programmatore ed esecutore. Il **programmatore** ha il compito di progettare e scrivere l'intera sequenza di istruzioni prima dell'esecuzione, cercando di prevedere come verranno interpretate. L'**esecutore**, invece, deve limitarsi a seguire le istruzioni alla lettera, senza “aggiustare” ciò che non è stato specificato e senza introdurre interpretazioni personali.

Questa distinzione rende visibile un'idea chiave: l'output di un programma non è “l'intenzione” di chi lo ha scritto, ma ciò che è effettivamente espresso nelle istruzioni disponibili e nel loro ordine.

Un ulteriore punto di attenzione riguarda il modo in cui le istruzioni vengono formulate: il linguaggio naturale è intrinsecamente ambiguo e dipende dal contesto, dalle conoscenze e dalle inferenze di chi ascolta o legge. Proprio per questo, molte attività mirano a far emergere ambiguità tipiche (“quanto?”, “dove?”, “rispetto a cosa?”, “con quale orientamento?”) e a costruire progressivamente convenzioni condivise (ad esempio un *manuale delle istruzioni* o un lessico comune per descrivere posizioni e relazioni tra figure). Questo passaggio prepara il terreno per comprendere perché in informatica si introducono **linguaggi formali**: essi non

eliminano la complessità del problema, ma rendono esplicite le regole di scrittura e di interpretazione, così che la stessa istruzione, nello stesso contesto formale, venga interpretata dall'esecutore in modo sempre uguale e prevedibile.

ATTIVITÀ 1: Sequenze e ordinamenti

→ Vedi Quaderno di Informatica pp. 2-3

Come abbiamo visto nel Capitolo 2, le istruzioni date a un elaboratore vengono eseguite in sequenza e cambiarne l'ordine produce, in generale, risultati diversi. Quindi, sarà importante proporre attività in cui alunni e alunne possano riflettere sull'importanza di seguire o rispettare un ordine preciso quando, ad esempio, si scompongono attività quotidiane o storie in una sequenza di momenti.

Preparazione

- Stampare e ritagliare diverse immagini che raffigurano momenti di un'attività o scene di una storia.
- Post-it (per disegnare o scrivere istruzioni, numeri, parole chiave).

Descrizione di proposte didattiche

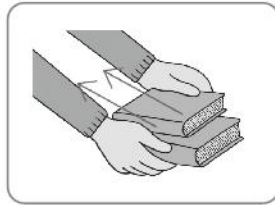
- **Riordinare delle immagini date.** In questo tipo di proposta a ciascuno viene consegnato un insieme di carte che raffigurano i diversi passi da fare per raggiungere un dato obiettivo (ad esempio avere i denti puliti, avere lo zaino pronto, andare a scuola) o di una storia. Le carte vengono inizialmente consegnate non ordinate e la richiesta è quella di metterle in ordine, anche eventualmente numerando i vari passi.
- **Disegnare immagini o scrivere istruzioni in ordine.** In questo tipo di proposta a ciascuno viene chiesto di scegliere una storia o un'attività, suddividerla in momenti ordinati e poi descrivere ciascun momento con un'immagine o un breve testo (ad esempio su singoli foglietti o post-it da disporre in sequenza).
- **Cambiare l'ordine e studiarne gli effetti.** In questo tipo di proposta a ciascuno viene richiesto di esplorare cosa accade cambiando l'ordine delle istruzioni di una sequenza data. Sarà importante proporre anche situazioni in cui scambiare l'ordine di alcune istruzioni non impatta sul risultato, così da poter incoraggiare riflessioni più profonde rispetto all'ordine delle istruzioni in una sequenza. Ad esempio, considerando l'attività quotidiana "lavarsi i denti", le istruzioni "metti il dentifricio sullo spazzolino" e "bagna lo spazzolino con un po' d'acqua" possono essere scambiate senza problemi mentre altre istruzioni della stessa sequenza (come "risciacqua" o "strofina lo spazzolino sui denti") non possono essere spostate liberamente senza modificare il significato dell'azione. Questo permette di introdurre, in modo intuitivo, l'idea che alcune istruzioni possono essere scambiate (sono indipendenti) mentre altre hanno un ordine vincolato.
- **Inventare storie "buffe".** Oltre a trovare la sequenza corretta, si può proporre una variante creativa: inventare "storie buffe" mescolando l'ordine delle scene. In questo caso, la sequenza prodotta diventa un dispositivo narrativo, un artefatto da raccontare, disegnare o persino mettere in scena. Gli errori o le sequenze "sbagliate" diventano risorsa: come suggerisce Rodari nella sua *Grammatica della fantasia*, l'ordine alterato può generare narrazioni impreviste e divertenti, insegnando a "smontare e rimontare" le storie come fossero costruzioni.



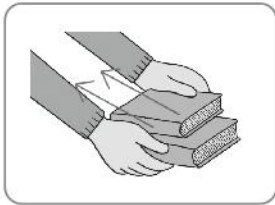
SUL QUADERNO DI INFORMATICA

Puoi proporre ai tuoi alunni e alunne le schede di pag. 2 e 3 RIORDINARE del Quaderno di Informatica.

Nell'attività proposta a pag. 2, ci sono due ordini corretti possibili. Uno è:



L'altro è:



Al termine dello svolgimento della scheda sarà importante avviare una riflessione collettiva sul fatto che possono esistere più sequenze corrette per svolgere la stessa attività.

Contenuto informatico

In informatica, la **sequenza di istruzioni** è la struttura di base su cui si costruisce ogni programma. Prima di introdurre concetti più complessi come le **selezioni** (*se-allora*) o le **ripetizioni** (*ripeti*), è essenziale che alunni e alunne comprendano a fondo cosa significhi eseguire azioni in un ordine preciso e senza ambiguità.

Le attività descritte permettono di lavorare su alcune idee chiave:

- Ogni azione deve essere descritta in modo esplicito, senza sottintesi;
- L'ordine conta: modificare la sequenza può cambiare radicalmente il risultato;
- Programmare richiede pianificazione: occorre pensare prima di agire e prevedere cosa farà l'esecutore;

Acquisire familiarità con la sequenza e con la capacità di pianificare è il primo passo della programmazione: solo dopo aver consolidato questi aspetti sarà possibile introdurre in modo efficace altre strutture fondamentali.

ATTIVITÀ 2: Istruzioni e programmi

→ Vedi Quaderno di Informatica pp. 4-5

Un programma è una sequenza di istruzioni, scritte in un qualche linguaggio di programmazione. Prima di affrontare l'idea di "linguaggio di programmazione", e dopo aver lavorato su ordine e sequenza, sarà importante lavorare sulla formulazione di istruzioni da mettere in sequenza per scrivere "programmi"¹. In questo insieme di attività viene introdotto il *Manuale delle istruzioni* che, idealmente, contiene l'elenco di tutte e sole le istruzioni disponibili, ciascuna accompagnata dalla descrizione precisa e univoca del suo effetto.

¹ Come anticipato nel Capitolo 2, per poter parlare di programmi veri e propri è necessario che l'esecutore sia meccanico e non umano. In questo capitolo, a partire dall'Attività 2, sarà possibile parlare intuitivamente di programmi solo a patto che chi assumerà il ruolo dell'esecutore si comporti "come se fosse un computer" ed esegua alla lettera le istruzioni per come descritte nel *Manuale delle istruzioni*.

Preparazione

- Istruzioni già stampate (per attività motorie o di altro tipo).
- Post-it per scrivere istruzioni.

Descrizione di possibili attività

- **Eseguire programmi dati.** In questo tipo di proposta vengono forniti dei programmi costituiti da sequenze di istruzioni che si riferiscono, ad esempio, ad attività motorie da eseguire (come brevi balletti o combinazioni di gesti). Ad alunni e alunne viene richiesto di eseguire il programma.

Dopo averlo eseguito, può essere utile proporre di compilare insieme il *Manuale delle istruzioni* in cui esplicitare quali sono le istruzioni disponibili (o usate nel programma) e, per ciascuna, descrivere che effetto ha. Si potrebbe impostare il *Manuale delle istruzioni* come una tabella del tipo:

Manuale delle istruzioni per...	
Istruzione	Effetto

Ad esempio, considerando il programma scritto sul Quaderno di Informatica a pag. 5:

il *Manuale delle istruzioni* conterrà:

- nella colonna a sinistra, le singole istruzioni separate, cioè: “ripeti 3 volte”, “batti le mani”, “batti i piedi”, “fai un saltino”, “dire coccodè”;
- nella colonna a destra, la spiegazione verbale di cosa significa ciascuna istruzione. Questo passaggio è tutt’altro che ovvio e sarà necessario esplicitare molti aspetti. Ad esempio, per l’istruzione “batti le mani”: quante volte? E vanno battute tra loro o su un tavolo? E cosa significa “batti i piedi”? Non ci sono risposte corrette a priori: basta solo cercare di formulare la descrizione delle istruzioni nel modo più preciso e meno ambiguo possibile.



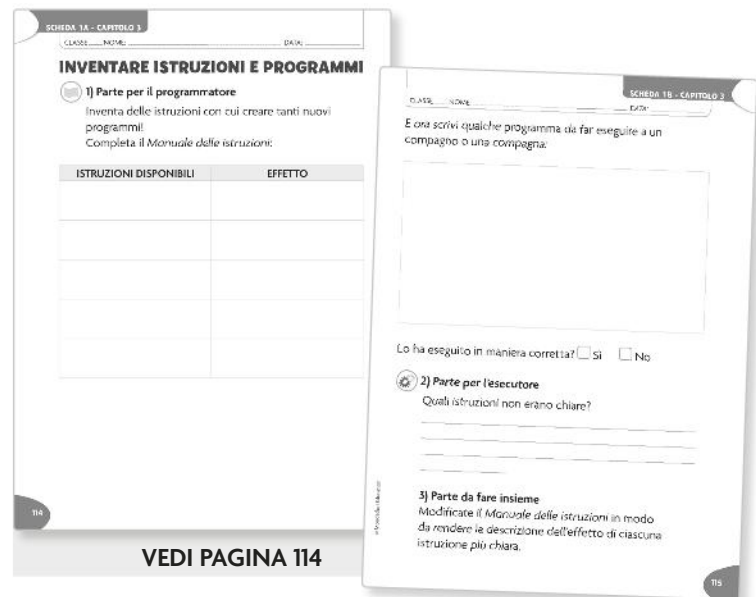
Consigliamo di realizzare la tabella come attività collettiva in classe in modo da poter raccogliere tutte le proposte, condividerle con alunni e alunne e avviare una riflessione per arrivare a un *Manuale delle istruzioni* condiviso.

- **Date le istruzioni, inventare programmi.** In questo tipo di proposta viene fornito l’insieme delle istruzioni disponibili (sotto forma di *Manuale delle istruzioni*) e si chiede ad alunni ed alunne di inventare dei programmi. Si può proporre il lavoro a coppie, in modo che a turno ciascuno possa comportarsi da esecutore del programma del compagno o della compagna. Al termine dell’attività sarà importante raccogliere e commentare tutti i diversi programmi per rendere esplicito che, dato un piccolo insieme di istruzioni, con queste si possono scrivere moltissimi programmi diversi.
- **Inventare istruzioni e programmi.** In questo tipo di proposta si chiede ad alunni ed alunne di inventare un set di istruzioni per poter realizzare un’attività (ad esempio un balletto), scriverne il *Manuale delle istruzioni* e poi inventare almeno 2-3 programmi diversi come

sequenza delle proprie istruzioni. Anche in questo caso è utile il lavoro a coppie, in modo che ciascuno a turno possa comportarsi da esecutore del programma del compagno o della compagna.

Con alunni ed alunne più grandi, può essere utile usare delle schede strutturate (come quella a pagina 114) in cui:

- i programmatori scrivono il proprio *Manuale delle istruzioni* e i propri programmi;
- gli esecutori danno un feedback sulla chiarezza delle istruzioni;
- alla fine, insieme, apportano modifiche al *Manuale delle istruzioni* per riflettere su come descrivere le istruzioni disponibili in maniera chiara, precisa e priva di ambiguità.



Al termine dell'attività sarà importante raccogliere e commentare i vari *Manuali delle istruzioni* per riflettere collettivamente su come descrivere le istruzioni disponibili.



SUL QUADERNO DI INFORMATICA

Puoi proporre ai tuoi alunni e alunne le schede di pag. 4 e 5 ISTRUZIONI E PROGRAMMI del Quaderno di Informatica.

Come attività preparatoria alle proposte di pag. 4 (in cui si chiede di far scrivere a qualcun altro le istruzioni per fare un panino al cioccolato per poi eseguirle per valutare se sono precise) si può mostrare e commentare in classe il video del Maestro Manzi².

Nella prima attività proposta a pag. 5 la scelta grafica per rappresentare le istruzioni è stata fatta per richiamare l'estetica delle istruzioni di un linguaggio visuale a blocchi (es. Scratch).

Contenuto informatico

In questa attività il *Manuale delle istruzioni* gioca metaforicamente il ruolo della **sintassi** (le regole del linguaggio) e della **semantica** (il significato delle istruzioni) nei linguaggi di programmazione: definisce quali istruzioni esistono e che cosa fanno. I programmi sono le sequenze di istruzioni scritte usando esclusivamente le istruzioni presenti nel *Manuale delle istruzioni*, mentre l'esecutore rappresenta il computer, che esegue il programma alla lettera senza poter fare domande né "indovinare" ciò che non è stato scritto.

Le attività proposte permettono di lavorare su alcune idee chiave dell'informatica:

- un insieme limitato di istruzioni può generare moltissimi programmi diversi;

² <https://q3.hubscuola.it/5zg> (DAL MINUTO 8:44)

- ogni istruzione deve avere un effetto descritto in modo esplicito e non ambiguo, perché l'esecutore esegue solo le istruzioni che riceve (se in una sequenza che descrive come fare un panino al cioccolato manca, ad esempio, "apri il barattolo", l'esecutore può limitarsi ad appoggiare il barattolo chiuso sul pane);
- per programmare serve pianificare: chi scrive il programma deve prevedere in anticipo come verrà interpretata ogni istruzione da chi esegue;
- il debugging è un processo ciclico di prova, osservazione e revisione: gli errori non sono un fallimento, ma un'occasione per migliorare.

ATTIVITÀ 3: Descrivere configurazioni geometriche

→ Vedi Quaderno di Informatica pp. 6-7

La Fase 1 di questa attività può essere svolta anche con alunni e alunne più giovani (classi 1° e 2°). La Fase 2 richiede invece di scrivere un testo e quindi è più adatta alla classe 3°.

Preparazione

- Stampare e ritagliare un tangram di carta per ciascuno (scheda nella guida digitale).
- Procurarsi anche altri tipi di forme geometriche di carta o altro materiale (es. legno o plastica).

Fase 1 - descrizione verbale

Obiettivo

Accordarsi su un linguaggio condiviso per disambiguare istruzioni date.

Descrizione

Alunni e alunne vengono divisi in coppie³ e a ciascun componente viene consegnato un tangram di carta. Le coppie si dispongono in modo da potersi ascoltare tra loro senza però vedere quello che ciascuno ha davanti a sé. Quindi, ad esempio, potranno sedersi di spalle oppure di fronte ma mettendo una sorta di "separè" in mezzo.

Si stabiliscono due ruoli, quello di **programmatore** e quello di **esecutore**.

La Fase 1 consiste di **3 diversi turni di gioco, ciascuno ripetuto due volte** per permettere ad alunni e alunne di scambiarsi i ruoli⁴. In ogni turno di gioco, la dinamica è sempre la stessa:

- chi ha il ruolo di programmatore dovrà creare una composizione di figure geometriche usando tutti e sette i pezzi del tangram;
- una volta creata la composizione, dovrà descriverla all'esecutore per fargliela ricreare esattamente identica. Non è permesso indicare, mimare o alzarsi per guardare il lavoro del compagno o della compagna: l'unico canale che si può usare è la parola;
- quando l'esecutore ritiene di aver terminato la sua ricostruzione, deve dichiararlo e la configurazione originale e quella ricostruita possono essere confrontate per valutare se sono esattamente identiche e sovrapponibili pezzo per pezzo oppure no.

³ Se il numero totale è dispari, si può formare un gruppo di 3.

⁴ Se c'è un gruppo da 3, si organizzano i turni di gioco in modo che ci sia un programmatore e due esecutori indipendenti tra loro.

Ciò che cambia tra un turno e l'altro è la possibilità di interazione da parte di chi ha il ruolo di esecutore:

- Nel **primo turno di gioco**, l'esecutore può fare al programmatore **qualsiasi domanda** di chiarimento che riterrà opportuna.
- Nel **secondo turno di gioco**, l'esecutore ha a disposizione **solo tre domande di chiarimento** da fare al programmatore.
- Nel **terzo turno di gioco**, l'esecutore **non potrà fare alcuna domanda** al programmatore.

Durante i vari turni di gioco, l'insegnante potrà girare tra le varie coppie e, dopo aver osservato gli scambi comunicativi senza intervenire, potrà facilitare l'emergere delle prime riflessioni ponendo delle domande stimolo alla fine di ciascun turno.



DOMANDE STIMOLO

- Siete riusciti sempre a capirvi?
- Quando non riuscivate a capirvi, da cosa dipendeva secondo voi?
- (Rivolta al programmatore) Quali configurazioni hai trovato più difficili da descrivere?
- (Rivolta all'esecutore) Quali istruzioni hai trovato più difficili da capire? E che domande facevi per capirle meglio?
- Al prossimo turno di gioco a cosa farete maggiore attenzione per capirvi meglio?

Al termine della Fase 1, si può proporre una discussione collettiva per condividere le osservazioni rispetto all'esperienza vissuta e raccogliere le caratteristiche di eventuali pattern di ambiguità emersi nei vari turni di gioco.

Contenuto informatico

Questa prima fase permette di distinguere diversi livelli di precisione con cui possono essere fornite istruzioni e prepara il terreno per la fase successiva. Inoltre, la progressiva riduzione delle domande di chiarimento (fino a nessuna domanda nel terzo turno di gioco) sottolinea la differenza fra una comunicazione dialogica tra umani e l'esecuzione di un programma da parte di un computer, che non può chiedere spiegazioni durante l'esecuzione.



SUL QUADERNO DI INFORMATICA

Come attività preparatoria o rafforzativa, puoi proporre ai tuoi alunni e alunne la scheda di pag. 6 DESCRIVERE UNA FIGURA del Quaderno di Informatica.

Soluzione: Tra Alice e Bob, è Alice a dare una descrizione che permette di individuare in modo univoco la figura corretta tra le due proposte.

Fase 2 - descrizione scritta

Obiettivo

Disambiguare la descrizione di una configurazione geometrica attraverso una pratica ispirata al debugging.

Descrizione

Alunni e alunne vengono divisi in un numero pari di gruppi da 3-4 persone ciascuno. Si mettono a disposizione della classe nuovi set di forme geometriche varie che siano diversi dal

tangram⁵. Ciascun gruppo sceglie un set di almeno 5 nuove forme con cui realizzare la propria configurazione geometrica.

I vari gruppi si dispongono nello spazio dell'aula così da riuscire a lavorare in modo abbastanza riservato rispetto agli altri (ad esempio creando un'"isola di lavoro" per ciascun gruppo).

La Fase 2 si divide in tre parti. La prima parte è preparatoria ed è dedicata alla realizzazione della configurazione geometrica da descrivere. La seconda parte è dedicata alla scrittura del testo che descrive la propria configurazione. La terza è dedicata al debugging del testo che avviene osservando un altro gruppo mentre lo interpreta e ricostruisce la figura seguendo le istruzioni.

- **Preparazione della configurazione geometrica.** Come prima cosa, ogni gruppo crea con i pezzi scelti la propria configurazione geometrica e la disegna in un foglio pezzo per pezzo: questo disegno andrà piegato e nascosto perché rappresenterà la soluzione.
- **Scrittura del testo.** Ogni gruppo dovrà scrivere un testo, il più preciso e meno ambiguo possibile, che contenga le istruzioni affinché un altro gruppo possa, seguendole alla lettera, ricostruire esattamente la stessa configurazione.

Ci sono due aspetti importanti da considerare:

- il testo **non può contenere disegni ma solo istruzioni verbali**;
- le informazioni scritte nel testo sono tutto ciò che il gruppo che deve ricostruire la configurazione avrà a disposizione: **chi ricostruisce non potrà fare domande**.

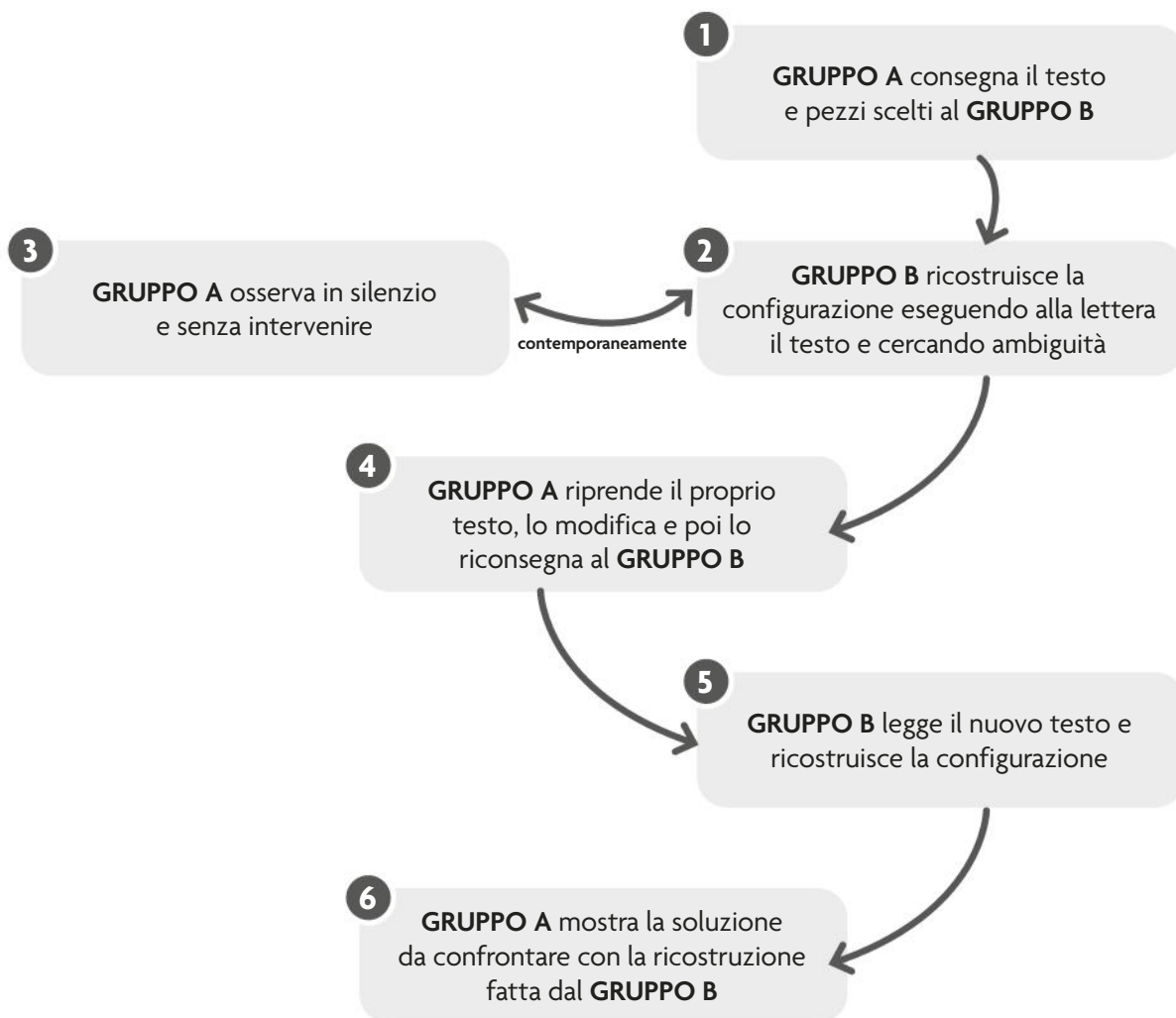
Una volta terminata la scrittura del testo, sarà importante smontare la configurazione geometrica costruita per non farla vedere a nessuno.

- **Debugging.** Quando due gruppi (che per comodità chiamiamo A e B) hanno terminato la scrittura del proprio testo:
 1. il gruppo A si sposta nella postazione del gruppo B e consegna il proprio testo e i pezzi scelti;
 2. il gruppo B dovrà seguire alla lettera le istruzioni scritte dal gruppo A e ricostruire la configurazione. Nel leggere e interpretare il testo, i componenti del gruppo B dovranno **eseguire esattamente ciò che c'è scritto andando "a caccia" di ambiguità**, cioè chiedendosi in ogni passaggio: *quante configurazioni diverse si possono descrivere con queste istruzioni?*
 3. nel frattempo, il gruppo A osserva in silenzio e senza intervenire il gruppo B mentre legge, interpreta il testo ed esegue la costruzione;
 4. quando il gruppo B termina la ricostruzione, il gruppo A può riprendere il proprio testo e modificarlo alla luce di quello che ha osservato durante l'interpretazione del gruppo B;
 5. il gruppo A può riconsegnare il testo modificato al gruppo B che farà un nuovo giro di lettura e ricostruzione;
 6. al termine della seconda ricostruzione del punto 5, il gruppo A mostrerà il disegno-soluzione da confrontare con la figura ricostruita dal gruppo B.

L'immagine che segue schematizza i passaggi da 1 a 6; poi, si ricomincia ripetendo i passaggi da 1 a 6 ma scambiando i due gruppi (quindi il gruppo A sarà l'interprete del testo scritto dal

⁵ Potranno essere nuove sagome di carta o cartone o altri tipi di costruzioni in plastica o legno. Per avere pezzi a sufficienza per tutti i gruppi, sarà necessario averne un buon numero.

gruppo B, che osserverà in silenzio per poi modificare il proprio testo sulla base di quanto osservato).



In questa Fase 2 dell'attività il testo scritto dal gruppo gioca il ruolo del programma, mentre il gruppo che lo legge e costruisce la figura gioca il ruolo dell'esecutore.



ASPETTI A CUI FARE ATTENZIONE

Perché tutto funzioni al meglio, è importante chiarire che:

- l'obiettivo comune dei due gruppi non è quello di ricostruire la figura "giusta" ma scrivere testi e istruzioni il più precise e meno ambigue possibili. In particolare, quando si scrive il testo, l'obiettivo è evitare il più possibile ambiguità; quando si interpreta un testo, l'obiettivo è trovare più ambiguità possibili;
- si fa debugging del testo e non delle persone (quindi, le osservazioni su passaggi poco chiari dovranno sempre essere espresse in termini di istruzioni che si possono scrivere più chiaramente o precisamente e mai in termini di "compagni che non sanno spiegare").

Al termine della Fase 2 sarà possibile condurre una riflessione collettiva per condividere le osservazioni rispetto all'esperienza vissuta e raccogliere le strategie usate per chiarire e precisare meglio le descrizioni scritte.

**DOMANDE STIMOLO**

- Che differenze avete trovato tra la prima fase dell'attività e la seconda?
- Quando osservavate gli altri interpretare il vostro testo, è successo qualcosa che non vi aspettavate? Se sì, cosa?
- Quando osservavate gli altri interpretare il vostro testo, cosa vi è stato utile considerare per migliorarlo?
- Quali regole di scrittura avreste potuto usare per scrivere dei testi chiari e precisi?

Contenuto informatico

Il linguaggio naturale è ambiguo per natura e la sua interpretazione da parte delle persone dipende da chi sono quelle persone (dalle loro esperienze, da ciò che conoscono, eccetera). Tuttavia, l'attività appena descritta permette ad alunni e alunne di riflettere sull'importanza di stabilire un linguaggio convenzionale condiviso per esprimere le istruzioni in modo tale che chi le fornisce e chi le esegue le interpretino allo stesso modo. Porre attenzione su questo aspetto prepara il terreno per cogliere le caratteristiche dei linguaggi formali (linguaggi di programmazione) che sono ciò che permette e assicura che l'esecutore esegua una certa istruzione sempre allo stesso modo, senza ambiguità.

**SUL QUADERNO DI INFORMATICA**

Come attività preparatoria o rafforzativa, puoi proporre ai tuoi alunni e alunne la scheda di pag. 7 DESCRIVERE UNA FIGURA del Quaderno di Informatica.

POSSIBILI ESTENSIONI, VARIAZIONI E PERSONALIZZAZIONI**Spunti per adattare l'attività**

In caso di difficoltà:

- Diminuire il numero di elementi da gestire (es. il numero di immagini da riordinare o il numero di pezzi a disposizione per creare la configurazione geometrica).
- Invece di chiedere di scrivere dei testi, fornire opzioni di testi già scritti tra cui scegliere quello corretto.

Come rendere l'attività più sfidante:

- Aumentare il numero o il tipo di elementi da gestire (ad esempio, nella Fase 2 dell'attività *Descrivere configurazioni geometriche* si può proporre di lavorare su configurazioni tridimensionali).
- Proporre una storia "sbagliata" in cui prima individuare le parti principali per poi riordinarle.

ASPETTI INFORMATICI

In informatica, la sequenza di istruzioni è la struttura di base su cui si costruisce ogni programma. Prima di introdurre strutture più complesse come le selezioni (*se-allora*) o le ripetizioni (*ripeti*), è essenziale che gli alunni e le alunne comprendano a fondo cosa significhi eseguire azioni in un ordine preciso e senza ambiguità.

Queste attività sono particolarmente efficaci perché:

- evidenziano la differenza fra chi crea un programma (programmatore) e chi lo esegue (esecutore);

- mostrano che un algoritmo deve essere pensato per un esecutore che può eseguire solo un certo insieme di istruzioni ben definito;
- introducono il concetto di debugging come revisione e correzione della sequenza;
- allenano la pianificazione, cioè la capacità di prevedere in anticipo tutti i passi necessari, organizzarli in modo logico e verificarne la completezza.

A differenza di una conversazione, in cui è possibile correggere passo-passo ciò che fa l'esecutore, solitamente quando si programma la sequenza viene scritta interamente prima dell'esecuzione⁶. Questo richiede un pensiero anticipatorio: occorre immaginare lo svolgimento dell'intero compito, prevedere le criticità e predisporre le istruzioni necessarie a evitarle.

Queste esperienze permettono ad alunni e alunne di esplorare alcune idee chiave:

- **Ogni azione deve essere descritta in modo esplicito.**
- **L'ordine conta:** modificare la sequenza può cambiare radicalmente il risultato.
- **Programmare richiede pianificazione:** occorre pensare prima di agire e prevedere cosa farà l'esecutore.
- **Le regole condivise** sono indispensabili perché il messaggio sia interpretato nello stesso modo da chi lo scrive e da chi lo legge.
- **Il debugging è parte del processo:** provare, osservare cosa non funziona, correggere e ripetere.

Acquisire familiarità con la sequenza, con la necessità di precisione e con la capacità di pianificare è il primo passo nel percorso di programmazione: solo dopo aver consolidato queste competenze sarà possibile introdurre in modo efficace altre strutture fondamentali come le istruzioni condizionali o le ripetizioni.

RELAZIONE CON L'INFORMATICA NELLE INDICAZIONI NAZIONALI

Relativamente alle Indicazioni Nazionali 2025, queste attività contribuiscono allo sviluppo della seguente competenza.

Competenze (alla fine della scuola primaria) nella disciplina Matematica

- *Descrivere procedure per lo svolgimento di compiti pratici mediante algoritmi.*

Le attività proposte contribuiscono ai seguenti obiettivi.

Obiettivi (alla fine della terza primaria) nella disciplina Matematica

- *Descrivere a parole attività della vita quotidiana tramite sequenze di passi precisi e non ambigui e saperle eseguire.*
- *Scrivere semplici programmi e verificare, mediante la loro esecuzione, se svolgono il compito previsto ed eventualmente correggerli.*

Sebbene queste attività siano unplugged (senza computer), si va nella direzione di raggiungere linguaggi formali con una sintassi e una semantica ben definite e non ambigue. Per questo sono una buona preparazione per questo obiettivo, poiché sono orientate alla progettazione e scrittura di programmi, e alla loro valutazione (controllo di correttezza, debugging, ...).

Infine, lavorano sulle seguenti conoscenze (inserite nelle indicazioni a titolo di esempio, non prescrittive):

- *concetto di algoritmo e sua esecuzione rigorosa;*

⁶ Tralasciamo per il momento forme più interattive di programmazione, che si stanno diffondendo oggi.

- *strutture di controllo fondamentali di un linguaggio di programmazione* (In questo capitolo si lavora in particolare sulla sequenza);
- *controllo di correttezza dei programmi*.

VALORE COSTRUZIONISTA

Seguendo la prospettiva di Papert, l'apprendimento passa per la costruzione di artefatti significativi di vario tipo che possono essere mostrati, discussi e migliorati insieme. In questo processo, l'errore non è qualcosa da evitare ma è una risorsa, un'informazione preziosa che rende visibili ambiguità e sottintesi e spinge a rendere le istruzioni sempre più chiare. Sequenze ambigue, indicazioni incomplete o termini poco chiari diventano occasioni per riflettere, riformulare e affinare il proprio lavoro attraverso cicli di prova, osservazione e revisione (debugging). La progressione delle attività proposte in questo capitolo - dall'ordinamento di istruzioni fino allo stabilire un linguaggio convenzionale condiviso per *descrivere configurazioni geometriche* - mette in evidenza un approccio tipico in informatica: affrontare la complessità scomponendo i problemi e stabilendo regole condivise che rendano le procedure eseguibili in modo prevedibile. In questo modo alunni e alunne imparano a progettare, verificare e migliorare le proprie soluzioni.

SUGGERIMENTI PER ULTERIORI ATTIVITÀ E RIFERIMENTI

- Bebras: <https://q3.hubscuola.it/gvl>

Catalogo dei quesiti Bebras: <https://q3.hubscuola.it/nbu>

Bebras è un'iniziativa internazionale che mira a promuovere nelle scuole l'informatica come disciplina scientifica. Nata in Lituania nel 2004, prevede ogni anno (in genere a novembre) un concorso online non competitivo nei Paesi aderenti.

In Italia, Bebras è organizzato da **ALaDDIn**, il *Laboratorio di Didattica e Divulgazione dell'Informatica* del Dipartimento di Informatica dell'Università degli Studi di Milano. I quesiti usati nei giochi Bebras possono essere affrontati senza particolari conoscenze pregresse e possono diventare uno stimolo per successivi approfondimenti individuali o di classe.

Nell'edizione 2025, il gruppo della Svizzera ha proposto a classi di quarta primaria il quesito seguente che, con opportuni adattamenti, può ispirare altre attività in linea con quelle presentate in questo capitolo:

Istruzioni per costruzioni (6 punti) +

Avete a disposizione i blocchi da costruzione qui a destra:

6 cubi	1 ponte	2 piramidi	2 pilastri

La sequenza di istruzioni qui sotto permette di realizzare diverse costruzioni: a destra ci sono due esempi.

1. Prendi 3 cubi.
2. Mettili uno sull'altro per formare una torre.
3. Forma una seconda torre con i 3 cubi rimasti.
4. Metti i 2 pilastri accanto alle torri realizzate.
5. Metti il ponte sopra la tua struttura.
6. Metti le due piramidi sopra la tua struttura.

Per ciascuna delle costruzioni qui sotto indicate se è Vero o Falso che possono essere realizzate con le istruzioni date.

V F

V F

V F

V F

- Nel progetto *Programma il Futuro* ci sono alcune proposte didattiche in linea con quanto visto in questo capitolo, ad esempio:
 - <https://q3.hubscuola.it/mxe>
L'attività, attraverso il disegno di creature fantastiche, introduce alcuni metodi e pratiche fondamentali che caratterizzano il pensiero computazionale: la scomposizione in sotto-problemi, l'individuazione di schemi ricorrenti, l'astrazione dai dettagli superflui e la creazione di algoritmi precisi. Lavorando in piccoli gruppi, alunni e alunne devono analizzare un catalogo di mostri per generare una sequenza di istruzioni che permetta di riprodurre correttamente un'immagine specifica.
 - <https://q3.hubscuola.it/tvh>
L'attività, attraverso il gioco del Tangram e la piegatura della carta, permette di lavorare sull'importanza di fornire istruzioni precise e di riflettere su come uno stesso obiettivo possa essere raggiunto seguendo percorsi differenti e con diversi livelli di efficienza.
- <https://q3.hubscuola.it/5xh>
Per ulteriori spunti e attività unplugged nella scuola primaria, è possibile consultare il documento *“Le attività unplugged – Coding con le fiabe”*, che propone esempi semplici e concreti ispirati alla vita quotidiana e alle fiabe, come cucinare, muoversi nello spazio o lavarsi i denti. Queste attività mostrano come riconoscere e descrivere algoritmi nelle azioni di ogni giorno, offrendo un ponte naturale con le esperienze presentate in questo capitolo.

CAPITOLO 4:

MOVIMENTI SU UNA GRIGLIA

Nel capitolo precedente abbiamo lavorato su situazioni quotidiane e giochi, in cui gli alunni e le alunne hanno sperimentato l'idea di descrivere **sequenze** di azioni e di osservare ciò che succedeva eseguendole.

In questo capitolo facciamo un passo avanti: ci spostiamo in un ambiente più strutturato, la griglia, che diventa il nostro campo di gioco.

Da un punto di vista informatico, la griglia è un contesto chiaro in cui formulare sequenze di **istruzioni** precise, osservare come un **esecutore** le interpreta, individuare eventuali errori e correggerli. È il modo più immediato per far capire ai bambini e bambine che un computer (o un robot) esegue sempre e solo ciò che “gli diciamo”, senza intuire ciò che non è stato specificato.

Da un punto di vista delle altre discipline, come la matematica (ma non solo), muoversi su una griglia significa ragionare in termini di spazio, angoli, distanze, direzioni, coordinate. È un terreno ideale per materializzare concetti di geometria e di ragionamento spaziale in attività pratiche.

TRAGUARDI E OBIETTIVI GENERALI

- Pianificare sequenze di istruzioni ordinate per raggiungere un obiettivo definito.
- Prevedere l'esecuzione di un programma, immaginando passo per passo cosa farà l'esecutore.
- Verificare sperimentalmente l'esecuzione di un programma, confrontando le previsioni con il risultato.
- Individuare e correggere errori (debugging) osservando gli effetti dell'esecuzione.
- Comprendere il valore di un linguaggio preciso e condiviso, che consente di comunicare efficacemente sia con un computer sia tra esseri umani.
- Riconoscere e utilizzare diversi sistemi di riferimento spaziali (ad es. destra/sinistra, punti cardinali) per descrivere movimenti e posizioni.

PREREQUISITI

Gli alunni e le alunne avranno già svolto le attività del Capitolo 3 - Algoritmi quotidiani, quindi:

- dovrebbero già avere un'idea iniziale di **algoritmo**, inteso in modo informale come una sequenza di passi per raggiungere un obiettivo;
- dovrebbero avere familiarità con il fatto che le istruzioni vengono eseguite da un esecutore (computer o umano “che fa il computer”), nell'ordine in cui sono state scritte.

CONTENUTI, IN BREVE

Questo capitolo contiene attività a difficoltà crescente in cui i bambini e le bambine devono far completare delle “missioni” a un personaggio che si muove su un campo di gioco composto da una griglia (ad esempio, una griglia 8x8 disegnata sul pavimento, composta da 64 caselle).

Può essere utile inserire l'attività in un contesto che dia senso alla missione e al personaggio. Ad esempio:

- il personaggio è Miguel e la sua missione è prendere un certo peluche usando la macchina con l'artiglio (Claw Crane);

- il personaggio è un'ape che vola tra i fiori e la sua missione è impollinare più fiori possibili;
- personaggio e missione sono ispirati a una storia letta in classe e poi simulata sulla griglia. Ad esempio, il personaggio è Cappuccetto Rosso e la sua missione è andare dalla nonna e portarle il pranzo, evitando il lupo per strada.

Attività 1 – Be my robot: gli alunni e le alunne guidano un compagno-robot sulla griglia con istruzioni semplici in linguaggio naturale, sperimentando diversi modi per descrivere movimenti e posizioni.

Attività 2 – Be my more intelligent robot: si arricchisce il linguaggio con istruzioni di ripetizione e selezione (“se... allora...”), per scrivere programmi più compatti e generali.

Attività 3 – Bee my robot: gli stessi concetti vengono ripresi usando veri robot educativi che si muovono sulla griglia, programmati in anticipo tramite tasti o carte-istruzioni.

MOVIMENTI SU UNA GRIGLIA E SISTEMI DI RIFERIMENTO

Nelle attività discusse in questo capitolo, i bambini e le bambine imparano a scrivere istruzioni per far muovere un compagno/a su una griglia. L'attività si svolge in coppia o in piccolo gruppo: a turno, alcuni progettano e scrivono (o, in alcuni casi, dettano) le istruzioni per il movimento (cioè, interpretano il ruolo dei **programmatori**) mentre altri le devono eseguire (cioè, interpretano il ruolo dell'**esecutore**).

Chi esegue il movimento lo fa in modo meccanico, senza aggiungere né indovinare nulla.



ASPETTO A CUI FARE ATTENZIONE

Distinguere tra chi **pianifica** e chi **esegue** è importante per avvicinare gli alunni e le alunne a un modo di pensare tipico dell'informatica. Il programmatore deve immaginare in anticipo cosa accadrà, scrivere le istruzioni correttamente e prevedere gli effetti di ciascuna istruzione. L'esecutore, invece, si comporta come un piccolo robot *senza volontà*: segue ciò che è scritto, anche quando qualcosa non ha senso, aiutando così il gruppo a capire se il programma è stato progettato bene o va corretto.

Le istruzioni riguardano i movimenti sulla griglia e si possono descrivere usando diversi sistemi di riferimento:

- **oggettivi o assoluti** (es. punti cardinali, riferimento cartesiano);
- **soggettivi egocentrici** (rispetto a chi dà le istruzioni, es. “alla mia destra”);
- **soggettivi allocentrici** (rispetto a chi si muove, es. “alla tua destra”).

Sistemi di riferimento oggettivi o assoluti

In questo caso, il programmatore descrive il movimento come se leggesse una mappa: le direzioni non dipendono da dove è orientato l'esecutore, ma da un sistema di riferimento esterno e fisso.

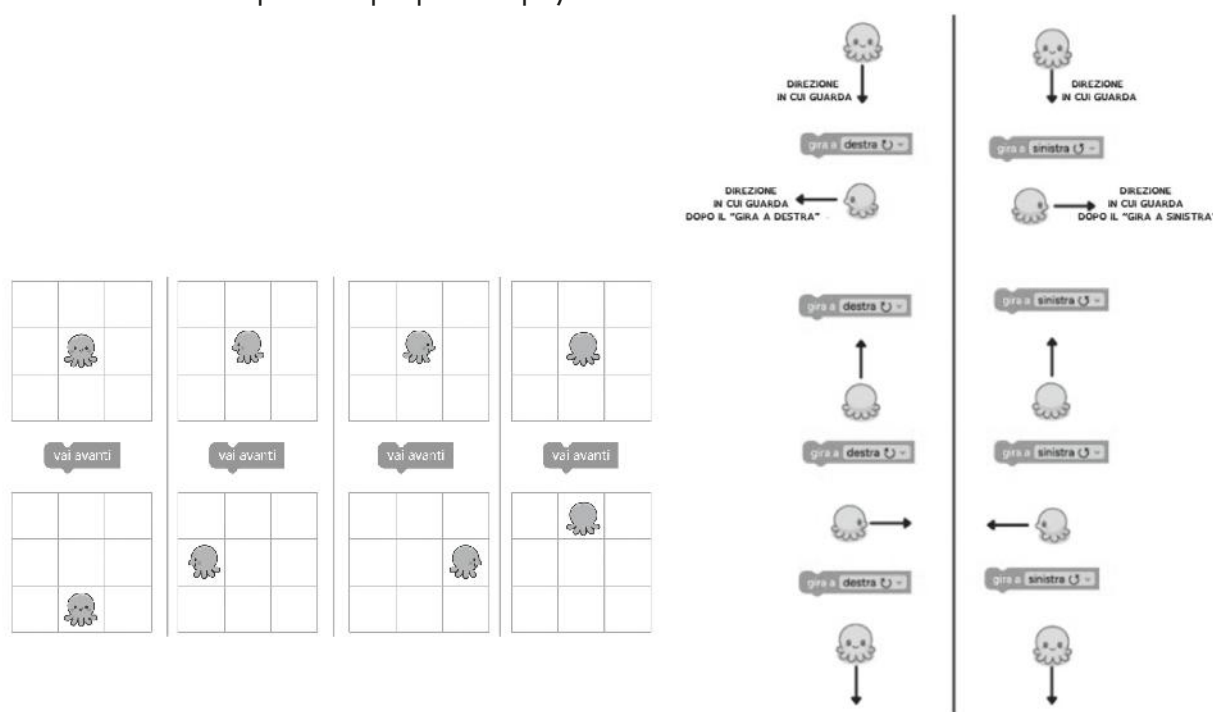
In questi sistemi, ad esempio, l'istruzione “vai a NORD di una casella” avrà sempre lo stesso effetto: il personaggio/robot/persona con il ruolo di esecutore andrà nella casella che si trova immediatamente a Nord (come definito dal sistema di riferimento, es. con una rosa dei venti stampata in un angolo della griglia) rispetto a quella in cui è posizionato, indipendentemente dalla direzione in cui è rivolto prima di eseguire l'istruzione.

Se in classe avete già introdotto i punti cardinali, queste attività possono essere un'occasione per rafforzare il lessico spaziale e per utilizzare i concetti appresi in contesti nuovi. Altrimenti, questa è l'occasione per presentarli!

Incontreremo questi sistemi anche nei Corsi A e B (2021) di Code.org, descritti nel Capitolo 7.

Sistemi di riferimento soggettivi

In questi sistemi, invece, sono fondamentali la direzione in cui l'esecutore è orientato ("dove sta guardando") e le istruzioni date dal programmatore saranno di tipo soggettivo allocentrico perché dovranno essere espresse dal punto di vista dell'esecutore ("vai avanti" potrebbe voler dire andare avanti di una casella nella direzione in cui sta guardando l'esecutore; "gira a sinistra" potrebbe significare che l'esecutore deve girarsi su se stesso sul posto di 90 gradi in senso antiorario rispetto al proprio corpo).



Questo tipo di approccio, in cui la descrizione del movimento viene espressa dal punto di vista del corpo di chi si muove, è stato proposto da Papert con il linguaggio educativo LOGO e viene chiamato "body syntonic", cioè in sintonia con il corpo: se l'esecutore è un robot, il bambino immagina di essere lui stesso il robot. Molte attività unplugged, molti robot (es. BeeBot) e molti linguaggi didattici, come Scratch e quelli di Code.org a partire dal Corso C, adottano questo approccio, su cui è quindi bene iniziare a lavorare già con attività unplugged come quelle descritte.

Oggettivi o soggettivi?

La scelta tra riferimenti oggettivi e soggettivi dipende molto dal momento dell'anno e dal grado di maturità della classe. Un aspetto molto interessante delle attività di movimento su griglia è che lo stesso percorso può essere descritto in modi diversi. Cambia la descrizione linguistica (e, possibilmente, alcuni dettagli di movimento e il numero di istruzioni eseguite), ma non il risultato del programma (per esempio, il personaggio attraversa le stesse caselle, arriva allo stesso punto, raccoglie gli stessi oggetti). Questo offre a bambini e bambine un'occasione preziosa per:

- confrontare diverse rappresentazioni di una soluzione;

- lavorare sulla flessibilità cognitiva;
- prendere consapevolezza del fatto che la descrizione linguistica non è neutra, ma influenza il modo in cui pensiamo e organizziamo le azioni.



DOMANDE STIMOLO

- Possiamo scrivere lo stesso programma usando parole diverse?
- Che differenza c'è tra dire solo la meta ("vai lì") e spiegare tutti i passi per arrivarci?
- Quale descrizione linguistica preferite?

Far notare agli alunni e alle alunne che stanno programmando l'esecutore per percorrere lo stesso percorso, ma in due modi diversi, è utile per lavorare su un primo livello di astrazione: stanno manipolando rappresentazioni dei movimenti da eseguire, e non solo camminando su una griglia.

MODALITÀ DI LAVORO

Tutte le attività che seguono sono pensate per essere svolte in classe.

- Il lavoro si svolge in piccoli gruppi o collettivamente, attraverso il movimento fisico, il disegno, la costruzione di percorsi o l'uso di robot educativi.
- Lo spazio deve essere abbastanza ampio da permettere di muoversi liberamente (es. corridoio o aula sgombra).
- È cruciale permettere agli alunni e alle alunne di sviluppare soluzioni alternative: l'insegnante comunica la consegna, osserva lo svolgimento e stimola riflessioni con domande guida.
- Per poter agganciare le domande guida a ciò che avviene in classe, è importante tenere traccia delle parole e delle produzioni degli alunni e delle alunne (ad esempio, fornendo schede strutturate per scrivere, ma anche scattando foto o video).

Come detto, in tutte le attività descritte è importante separare nettamente la parte di "scrittura del programma", cioè la fase in cui uno o più alunni o alunne ipotizzano (e scrivono) l'intera sequenza di passi che l'esecutore deve (e)seguire per svolgere il compito assegnato, da quella in cui un bambino o una bambina che impersona l'esecutore esegue tali istruzioni in modo acritico. La fase di scrittura dovrebbe essere separata dall'esecuzione: dopo una fase introduttiva in cui si provano le diverse istruzioni "per vedere l'effetto che fanno", dovranno provare a scrivere il programma "immaginando" che cosa farà l'esecutore, senza poterlo vedere esplicitamente.

Spesso, infatti, c'è la tendenza di chi ha scritto la sequenza di istruzioni, o di chi la sta leggendo o eseguendo, ad accorgersi dell'errore e a volerlo correggere immediatamente. Per mantenere coerente la metafora informatica alla base di questa attività (il programmatore scrive un programma, lo esegue, osserva gli errori che avvengono/l'output errato, modifica il programma, e poi, solo in seguito alle modifiche, lo ri-esegue, e così via) è importante quindi che, se qualcuno (idealmente chi ha scritto il programma) si accorge di qualche errore, possa "fermare l'esecuzione" (o attendere la fine), e solo a questo punto modificare il programma e chiedere che venga ri-eseguito.

Come detto, chi interpreta l'esecutore, dal canto suo, deve invece limitarsi a eseguire le istruzioni così come vengono fornite / come le legge, anche se dovesse rendersi conto che non sono corrette per l'obiettivo.

Quanto più possibile, il feedback non dovrà essere fornito dall'insegnante: bambini e bambine potranno comprendere se hanno scritto un programma corretto verificando che, quando eseguito, raggiunga l'obiettivo desiderato.

ATTIVITÀ 1: Be my robot

→ Vedi Quaderno di Informatica pp. 10-11

Preparazione

Stampare i materiali necessari. Serviranno sicuramente:

- supporti (es. carte, fogli, oggetti) che rappresentano o contengono le istruzioni (es. un foglio con una freccia e la relativa istruzione testuale, come **VAI AVANTI**);
- supporti vuoti per far costruire agli alunni e alle alunne istruzioni personalizzate.

Se si usa un sistema di riferimento oggettivo:

- frecce Nord / Sud / Ovest / Est;
- rosa dei venti.

Se si usa un sistema di riferimento soggettivo:

- frecce **VAI AVANTI DI UN PASSO / GIRA A DESTRA DI UN ANGOLO RETTO / GIRA A SINISTRA DI UN ANGOLO RETTO** disegnate dentro ai supporti.

Costruire una griglia 8x8 (es. utilizzando le piastrelle, o il nastro adesivo di carta sul pavimento, o i tappetini puzzle a quadrati) e posizionare eventuali ostacoli, oggetti da raccogliere, e così via.

Fase 0 - attività propedeutica**Obiettivo**

Scoprire quali istruzioni l'esecutore è in grado di comprendere¹.

Descrizione

Nella griglia si prepara una missione da compiere, ad esempio: la griglia rappresenta un orto e il personaggio è un robot raccogli-verdure e la sua missione è raccogliere tutte le verdure e portarle in una cassetta.



L'insegnante interpreterà il personaggio e dovrà eseguire le istruzioni di bambini e bambine. Non le dovrà eseguire tutte, ma solo quelle precise e comprensibili, rifiutando quelle troppo generiche o ambigue (dicendo esplicitamente "non capisco").

Per esempio, l'istruzione **CERCA E RACCOGLI TUTTE LE CAROTE E RIPORTALE NELLA CASSETTA** sarà scartata perché troppo generale, mentre **VAI INDIETRO DI UNA CASELLA** potrà essere accettata e messa in atto.

Ai bambini e alle bambine viene consegnata la SCHEDA STUDENTI FASE 0 (si veda l'esempio a pagina seguente). Essi, osservando attentamente e attivamente il comportamento dell'insegnante-esecutore, dovranno prima individuare le istruzioni ammesse, costruendo insieme l'insieme limitato di istruzioni che l'esecutore è in grado di comprendere ed eseguire.

Analizzando la tabella compilata, dovranno cercare di capire quali sono le caratteristiche delle istruzioni che il robot riesce a comprendere. Con le schede compilate si discute in classe.

**DOMANDE STIMOLO**

- Cosa avete trovato di simile tra le istruzioni che il robot capiva?
- E cosa avete trovato di simile tra quelle che non capiva?

Al termine dell'attività, alla lavagna si scrivono i nomi delle istruzioni ammesse che devono codificare i movimenti sulla griglia (es. vai avanti di un passo, gira a destra di un angolo retto,

¹ Questo approccio è ispirato alle attività proposte da Lodi e colleghi [2020].

gira a sinistra di un angolo retto) e le operazioni necessarie a raggiungere gli obiettivi (es. raccogli la frutta e metti la frutta nella cassetta).



ASPETTO A CUI FARE ATTENZIONE

Perché tutto funzioni, le istruzioni devono essere:

- scritte sempre nello stesso modo, seguendo una forma precisa: questo riguarda la **sintassi**, ovvero le “regole del linguaggio” (per esempio: si decide che si scrive sempre **AVANTI 2** e non “vai avanti di due caselle”);
- chiare nel significato, così che tutti comprendano esattamente cosa deve succedere: questo riguarda la **semantica**, ovvero il significato di ciascuna istruzione (per esempio: **GIRA A SINISTRA** vuol dire ruotare di 90 gradi in senso antiorario sul posto);

Inoltre, è importante tenere a mente i casi limite: cosa succede se il robot è su una casella sul bordo della griglia e gli viene chiesto di fare un passo in una direzione che lo porterebbe fuori dalla griglia?

A seconda del contesto scelto (una storia, una missione, un gioco), si possono introdurre anche altre istruzioni: raccogli un oggetto, salta, lascia un oggetto, fai un suono, e così via. Questo rende l'attività più coinvolgente e stimola l'immaginazione, mantenendo l'attenzione sulla precisione del linguaggio.

Contenuto informatico

Questa attività introduce in modo naturale il concetto di linguaggio formale e prepara il terreno per le fasi successive.

SCHEDA STUDENTI FASE 0

L'orto è pieno di verdure!

Dai a voce le istruzioni al robot-raccogli verdura per raccogliere tutta la verdura e metterla nella cassetta. Attenzione però! Il robot capisce solo alcune istruzioni: quali? Cerca di scoprirlo!

Primo passo: dai diverse istruzioni al robot e compila la tabella.

ISTRUZIONI CAPITE	ISTRUZIONI NON CAPITE

Secondo passo: osserva la tabella. Cosa hanno di speciale le istruzioni che il robot capisce? Scrivi tutte le tue osservazioni.

Fase 1 - Prendere familiarità con il linguaggio

Obiettivo

Riconoscere l'associazione tra istruzioni e comportamenti dell'esecutore in un contesto in cui le istruzioni sono espresse mediante un sistema di riferimento soggettivo rispetto al punto di vista dell'esecutore (legame tra sintassi e semantica).

Descrizione

Le istruzioni ammesse sono quelle scritte sulla lavagna al termine della Fase 0.

Nella griglia si prepara una missione da far compiere al personaggio-esecutore, ad esempio il robot raccogli-verdura.

Il personaggio ha sempre una direzione iniziale e interpreta le istruzioni in base a essa.

Stavolta a interpretare il personaggio saranno, a turno, gli alunni e le alunne. L'esecutore si posiziona in un punto della griglia. Gli altri si mettono tutti da un lato della griglia (devono avere lo stesso punto di vista) e danno, una alla volta, le istruzioni al compagno-esecutore per compiere la missione.

Sarà importante far sperimentare sia il ruolo di chi dà le istruzioni sia quello di chi le esegue.

Questa fase si può ripetere facendo interpretare l'esecutore (nel nostro esempio, il robot raccogli-verdure) anche a un oggetto (di cui sia chiaro l'orientamento nello spazio: per alcuni oggetti, per esempio una palla, è difficile capire cosa significa "andare avanti di un passo", non essendo chiara la direzione in cui è orientata).

Dopo aver fatto qualche prova, si discute in classe, cercando di riportare alla lavagna tutte le osservazioni dei bambini e delle bambine.



DOMANDE STIMOLO

- Che cosa vuol dire "gira a destra di un angolo retto"?
- Che cosa vuol dire "gira a sinistra di un angolo retto"?
- Che cosa vuol dire "vai avanti di un passo"?
- Se parto guardando in una certa direzione, quanti "gira a destra di un angolo retto" consecutivi devo fare per tornare a guardare nella direzione iniziale?
- E quanti "gira a sinistra di un angolo retto" consecutivi?
- Se dalla posizione iniziale il personaggio esegue le istruzioni "gira a sinistra di un angolo retto" - "vai avanti di un passo" - "vai avanti di un passo" dove arriva?
- E ora che istruzioni date per farlo tornare alla posizione iniziale?

Al termine della discussione, alla lavagna in corrispondenza di ciascuna istruzione, si aggiunge la descrizione a parole del corrispondente comportamento del personaggio esecutore.

Per chiudere, gli studenti scrivono le istruzioni e i significati nella MANUALE DELLE ISTRUZIONI.

SCHEMA 2A - CAPITOLO 4

CLASSE _____ NOME _____ DATA _____

MANUALE DELLE ISTRUZIONI

Completa la tabella.

Istruzione	Cosa fa il personaggio? Si sposta di una casella nella direzione in cui "sta guardando" (è orientato)	Disegna sulla griglia
vai avanti di un passo		• prima
		• dopo
gira a destra di un angolo retto		• prima
		• dopo

VEDI PAGINA 116

SCHEMA 2B - CAPITOLO 4

CLASSE _____ NOME _____ DATA _____

Istruzione	Cosa fa il personaggio?	Disegna sulla griglia
gira a sinistra di un angolo retto		• prima
		• dopo
		• prima
		• dopo

VEDI PAGINA 117

Fase 2 - Programmare il movimento sull'intero percorso

Obiettivi

- Scrivere un programma pianificando il comportamento dell'esecutore e prevedere il suo comportamento senza eseguire il programma.
- Verificare la propria previsione osservando l'esecuzione del programma scritto.

Descrizione

Una volta acquisita familiarità con le istruzioni e il loro significato, si propone un'attività di scrittura di un programma ottenuto mettendo in sequenza diverse istruzioni.

Nella griglia, si sceglie un personaggio e si stabilisce per il personaggio una missione uguale per tutti che prevede un punto di partenza e un obiettivo (es. partendo dalla cella di partenza, raggiungere un'altra cella con un oggetto).

Si distribuisce la PROGRAMMA UNA MISSIONE!: studenti e studentesse saranno i programmatori e devono immaginare in anticipo cosa accadrà, progettare una strategia per raggiungere l'obiettivo e scrivere il programma per realizzarla.

SCHEDA 118 - CAPITOLO 6

CLASSE _____ NOME _____ DATA _____

PROGRAMMA UNA MISSIONE!

Personaggio: _____

Missione: _____

Disegna nella griglia la situazione iniziale, l'obiettivo e, con la matita, la strada che vuoi far fare al personaggio. Poi completa la tabella.

ISTRUZIONI DISPONIBILI	PROGRAMMA PER COMPIERE LA MISSIONE

118

VEDI PAGINA 118

Quando ciascuno ha compilato la propria scheda, si avvia un confronto collettivo tra le diverse strategie e i diversi programmi scritti: l'insegnante si comporta da esecutore e esegue alcuni dei programmi scritti dai bambini e dalle bambine. Sarà importante scegliere programmi che contengano errori, per stimolare una riflessione su come correggerli.

Mentre l'insegnante esegue il programma di qualcuno, muovendosi sulla griglia del pavimento, chi l'ha scritto traccia sulla propria scheda il percorso seguito dall'insegnante con una penna di un altro colore. Al termine dell'esecuzione, si avvia una discussione.



DOMANDE STIMOLO

- Il percorso fatto era quello che ti aspettavi?
- Come mai, secondo te?
- Guardando quello che è successo, come correggeresti il programma?
- Qualcun altro aveva fatto così?

**SUL QUADERNO DI INFORMATICA**

Puoi far lavorare bambini e bambine, in classe o a casa, sulla scheda LABIRINTI (pag. 8 e 9 del Quaderno di Informatica).

La soluzione al primo esercizio, tra le tre proposte, è la prima:



La soluzione alla sequenza da completare (sono già fornite le prime 5 istruzioni) è

**Fase 3 - Programmare evitando gli ostacoli**

Per aumentare la complessità, si disegnano ostacoli sulla griglia o si aggiungono regole (es. “non puoi passare sulle caselle rosse”). I bambini e le bambine devono trovare un percorso valido, usando solo le istruzioni disponibili.

**DOMANDE STIMOLO**

- Esistono più soluzioni?
- Se aggiungo un ostacolo, come cambia il programma?

**SUL QUADERNO DI INFORMATICA**

Puoi far lavorare i tuoi alunni e alunne, in classe o a casa, sulla scheda ROBOT DELLE PULIZIE (pag. 10 e 11 del Quaderno di Informatica).

L'attività è analoga a quanto visto in classe. Integra la presenza di ostacoli da evitare e la scelta di due istruzioni diverse in base a ciò che è presente nella casella in cui si trova il robot.

Esistono molte soluzioni possibili: in prima battuta, tutte le soluzioni che consentono di pulire la griglia (cioè passare su tutte le macchie e la polvere e rimuoverle con il relativo strumento adeguato) vanno bene.

In una seconda fase, è possibile invitare gli alunni e le alunne a provare a utilizzare il minor numero di istruzioni possibile.

Una soluzione corretta ma che usa 14 istruzioni è la seguente



La soluzione ottima, che richiede 12 istruzioni, è la seguente



Fase 4 - Programmare a coppie**Obiettivo**

- Progettare una situazione problematica (missione) e scrivere un programma che la risolva.
- Verificare la propria previsione osservando l'esecuzione del programma scritto.
- Individuare gli errori nel programma scritto e correggerli modificando la sequenza di istruzioni.

Descrizione

In questa fase si lavora in coppia. Ciascuno sceglie un personaggio e una missione, e disegna nella griglia la situazione iniziale. Un componente della coppia fa il programmatore e scrive una sequenza di istruzioni su carta, poi la consegna all'altro componente, che agisce come un esecutore e muove una pedina (o disegna il percorso) su una griglia cartacea.

Suggerimento: dopo un po' si possono invertire i ruoli, in modo che tutti sperimentino entrambe le prospettive.

**DOMANDE STIMOLO**

- Quale ruolo hai trovato più facile? Quello del programmatore o dell'esecutore?
- Come mai secondo te?
- Quali errori avete fatto? E come li avete corretti?

**SUL QUADERNO DI INFORMATICA**

Puoi far lavorare in classe o a casa, sulla scheda ALICE RADICE (pag. 12 e 13 del Quaderno di Informatica).

Qui bambini e bambine completano un programma, ne scrivono uno da zero (cercando anche di farlo con il minimo numero di istruzioni possibile) e poi progettano nuove situazioni problematiche da far risolvere ai propri compagni.

Soluzione Esercizio 1:



Una possibile soluzione Esercizio 2 (quella con meno istruzioni):



ATTIVITÀ 2: Be my more intelligent robot

→ Vedi Quaderno di Informatica pp. 10-11

Oltre alle classiche istruzioni del robot del tipo:

VAI AVANTI	Va avanti di una casella nella direzione in cui è orientato
GIRA A SINISTRA	Gira a sinistra ruotando su se stesso di 90 gradi
GIRA A DESTRA	Gira a destra ruotando su se stesso di 90 gradi
RACCOGLI X	Se nella casella in cui si trova c'è un X, ne raccoglie un'unità. Altrimenti non fa nulla / si blocca e dà errore / ...

è possibile introdurre **strutture di controllo** come la **ripetizione** e la **selezione**.

Ripetizione un numero fissato di volte

Probabilmente la più naturale, e quella che emerge spesso dai bambini stessi, è la ripetizione di una o più istruzioni per un numero fissato di volte.

È importante fornire agli alunni e alle alunne un contesto in cui usare la ripetizione è necessario o desiderabile, per esempio:

- si hanno a disposizione un numero limitato di istruzioni **VAI AVANTI** e quindi è necessario usare, ad esempio, **RIPETI 4 VOLTE: VAI AVANTI**.
- le istruzioni **VAI AVANTI** sono molto costose e dunque si fa una gara a chi riesce a risolvere un problema “spendendo meno”.

Qualsiasi sintassi si scelga, è importante che non sia ambigua: deve essere chiaro quali istruzioni vanno ripetute e quali, invece, vengono dopo la ripetizione.

È possibile utilizzare le parentesi oppure l'indentazione, cioè **SCRIVERE SPOSTATE A DESTRA** tutte le istruzioni da ripetere.

**DOMANDE STIMOLO**

- Che cosa farà il robot con queste istruzioni?

VAI AVANTI

RIPETI 4 VOLTE:

VAI AVANTI

GIRA A DESTRA

- I due programmi seguenti fanno eseguire al robot le stesse istruzioni?

GIRA A DESTRA

RIPETI 3 VOLTE:

VAI AVANTI

GIRA A DESTRA

VAI AVANTI

GIRA A SINISTRA

GIRA A DESTRA

RIPETI 3 VOLTE:

VAI AVANTI

GIRA A DESTRA

VAI AVANTI

GIRA A SINISTRA

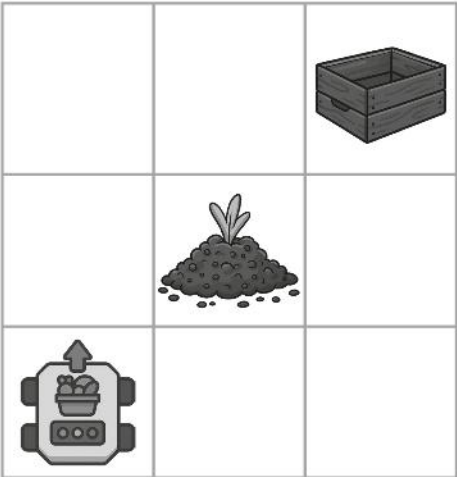
Selezione

Finora i programmi scritti da bambini e bambine risolvono problemi specifici e non sono soluzioni a problemi generali: permettono al personaggio di raggiungere il traguardo solo se il programma viene eseguito nella precisa condizione (cioè, nello stato preciso) mostrata sulla griglia (posizione e orientamento iniziali del protagonista, posizione degli ostacoli o degli oggetti, posizione del traguardo ecc.).

Aggiungere istruzioni che permettano di decidere cosa fare in base a una condizione consente di scrivere programmi più generali.

Per esempio, supponiamo di sapere che in una certa casella c'è un ortaggio o una talpa, ma non sappiamo quale (ad esempio, può essere nascosta sotto un'immagine di un mucchio di terra).

Il robot raccogli ortaggi potrebbe eseguire un programma del tipo:

VAI AVANTI GIRA A DESTRA VAI AVANTI SE C'È UNA CAROTA: RACCOGLI LA CAROTA ALTRIMENTI: CHIEDI SCUSA ALLA TALPA VAI AVANTI GIRA A SINISTRA VAI AVANTI DEPOSITA LA VERDURA	
---	---

Si può creare una griglia con tanti mucchietti e far decidere a un gruppo di bambini se mettere sotto ciascun mucchietto una talpa o una carota.

Un altro gruppo dovrà scrivere il programma, sapendo dove sono i mucchi di terra ma non sapendo che cosa i compagni hanno messo sotto.

Il programma dovrebbe funzionare per tutte le possibili scelte dei bambini che hanno posizionato le carote e le talpe, e potrà essere testato più volte in scenari diversi.

ATTIVITÀ 3: Bee my robot

→ Vedi Quaderno di Informatica pp. 10-11

Sebbene non sia necessario, se hai in dotazione robot educativi, dopo aver svolto le attività delle prime due schede (o almeno della prima), puoi utilizzarli al posto del “compagno-robot”. Il robot, essendo un vero computer, incarna perfettamente l'esecutore “stupido e veloce”: esegue sempre e solo il programma che i bambini hanno inserito, senza intuire, stancarsi o “aiutarli”.

Preparazione

- Spesso i robot vengono venduti con tappeti griglia che forniscono contesti narrativi. In alternativa, è possibile costruire le proprie griglie: l'importante è assicurarsi che le caselle abbiano una dimensione tale da far sì che ogni passo mandi il robot dal centro della casella in cui si trova al centro della casella successiva nella direzione richiesta (e non, ad esempio, a metà tra due caselle). Dunque, le caselle non devono essere né troppo piccole né troppo grandi.
- Prepara o riusa le carte-istruzione coerenti con le istruzioni del robot, in continuità con il linguaggio usato nelle prime due schede. Oppure predisponi un foglio su cui gli alunni e le alunne possano scrivere le istruzioni da dare al robot.

Quale robot scegliere?

Qualsiasi robot che si muove su una griglia con istruzioni avanti/gira è compatibile con le attività del capitolo. Ne esistono moltissimi; ne citiamo alcuni tra i più famosi.

- Bee-Bot (<https://q3.hubscuola.it/j7o>). Robot a forma di ape, con istruzioni direttamente sul dorso: avanti, indietro, gira a destra/sinistra, pausa. È estremamente semplice e robusto; subito comprensibile; con grande disponibilità di tappeti tematici e materiali già pronti. Fai attenzione: il programma rimane “nascosto” (i bambini non vedono la sequenza se non la scrivono a parte); si lavora praticamente solo con sequenze, senza strutture più ricche.
- Blue-Bot (<https://q3.hubscuola.it/nk6>). È una versione “trasparente” di Bee-bot, il che suscita curiosità nei bambini. Si può ugualmente comandare con i tasti sul dorso, ma anche con dei tasselli (simili alle carte) separati, oppure con una app per tablet/pc.
- Cubetto (<https://q3.hubscuola.it/map>). È un robot in legno che si programma inserendo blocchi fisici (avanti, gira a destra/sinistra, funzione...) su un quadro di legno, in stile Montessori. Rende molto evidente che il programma è qualcosa di distinto dall'esecuzione (prima costruisco la sequenza sulla console, poi premo GO); il blocco “funzione” permette di introdurre, in modo molto informale, l'idea di sottoprogramma.
- Tale-Bot Pro (<https://q3.hubscuola.it/0tz>). Robot programmabile tramite tasti o carte-istruzione, pensato per l'infanzia e l'inizio della primaria. Il funzionamento è simile a quello della Bee-Bot, ma dispone di una striscia di LED che visualizza il programma inserito fino a quel punto. Sono disponibili diversi accessori (tappeti, carte di storia, possibilità di farlo disegnare o di montarci sopra dei LEGO® Duplo®). Inoltre, dispone di ulteriori istruzioni (es. registrazione e riproduzione vocale) adatte alle attività di storytelling.

Attività:

- **Esplorare le istruzioni.** Fai provare ai bambini i tasti del robot uno alla volta. Per ogni istruzione, chiedi: “Cosa succede?” e collega il movimento alla griglia (quante caselle? che rotazione?)
- **Scrivere il programma prima di eseguire.** Come nelle attività unplugged, chiedi agli alunni e alle alunne di progettare la sequenza con le carte-istruzione (o su un foglio) e solo dopo fagliela inserire nel robot.

- **Debugging.** Se il robot non raggiunge l'obiettivo, è il programma ad essere sbagliato: fermate l'esecuzione, rileggete la sequenza, correggete ed eseguite nuovamente. Il fatto che il robot non sia in grado di comprendere l'errore aiuta a capire che non sta "ragionando", sta solo eseguendo la sequenza di istruzioni che gli è stata fornita.

Contenuto informatico

Rispetto al compagno-robot, il robot fisico:

- non interpreta le istruzioni,
- ripete il programma identico quante volte si vuole.

Questo rende visibile l'idea di una macchina veloce e instancabile, ma "stupida", cioè "meccanica".

POSSIBILI ESTENSIONI, VARIAZIONI E PERSONALIZZAZIONI

Setting

Se non si può andare in giardino / usare le mattonelle dell'aula, è possibile usare un foglio di carta e oggettino (es. pedine di un gioco e simile) come personaggio.

Spunti per adattare l'attività

In caso di difficoltà:

- Far indossare all'esecutore due laccetti di colori diversi, uno al polso sinistro e uno al polso destro.
- Utilizzare codici colori per le diverse istruzioni (con colori adeguati per studenti con daltonismo).

Come rendere l'attività più sfidante:

- Interpretare un programma senza eseguirlo: il bambino-esecutore deve dire dove andrà il robot senza eseguire i passi.
- Carte "imprevisto": tieni un mazzo di carte (o dei post-it) con eventi casuali (es. «Piastrella rotta: scegli una strada alternativa»). L'osservatore pesca una carta e la inserisce, costringendo il gruppo a modificare il programma.
- Caccia al tesoro: raggiungere una certa casella fa raggiungere un indizio che è un altro programma per nuovi movimenti.

ASPETTI INFORMATICI

Un aspetto importante affrontato in questo capitolo riguarda il "set di istruzioni disponibili": l'esecutore – che sia un compagno, un robot o un oggetto simbolico – può seguire **solo** le istruzioni previste. Questa limitazione rende evidente quanto sia cruciale avere un linguaggio condiviso e quanto la scelta delle istruzioni (da parte di chi progetta il linguaggio) influenzi la facilità o la difficoltà nel risolvere un problema.

Nelle attività in classe può capitare che bambini e bambine sentano il bisogno di aggiungere nuove istruzioni perché quelle già disponibili non sono sufficienti o risultano troppo scomode. Questo è un passaggio molto interessante, perché li porta naturalmente all'idea di definire proprie istruzioni, cioè di creare "scorciatoie" – istruzioni più complesse a partire da quelle elementari. È lo stesso processo che avviene nella programmazione vera e propria, quando costruiamo nuove funzioni o procedure per semplificare il lavoro: invece di ripetere sempre gli stessi passi, possiamo raggrupparli e dare loro un nome. In questo modo bambini e bambine non solo imparano a seguire le regole, ma anche a riflettere criticamente sul linguaggio che usano, a modificarlo e a renderlo più adatto alle proprie esigenze.

Dopo aver compreso la logica delle sequenze, i movimenti su griglia diventano il terreno ideale per introdurre due altre strutture fondamentali della programmazione: le **selezioni** e le **ripetizioni**. Le selezioni permettono di prendere decisioni (“se trovi un ostacolo, gira a destra”), mentre le ripetizioni consentono di semplificare un compito che altrimenti richiederebbe molte istruzioni (“ripeti tre volte: vai avanti di una casella”).

Queste attività aiutano bambini e bambine a scoprire che un programma non è solo una sequenza lineare di istruzioni, ma può contenere regole che guidano il comportamento dell'esecutore in situazioni variabili. In questo modo il programma diventa più flessibile e generale: invece di descrivere ogni singolo passo, si impara a progettare programmi che funzionano in più contesti.

Lavorando su selezioni e ripetizioni, alunni e alunne lavorano anche sulla capacità di pianificazione: devono immaginare non solo il percorso corretto, ma anche le possibili varianti o ostacoli e predisporre regole che guidino l'esecuzione senza possibilità di fraintendimenti. In questo contesto, l'errore non è un evento da evitare, anzi: il riconoscere gli errori e correggerli (*debugging*) è una pratica fondamentale in ambito informatico ed è parte integrante dell'attività.

Nella programmazione, il linguaggio diventa uno strumento per controllare un'azione: non basta avere un'intuizione o sapere cosa fare, bisogna tradurre l'intenzione in una sequenza di istruzioni chiara, non ambigua e comprensibile anche da un altro (una persona, un robot, un personaggio).

Due elementi tipici della disciplina informatica entrano così in gioco:

- la **pianificazione**, ovvero la capacità di progettare in anticipo una sequenza di istruzioni prima che venga eseguita, ragionando sugli effetti che ogni azione avrà;
- la **correzione degli errori** (*debugging*), cioè la disponibilità ad osservare ciò che non ha funzionato, risalire alla causa e modificare le istruzioni in modo efficace.

Allenarsi a pensare in questo modo, anche attraverso attività giocose, permette di lavorare sullo sviluppo di un approccio ordinato, riflessivo e orientato alla soluzione dei problemi – che è alla base del pensiero computazionale e quindi dell'informatica.

RELAZIONE CON L'INFORMATICA NELLE INDICAZIONI NAZIONALI

Relativamente alle Indicazioni Nazionali 2025, queste attività contribuiscono allo sviluppo delle seguenti competenze.

Competenze (alla fine della scuola primaria) nella disciplina Matematica:

- *Descrivere procedure per lo svolgimento di compiti pratici mediante algoritmi.* Questa competenza era già coperta dalle attività della sezione “Algoritmi quotidiani”. Nelle attività proposte in questa sezione i compiti non sono più quelli della vita quotidiana, ma sono gli obiettivi dettati dal contesto-storia in cui l'insegnante colloca le attività sulla griglia.
- *Scrivere e comprendere semplici programmi, espressi in elementari linguaggi di programmazione a scopo didattico, e valutarne l'adeguatezza rispetto al compito che si vuole automatizzare.*

Sebbene queste attività siano unplugged (senza computer), i linguaggi utilizzati sono comunque **linguaggi formali** con una sintassi e una semantica ben definite e non ambigue. Per questo sono una buona preparazione a questa competenza, poiché sono orientate alla progettazione e scrittura di programmi, e alla loro valutazione (controllo di correttezza, debug, ...)

Tenendo presenti le stesse specifiche appena fatte per le competenze, le attività proposte contribuiscono ai seguenti obiettivi.

Obiettivi (alla fine della terza primaria) nella disciplina Matematica

- *Descrivere a parole attività della vita quotidiana tramite sequenze di passi precisi e non ambigui e saperle eseguire.*
- *Scrivere semplici programmi e verificare, mediante la loro esecuzione, se svolgono il compito previsto ed eventualmente correggerli.*

Infine, lavorano sulle seguenti conoscenze (inserite nelle Indicazioni Nazionali 2025 a titolo di esempio, non prescrittive):

- concetto di algoritmo e sua esecuzione rigorosa;
- strutture di controllo fondamentali di un linguaggio di programmazione. Qui si lavora in particolare sulla sequenza, la più semplice di queste strutture, sulla selezione e sulla ripetizione;
- controllo di correttezza dei programmi.

VALORE COSTRUZIONISTA

Le attività sulla griglia non sono semplici giochi: diventano esperienze in cui gli alunni e le alunne materializzano le proprie idee, trasformandole in movimenti sulla griglia visibili da tutti. La condivisione permette il confronto e la discussione con gli altri: ciascuno può osservare il risultato delle istruzioni fornite, confrontarlo con le proprie intenzioni e rielaborarlo insieme al resto della classe.

In questo processo, l'errore è un'occasione preziosa per inventare varianti, immaginare soluzioni alternative e approfondire i propri ragionamenti. Le attività descritte non sono utili ai bambini solo per apprendere a pianificare un percorso, ma anche per riflettere su come hanno pensato e comunicato, e per iniziare a sviluppare autonomia, metacognizione e creatività.

SUGGERIMENTI PER ULTERIORI ATTIVITÀ E RIFERIMENTI

I movimenti su una griglia sono tra le più diffuse tra le proposte di informatica (spesso etichettate con il termine “coding”), e quindi è possibile trovarne in moltissimi siti web, libri, blog. Il capitolo che hai appena letto dovrebbe aiutarti a valutare criticamente, da un punto di vista informatico, le proposte che potresti incontrare.

- Vista la vasta offerta, abbiamo deciso di segnalare le proposte del progetto forse più antico e sicuramente più influente sul tema: **CS Unplugged**.

Si tratta di un libro inizialmente scritto da Tim Bell, Ian H. Witten and Mike Fellows, e ora divenuto un insieme di risorse (in inglese, con licenza CC BY-SA) curate dall'Università di Canterbury, Nuova Zelanda (<https://q3.hubscuola.it/ht4>).

Le attività di movimento sulla griglia vengono chiamate “Kidbots” (“Bambini robot”) - in particolare, segnaliamo le attività:

- **Rescue mission:** <https://q3.hubscuola.it/weo>
- **Sending a rocket to Mars:** <https://q3.hubscuola.it/tie>

- Nella versione “classica” di CS Unplugged, di cui è disponibile (sempre con licenza CC BY-NC) una traduzione a cura di Renzo Davoli, Giovanni Michele Bianco, Piergiovanna Grossi, sono presenti proposte in linea con le nostre nell’attività “13 - Gli ordini di marcia” (pp. 153-157), in cui gli studenti devono riprodurre un disegno basandosi esclusivamente sulle indicazioni di un compagno. Nelle variazioni, si propone, ad esempio, di dare indicazioni a uno studente per raggiungere un punto segreto della scuola. Di seguito il link alla traduzione: <https://q3.hubscuola.it/6zb>
- Un percorso didattico in italiano (con licenza CC BY-SA) è quello descritto in: Michael Lodi, Renzo Davoli, Rebecca Montanari, Simone Martini [2020]. Informatica senza e con computer nella Scuola Primaria. Capitolo del volume “Coding e oltre: Informatica nella scuola”, a cura di Enrico Nardelli, edito da Lisciani. Una versione libera del capitolo (con tutti i riferimenti ai materiali didattici liberamente scaricabili) è disponibile a <https://q3.hubscuola.it/8dr>

CAPITOLO 5: RAPPRESENTAZIONE DELL'INFORMAZIONE

L'essere umano ha sempre cercato modi per codificare le informazioni, cioè per tradurle in simboli che possano essere conservati, condivisi e compresi da altri. Molti di questi sistemi fanno parte della nostra vita quotidiana e li usiamo senza nemmeno pensarci:

- l'alfabeto per rappresentare i suoni delle parole attraverso lettere scritte;
- le note musicali per rappresentare i suoni di una melodia su un pentagramma;
- le cifre per rappresentare quantità e numeri;
- i cartelli stradali per rappresentare regole di comportamento;
- più recentemente, gli emoji per rappresentare emozioni.

Questi sono tutti esempi di rappresentazioni: modi convenzionali per trasformare qualcosa di complesso (un suono, un'immagine, un luogo, un'azione) in un insieme di simboli che possiamo leggere, interpretare e ricostruire.

In informatica, il principio è lo stesso: vengono prima scelti dei dati da usare per parlare di un'informazione (un testo, un disegno, un suono, un video); poi, i dati scelti vengono codificati con simboli seguendo regole precise. Si può anche percorrere il processo contrario, cioè quello della decodifica, che dalla rappresentazione dei dati permette di ricostruire l'informazione. Il processo di codifica e decodifica permette di memorizzare, trasmettere e ricostruire in modo affidabile l'informazione per un esecutore automatico; per le persone, l'interpretazione dipende comunque dal contesto e dalle conoscenze pregresse.

A seconda della rappresentazione scelta, i dati potranno essere più o meno facili da elaborare da una macchina, occupare più o meno spazio, essere più o meno veloci da leggere o da scrivere.

Le attività di questa sezione aiuteranno alunni e alunne a sperimentare in modo concreto queste idee: esploreranno diversi modi per descrivere la stessa cosa e si confronteranno con i processi di codifica e decodifica.

TRAGUARDI E OBIETTIVI GENERALI

- Descrivere a parole il funzionamento di semplici sistemi di codifica e decodifica.
- Applicare correttamente le regole di un sistema di codifica dato per: decodificare immagini in *pixel art*; ricostruire messaggi a partire da sequenze di simboli.
- Codificare immagini o messaggi in una rappresentazione simbolica.
- Confrontare e valutare diversi sistemi di codifica in termini di univocità e costo.
- Confrontare diverse rappresentazioni della stessa informazione.
- Individuare e correggere ambiguità o errori nel processo di codifica e/o decodifica (debugging).

PREREQUISITI

Le attività proposte non richiedono particolari conoscenze informatiche pregresse. È sufficiente che bambini e bambine:

- sappiano seguire consegne orali e scritte e riconoscano alcune parole o frasi brevi (per le attività sui messaggi di Alice e Bob);
- abbiano familiarità con griglie, con il riconoscimento di una casella come incrocio tra righe e colonne (ad esempio, hanno giocato almeno una volta a battaglia navale), e con i numeri naturali (per contare caselle, leggere coordinate);
- riconoscano e utilizzino alcuni colori per completare disegni su una griglia.

CONTENUTI, IN BREVE

Questo capitolo propone tre attività, a difficoltà crescente, in cui studenti e studentesse imparano a decodificare prima e rappresentare poi informazioni usando codifiche condivise. Le attività li guidano gradualmente dal ruolo di esecutore, che interpreta e applica una codifica data, al ruolo di programmatore, che crea e comunica una propria codifica affinché altri possano ricostruire correttamente le informazioni.

Per rendere l'esperienza più coinvolgente, è possibile inserire ogni attività in un contesto narrativo o ludico (messaggi segreti, mappe del tesoro, disegni nascosti, e così via).

Attività 1 – Capire il sistema di codifica: bambini e bambine hanno a disposizione alcune immagini su griglie di piccole dimensioni (ad esempio 2×2) associate a un sistema di codifica e, lavorando per ipotesi e confronti, ricostruiscono le regole del sistema di codifica (ad esempio l'ordine di lettura, il significato di eventuali simboli numerici o alfabetici presenti, associazione tra simboli e dati scelti per descrivere l'immagine).

Attività 2 – Pixel art: bambini e bambine lavorano con immagini in pixel art su griglie (fino a 10×10) per sperimentare due ruoli: l'“esecutore”, che decodifica una codifica data per ricostruire un disegno, e soprattutto il “programmatore”, che scrive la codifica di un'immagine (inventata o assegnata) seguendo regole e simboli condivisi.

Attività 3 – Luci di Natale: bambini e bambine prima confrontano diversi codici per rappresentare un insieme di messaggi per trovare quelli che permettono di capirsi sempre e quali sono più costosi. In una seconda parte dell'attività, progettano un nuovo codice condiviso che rispetti alcuni vincoli di brevità.

MODALITÀ DI LAVORO

Tutte le attività che seguono possono essere svolte in classe.

- Si alterneranno fasi di lavoro a coppie con momenti di confronto collettivo o a piccoli gruppi e sarà importante dare spazio sia alla fase esecutiva (seguire una codifica) sia a quella creativa (progettare una propria codifica), valorizzando le differenze tra le soluzioni.
- Sarà importante osservare lo svolgimento delle varie attività tenendo traccia delle parole e delle produzioni di alunni e alunne (ad esempio, fornendo schede strutturate per scrivere o disegnare, ma anche scattando foto alle loro produzioni) da usare a supporto di discussioni collettive.

Le Attività 1 e 2 propongono di svolgere codifica e decodifica di immagini e spesso faremo riferimento ad attività del Quaderno di Informatica in cui vengono proposti due diversi sistemi di codifica: per righe e per coordinate.

Codifica per righe. È usata nelle attività sulle pagine 14 (riportata di seguito come esempio), 15, 16 e 17.

REGOLE DI COLORAZIONE	
☒ Riga 1: 1B - 2R - 2B - 2R - 1B	LEGENDA COLORI B = bianco R = rosa V = verde G = giallo
☐ Riga 2: 1B - 1R - 1G - 2R - 1G - 1R - 1B	
☐ Riga 3: 1B - 1R - 4G - 1R - 1B	
☐ Riga 4: 1B - 2R - 2G - 2R - 1B	
☐ Riga 5: 2B - 4R - 2B	
☐ Riga 6: 3B - 2R - 3B	
☐ Riga 7: 2B - 3V - 3V	
☐ Riga 8: 3B - 3V - 2B	

Riga 1									
Riga 2									
Riga 3									
Riga 4									
Riga 5									
Riga 6									
Riga 7									
Riga 8									

Questo sistema di codifica funziona così: la griglia viene letta una riga alla volta da sinistra verso destra. Per ogni riga, la figura non viene descritta quadretto per quadretto, ma come una sequenza di caselle tutte dello stesso colore. Ogni sequenza di caselle è scritta con:

- un numero, che indica quanti quadretti consecutivi vanno considerati;
- una lettera, che indica il colore (legenda: B = bianco, R = rosa, V = verde, G = giallo).

Ad esempio, “1B – 2R – 2B – 2R – 1B” per la riga 1 significa che, in quella riga, ci sono nell’ordine: 1 quadretto bianco, poi 2 rosa, poi 2 bianchi, poi 2 rosa, poi 1 bianco.

Codifica per coordinate. È usata nelle attività delle pagine 18 (riportata di seguito come esempio), 19 e 20.

	1	2	3	4	5	6	7	8	9	10
A										
B										
C										
D										
E										
F										
G										
H										
I										
L										

GIALLO	
☒ (5, A)	☒ (7, A)
☐ (5, B)	☐ (6, B) ☐ (7, B)

AZZURRO	
☐ (3, A)	☐ (9, A)
☐ (3, B)	☐ (9, B)
☐ (3, C)	☐ (9, C)
☐ (3, D)	☐ (4, D) ☐ (8, D) ☐ (9, D)
☐ (4, E)	☐ (5, E) ☐ (6, E) ☐ (7, E)
☐ (8, E)	
☐ (5, E)	☐ (6, E) ☐ (7, E)

VIOLA	
☐ (4, A)	☐ (8, A)
☐ (4, B)	☐ (8, B)
☐ (4, C)	☐ (5, C) ☐ (6, C) ☐ (7, C) ☐ (8, C)
☐ (5, D)	☐ (6, D) ☐ (7, D)

VERDE	
☐ (3, F)	
☐ (3, G)	☐ (4, G) ☐ (6, G)
☐ (4, H)	☐ (5, H) ☐ (6, H)
☐ (6, I)	☐ (8, I) ☐ (9, I)
☐ (6, L)	☐ (7, L) ☐ (8, L)

Questo sistema di codifica funziona così: la griglia è formata da righe indicate con le lettere (A, B, C, ..., L) e colonne indicate con i numeri (1, 2, 3, ..., 10). Ad ogni casella dell'immagine, viene associata una coppia ordinata (numero, lettera) in cui il numero indica la colonna della casella, mentre la lettera indica la riga della casella.

Per decodificare l'immagine, si prende un colore alla volta tra quelli indicati e si colorano le celle che corrispondono alle coordinate nell'elenco corrispondente. Ad esempio, nella sezione *GIALLO*, la coppia **(5, A)** significa: "colora di giallo la casella che si trova nella colonna 5, riga A".

ATTIVITÀ 1: Capire il sistema di codifica

→ Vedi Quaderno di Informatica pp. 14-20

Data un'informazione, una scelta di dati ad essa associati e un sistema di codifica per rappresentare tali dati, il primo passo è quello di capire come funziona quel sistema di codifica.

Preparazione

- Stampare e ritagliare schede con varie informazioni rappresentate da dati codificati secondo un certo sistema di codifica da analizzare (o eventualmente completare/creare).
- Post-it (per disegnare o scrivere numeri, parole chiave).

Obiettivo

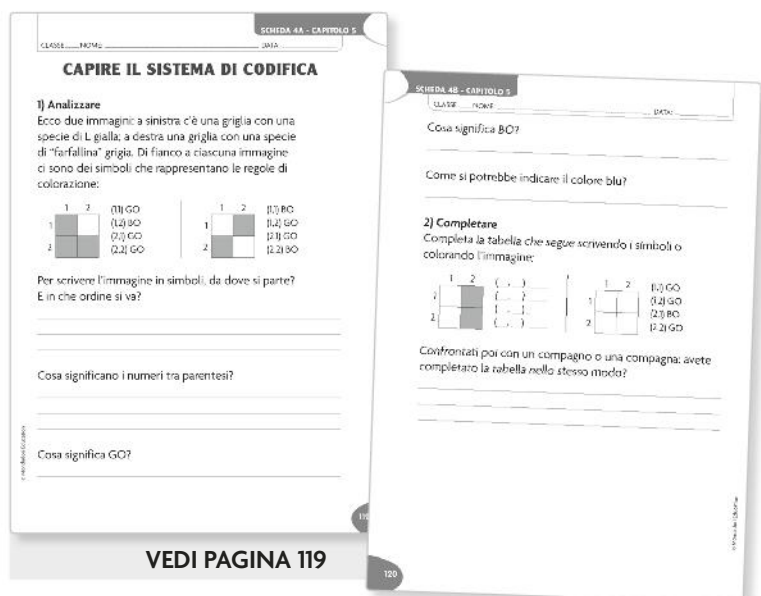
Rendere esplicito il collegamento tra i simboli usati nella codifica e i dati scelti per parlare di un'informazione.

Descrizione

Si predispongono una serie di schede, ciascuna contenente delle immagini che consistono in una griglia 2x2 con alcune caselle colorate con diverse combinazioni di colori e, di fianco, una sequenza di simboli che rappresenta la codifica simbolica di quell'immagine secondo un certo sistema di codifica. L'obiettivo sarà proprio quello di scoprire come funziona il sistema di codifica, esplicitando il legame tra l'immagine che si vuole codificare e la sequenza di simboli usata per codificarla.

Di seguito proponiamo un esempio di scheda, ma consigliamo di ideare diversi tipi di associazioni, con l'obiettivo di sottolineare sia l'arbitrarietà nella scelta dei simboli per codificare sia l'importanza della legenda per assicurare di decodificare i dati rappresentati senza ambiguità. Può anche essere utile proporre quesiti in cui si chiede di inventare un sistema di codifica associato alla sua legenda.

Nell'esempio della scheda proposta, l'informazione è l'immagine colorata; i dati sono i colori presenti in ciascuna cella della griglia e la posizione della cella; la codifica è il modo in cui trasformiamo posizione e colore in una sequenza di simboli. In particolare, la coppia di numeri indica la posizione della cella nella griglia (il primo numero è la riga, il secondo numero la colonna); le lettere indicano la prima



VEDI PAGINA 119

e l'ultima lettera del nome del colore (in italiano). Naturalmente, si tratta di un esempio di codifica, sarà possibile fare qualsiasi altra scelta.

Al termine dello svolgimento della scheda (o di altre versioni realizzate per la propria classe) sarà utile avviare una discussione per far emergere le riflessioni di alunni e alunne in relazione al sistema di codifica. In questo senso, le domande stimolo che seguono possono aiutare a guidare la discussione.

**DOMANDE STIMOLO**

- Cosa avete scoperto? Come funziona questo sistema di codifica?
- Cosa avete osservato per riuscire a capire come funzionavano i simboli?
- Che cosa sarebbe successo se fosse cambiato l'ordine con cui si leggono le celle? Proviamo a fare qualche esempio insieme.
- Quali altri modi vi vengono in mente per rappresentare il colore? Proviamo a fare qualche esempio insieme.
- A cosa serve avere una legenda?
- Esistono due disegni diversi che, con questo sistema di codifica, corrispondono agli stessi simboli?

Contenuto informatico

In questa attività, studenti e studentesse lavorano sull'idea che un'informazione (l'immagine colorata) venga prima descritta attraverso alcuni dati (i colori presenti nelle celle ordinate della griglia) e poi rappresentata con un codice simbolico. Per poter decodificare la sequenza di simboli è necessario sapere anche come usarli (ordine di lettura, significato dei numeri, significato delle lettere). L'attività mette in evidenza che:

- la scelta dei simboli e delle regole di codifica è arbitraria, ma deve essere condivisa e coerente per permettere la decodifica;
- lo stesso insieme di dati può essere rappresentato in modi diversi, a seconda del sistema di codifica scelto;
- per ricostruire correttamente l'informazione non basta conoscere l'insieme di simboli che rappresentano i dati scelti per esprimerla: servono anche le regole che legano i dati all'informazione (la legenda).

ATTIVITÀ 2: Pixel art

→ Vedi Quaderno di Informatica pp. 14-20

Le attività di pixel art sono estremamente note e diffuse ma, anche in questo contesto, distinguere tra il ruolo di "esecutore" e quello di "programmatore" è cruciale. Infatti, un conto è, data una griglia e un sistema di codifica noto, proporre a bambini e bambine di eseguire il processo di decodifica; un altro conto è codificare un'immagine data (o realizzata) in un certo sistema di codifica (che potrà essere già assegnato o inventato). Per far sì che il focus informatico sia in primo piano, è necessario dare molta più importanza al secondo tipo di proposte, cioè quelle in cui si assume il ruolo del "programmatore". Seguono alcuni esempi di spunti di attività.

Preparazione

- Griglie vuote a dimensioni variabili ed eventualmente crescenti (consigliamo comunque di non superare il 10x10).

- Schede con codifiche date da analizzare e da usare per decodificare le immagini di partenza.
- Griglie con immagini visibili (disegni in pixel art) a dimensioni variabili ed eventualmente crescenti (consigliamo comunque di non superare il 10x10).
- Schemi da compilare per scrivere la codifica (ad esempio tabelle con colonne “riga/colonna/colore” o spazi per scrivere le sequenze di simboli).

Descrizione di possibili attività

Dato il sistema di codifica, scoprire l'immagine attraverso la decodifica.

In questo tipo di proposta vengono fornite griglie vuote, accompagnate da una codifica già scritta in simboli secondo un certo sistema di codifica (ad esempio una lista di coordinate e colori, o una sequenza di simboli per ciascuna riga della griglia in cui ogni combinazione di simboli indica il colore di una cella e il numero di celle di quel colore). A studenti e studentesse viene chiesto di colorare la griglia seguendo alla lettera le istruzioni, per “scoprire” l'immagine nascosta. Questa attività, eventualmente anticipata da un'analisi del sistema di codifica (cfr. Attività 1 del presente capitolo), può essere usata:

- come introduzione per familiarizzare con il sistema di codifica scelto;
- per mettere in evidenza il ruolo dell'ordine di lettura (per righe, per colonne ecc.) e la necessità di una legenda chiara.



SUL QUADERNO DI INFORMATICA

Puoi proporre ai tuoi alunni e alunne le schede di pag. 14 e 18 del Quaderno di Informatica.

Dato il sistema di codifica, codificare un'immagine.

In questo tipo di proposta, dopo aver concordato un sistema di codifica, ad alunni e alunne è richiesto di scrivere la codifica di un'immagine data, rispettando tutte le regole (ordine, simboli, convenzioni). L'immagine di partenza in questo caso viene consegnata già fatta e:

- se viene assegnata a tutti la stessa immagine, si potranno poi confrontare le varie codifiche scritte;
- se si assegnano immagini diverse, una volta che ciascuno avrà scritto la propria codifica, sarà possibile scambiarsi le codifiche prodotte. In questo modo, ognuno prova a decodificare l'immagine a partire dalla sequenza di simboli scritta da qualcun altro, e poi si confronta il disegno originale con quello riprodotto per verificare se la codifica è stata scritta correttamente.



SUL QUADERNO DI INFORMATICA

Per allenarsi sul processo di codifica, puoi proporre ai tuoi alunni e alunne le schede di pag. 15 e 19 del Capitolo RAPPRESENTARE INFORMAZIONI del Quaderno di Informatica.

Inventare, codificare e decodificare immagini.

In questo tipo di proposta, dopo aver concordato un sistema di codifica, gli alunni e le alunne danno sfogo alla propria fantasia: ciascuno inventa un proprio disegno, lo codifica usando il sistema indicato e poi scambia la propria codifica con un compagno o una compagna che dovrà riprodurre l'immagine decodificando; alla fine, si confronta il disegno originale con quello riprodotto tramite decodifica per verificare che l'esito ed eventualmente correggere dove necessario.



SUL QUADERNO DI INFORMATICA

Per farlo puoi proporre ai tuoi alunni e alunne le schede di pag. 16, 17 e 20 del Quaderno di Informatica.

Riflessione alla fine delle attività.

Al termine di ciascuna attività potrà essere utile avviare una discussione per fare emergere le riflessioni di alunni e alunne in relazione all'esperienza fatta.



DOMANDE STIMOLO

- Quando decodifichi per scoprire l'immagine, qual è la prima cosa che fai prima di iniziare a colorare?
- Se una persona vuole capire se sta sbagliando mentre lavora, senza avere l'immagine finale, qual è la prima cosa che può controllare?
- Se una persona scrive la codifica di un'immagine per un compagno o una compagna, che cosa non deve mancare mai perché l'altra persona possa ricostruirla correttamente?
- Vi viene in mente una situazione in cui il sistema di codifica che abbiamo usato potrebbe essere ambiguo? E come potremmo modificarlo per renderlo più preciso?
- Come fareste a spiegare a parole, a qualcuno che non lo conosce, come usare questo sistema di codifica per scoprire l'immagine nascosta?
- (Nell'attività in cui ci si scambiano le codifiche) Dopo aver visto il tentativo del compagno o della compagna, che cosa cambiereste nel vostro modo di scrivere la codifica o nelle regole del sistema?

Contenuto informatico

In questa attività, l'immagine in pixel art rappresenta l'informazione, mentre i pixel (descritti da posizione e colore) sono i dati scelti per descriverla. Il sistema di codifica stabilisce come trasformare questi dati in simboli (per riga, liste di coordinate ecc.) e come, a partire dai simboli, ricostruire l'immagine tramite la decodifica. Lavorando sia come "esecutori" sia come "programmatori", alunni e alunne hanno la possibilità di sperimentare che: uno stesso disegno può essere rappresentato in modi diversi; per ricostruire correttamente l'informazione, cioè per usare correttamente il sistema di codifica, servono sia i simboli sia le regole d'uso (ordine di lettura, legenda); cambiando sistema di codifica cambiano anche caratteristiche importanti della rappresentazione, come la sua lunghezza, la facilità di lettura per una persona o la semplicità di elaborazione da parte di una macchina.

ATTIVITÀ 3: Luci di Natale

→ Vedi Quaderno di Informatica pp. 21-23

**SUL QUADERNO DI INFORMATICA**

L'attività qui descritta fa riferimento alle pagine da 21 a 23 del Capitolo RAPPRESENTARE INFORMAZIONI del Quaderno di Informatica.

Obiettivi

- Riconoscere che la stessa informazione può essere rappresentata con codici diversi, che hanno caratteristiche differenti.
- Confrontare codici diversi in termini di chiarezza e lunghezza (quante lettere servono), individuando quali permettono di capirsi sempre e quali possono generare ambiguità.
- Progettare un codice (che sia condiviso e non ambiguo) e verificarne il funzionamento provandolo in classe.

Descrizione

Dopo aver letto la storia e contestualizzato la situazione, nella prima fase di lavoro alunni e alunne possono analizzare i 5 codici proposti eventualmente rispondendo per iscritto a delle domande guida. A livello metodologico, può essere utile iniziare proponendo di condurre l'analisi individualmente e poi confrontare le riflessioni - prima lavorando in piccoli gruppi e poi in plenaria (raccolgendo e commentando tutte le varie produzioni realizzate).

**DOMANDE STIMOLO (ispirate a quelle sul Quaderno di Informatica)**

- Quali codici permettono ad Alice e Bob di capirsi sempre, senza mai sbagliarsi? Da cosa lo hai capito?
- Tra i codici che permettono ad Alice e Bob di capirsi, quale permette loro di accendere meno lettere possibili, così da far prima? Come fai a dirlo?

I 5 codici proposti e la richiesta di inventarne uno

I codici proposti nell'attività sono stati progettati apposta per riflettere su univocità e costo:

- Per il codice 1, che richiede di scrivere tutta la frase lettera per lettera, abbiamo la massima chiarezza ma anche il massimo "costo".
- Per il codice 2, che richiede di scegliere a caso una parola della frase e scriverla lettera per lettera, univocità e costo dipendono dalla parola scelta.
- Per il codice 3, che richiede di usare solo la prima lettera di ogni frase, la lunghezza della sequenza dei simboli (e quindi il costo) è minore rispetto a quello degli altri codici, ma più messaggi possono avere la stessa lettera iniziale (es. la N) e quindi si tratta di un codice non univoco.
- Il codice 4, che richiede di usare alcune parti di frase prestabilite (pezzi iniziali o centrali), lettera per lettera, è certamente univoco ma costoso.
- Il codice 5, che richiede di associare a ogni messaggio una sigla di due lettere secondo una tabella condivisa, è univoco ed è quello a minor costo tra quelli univoci proposti.

La seconda fase di lavoro richiede di inventare un nuovo codice in cui ogni messaggio è rappresentato da una sola lettera. Questa richiesta è cruciale per attivare un'analisi dei messaggi e una scelta dei simboli che sia strategica rispetto all'obiettivo di avere la massima chiarezza al costo minore. Infatti, basta una lettera o un simbolo diverso per ogni frase, per esempio

A B C D E F, purché siano simboli su cui Alice e Bob sono d'accordo e per i quali entrambi abbiano a disposizione la legenda.



ASPETTO A CUI FARE ATTENZIONE

La scelta dei simboli è, in generale, arbitraria. Decidere di codificare ciascun messaggio usando un simbolo qualsiasi è perfettamente accettabile, a patto che sia stata condivisa l'associazione messaggio - simbolo che lo codifica.

Contenuto informatico

È possibile modellizzare l'attività in questo modo: il messaggio che Alice e Bob vogliono scambiarsi rappresenta l'informazione; i dati sono le sequenze di lettere che descrivono i vari messaggi e le luci sono il modo in cui i dati vengono fisicamente rappresentati. La domanda che ci si sta ponendo in questa attività è quindi: quali dati sono informativi per Alice e Bob? E quale rappresentazione del dato è la meno costosa?

In questa attività alunni e alunne quindi sperimentano che:

- per potersi capire sempre serve un codice condiviso e non ambiguo (nessun messaggio deve “collidere” con un altro);
- codici diversi hanno costi diversi: alcuni richiedono molte lettere/luci (più tempo, più “energia”); altri sono più compatti ma meno chiari;
- progettare un buon codice significa trovare un compromesso tra brevità e chiarezza, tenendo conto del contesto (Quanti messaggi devo distinguere? Quali rischi di confusione posso accettare?).

In termini più generali, l'attività introduce l'idea informatica che la stessa informazione può essere codificata in modi diversi, e che la progettazione di un codice riguarda sia le regole di corrispondenza (tabella messaggio–sigla) sia le proprietà desiderate della rappresentazione (univocità, lunghezza, semplicità d'uso).

POSSIBILI ESTENSIONI, VARIAZIONI E PERSONALIZZAZIONI

Spunti per adattare l'attività

In caso di difficoltà:

- Segmentare l'analisi del sistema di codifica eventualmente aggiungendo delle domande guida esplicite che possano orientare l'attenzione.
- Proporre alternative multimodali (per esempio nelle attività di pixel art sostituendo i colori con materiali con diverse texture, come velluto, carta crespata, cartone zigrinato, e così via).

Per rendere l'attività più sfidante:

- Si può proporre di inventare un sistema di codifica;
- Si possono proporre esempi svolti che contengono errori, chiedendo ai bambini e alle bambine di individuarli e correggerli.

ASPETTI INFORMATICI

A prima vista queste attività possono sembrare semplici, ma in realtà avvicinano bambini e bambine ad alcune idee fondamentali dell'informatica. Ogni informazione – che sia un disegno, un suono, un testo o un video – può infatti essere descritta attraverso simboli e regole condivise: è quello che in informatica chiamiamo **codifica**. Seguendo la prospettiva

proposta in Lonati e Monga [2025], possiamo dire che scegliamo alcuni dati per esprimere un'informazione, e poi decidiamo con quale formato o codice rappresentare questi dati. Da soli, i simboli scelti per rappresentare un dato non “dicono” niente: diventano interpretabili solo se chi li legge conosce il funzionamento del sistema di codifica (con una legenda dei simboli e una spiegazione di come funziona la descrizione simbolica). Quando codifichiamo un insieme di dati, trasformiamo quei dati (che sono stati scelti per esprimere una certa informazione) in una sequenza di simboli seguendo il sistema di codifica; quando interpretiamo una sequenza di simboli per ricostruire il dato di partenza, facciamo l'operazione inversa, cioè la **decodifica**. Nelle attività proposte in questo capitolo, alunni e alunne hanno la possibilità di sperimentare sia il processo di codifica che quello di decodifica e in ogni passaggio gli errori sono sempre un'occasione preziosa per riflettere su quanto sia importante che le convenzioni usate siano condivise e su come renderle più chiare.

Un altro elemento importante su cui le attività proposte permettono di lavorare è che non c'è un unico modo per rappresentare un certo dato che esprime un'informazione. Un'immagine, ad esempio, può essere scomposta in un reticolo di quadrati e descritta con pixel colorati. Ogni metodo ha i suoi punti di forza e di debolezza: uno può essere più veloce, un altro più compatto, uno più facile da capire per una persona e un altro più facile da elaborare in maniera automatica. In questo modo si apre la strada anche al tema dell'**efficienza**: possiamo esprimere la stessa cosa usando più/meno simboli, senza perdere le informazioni che ci interessano, proprio come avviene nel caso della compressione di un file per occupare meno spazio.

In tutte le attività ritorna lo stesso filo rosso, che può guidare la discussione in classe:

- Quale informazione mi interessa?
- Quali dati scelgo per esprimerla?
- Quale codifica scelgo, in modo che sia chiara (univoca) ma anche conveniente (con un costo accettabile)?

Le attività invitano inoltre a un uso creativo del modo in cui si rappresentano i dati che esprimono informazioni. Non si tratta solo di applicare codifiche già date, ma di confrontarle, di inventarne di nuove, di sperimentare soluzioni e di vedere cosa succede quando si cambia prospettiva.

Nell'ambito dell'elaborazione digitale dell'informazione, questi temi intervengono ad esempio nella gestione delle immagini. Alcuni formati, come il bitmap o il PNG, funzionano “a pixel”, in modo simile a quanto avviene nella pixel art. L'immagine viene scomposta in un reticolo di quadrati (chiamati pixel) ciascuno dei quali ha un colore preciso. Il numero di quadrati che compongono il reticolo è legato alla risoluzione dell'immagine e, in questi formati, se ingrandiamo troppo le immagini diventano “sgranate” proprio perché il numero di pixel disponibili è fissato dalla risoluzione. Altri formati chiamati vettoriali (come gli SVG), funzionano invece “a forme”: descrivono linee e curve che “modellizzano” l'immagine, un po' come quando si uniscono i puntini per creare un disegno. In questo caso, l'immagine rimane sempre nitida anche quando viene ingrandita. Non a caso, i loghi e i disegni geometrici si salvano quasi sempre in formato SVG, perché così potranno essere stampati su supporti di qualsiasi dimensione (un biglietto da visita o su un grande cartellone) senza perdere qualità.

RELAZIONE CON L'INFORMATICA NELLE INDICAZIONI NAZIONALI

Relativamente alle Indicazioni Nazionali 2025, queste attività contribuiscono allo sviluppo delle seguenti competenze e obiettivi.

Competenze (alla fine della scuola primaria) nella disciplina Matematica:

- *Esprimere informazioni mediante dati di varia natura e codificare tali dati anche digitalmente.*

Obiettivi (alla fine della terza primaria) nella disciplina Matematica

- *Scegliere ed utilizzare oggetti o simboli per rappresentare informazioni.*

Infine, lavorano sulle seguenti conoscenze (inserite nelle Indicazioni Nazionali 2025 a titolo di esempio, non prescrittive)

- *Rappresentazione di dati e informazioni e loro differenza.*

VALORE COSTRUZIONISTA

Le attività sulla rappresentazione delle informazioni permettono a bambini e bambine di passare dal semplice seguire regole alla creazione di immagini e codifiche da condividere con altre persone. La possibilità di apprendere nasce dalla costruzione di artefatti significativi – immagini, codifiche, messaggi – che assumono valore perché devono essere condivisi e compresi da altri.

Gli errori non sono ostacoli, ma occasioni per fare debugging: riflettere sulla chiarezza delle regole, migliorare le proprie convenzioni e sperimentare soluzioni alternative. Confrontando codifiche diverse, bambini e bambine scoprono che uno stesso contenuto può essere rappresentato in più modi, ciascuno con vantaggi e limiti, sviluppando così consapevolezza critica e capacità di scelta.

Il valore costruzionista sta soprattutto nel passaggio di ruolo: da esecutori che applicano un codice dato, a programmatori che progettano e ottimizzano le proprie rappresentazioni, fino a inventori di linguaggi condivisi. In questo percorso gli alunni non imparano solo a “seguire istruzioni”, ma anche a riflettere sul proprio modo di pensare, a comunicarlo in modo efficace e a materializzarlo in prodotti condivisi e discutibili insieme.

SUGGERIMENTI PER ULTERIORI ATTIVITÀ E RIFERIMENTI

- Violetta Lonati e Mattia Monga. *Rendere concreta l'informazione per elaborarla*. Nuova Secondaria. 43:3(2025), pp. 74-79.
- Nel progetto *Programma il Futuro* ci sono alcune proposte didattiche in linea con quanto visto in questo capitolo, ad esempio: <https://q3.hubscuola.it/0rv>
L'attività, attraverso il disegno su carta a quadretti, esemplifica come le macchine interpretino istruzioni precise, evidenziando le difficoltà della programmazione e l'importanza del debug. Potrebbe essere usata come attività ponte tra i concetti informatici presentati in questo capitolo (relativi alla rappresentazione di dati che esprimono informazioni) e quelli oggetto del Capitolo 3 (in cui si è parlato delle caratteristiche della lettura algoritmica di un compito) e del Capitolo 4 (in cui si è parlato di movimento su griglie).
- Il laboratorio *ALaDDIn* dell'Università degli Studi di Milano propone numerose attività che permettono di lavorare sui temi trattati in questo capitolo. Un esempio si trova nell'articolo di Bellettini, Lonati, Malchiodi, Monga e Morpurgo [2019] <https://q3.hubscuola.it/imi>
L'attività *Rettangoli* descritta nella sezione 4 è completamente centrata sul tema della rappresentazione delle informazioni tramite codifiche simboliche. È pensata per il secondo ciclo di scuola primaria ma, con opportuni adattamenti, può ispirare altre attività analoghe anche per il primo triennio.

CAPITOLO 6: STRATEGIE DI SOLUZIONE DI PROBLEMI

Nel Capitolo 3, le alunne e gli alunni hanno lavorato su situazioni quotidiane e giochi in cui hanno sperimentato l'idea di descrivere sequenze di azioni e di osservare ciò che accadeva eseguendole. Nel Capitolo 4, hanno scritto sequenze di istruzioni (tramite un preciso formalismo iconico) per descrivere percorsi di un personaggio su una griglia. Hanno quindi sperimentato che dare istruzioni non significa “dire a parole cosa fare”, ma costruire una sequenza precisa di passi che un esecutore possa seguire senza intuire nulla di non detto. In questo capitolo faremo un passo in più: non ci interessa solo trovare una soluzione, ma anche una buona strategia. Le alunne e gli alunni cominceranno anche a pensare a strategie più generali per risolvere classi di problemi (es. trovare il più pesante tra certi oggetti, ordinare una sequenza).

TRAGUARDI E OBIETTIVI GENERALI

- Costruire una strategia per risolvere un problema o una classe di problemi.
- Rappresentare e comunicare una strategia sotto forma di sequenza di istruzioni.
- Confrontare strategie diverse: osservare che possono esserci più soluzioni corrette.
- Verificare se una strategia funziona “sempre” (su più prove, con dati diversi), non solo una volta.

PREREQUISITI

- Aver lavorato su sequenze e sul fatto che “l'ordine conta”.
- Familiarità con l'idea di *Manuale delle istruzioni* (insieme limitato di azioni disponibili).
- Saper gestire (anche in modo informale) i confronti: più pesante/più leggero, più lungo/più corto, più piccolo/più grande.

CONTENUTI, IN BREVE

Attività 1 - Trova il sacchetto più pesante: si lavorerà sul progettare una strategia per trovare l'elemento più pesante, potendo solo effettuare confronti a coppie (es. con una gruccia usata come bilancia a due piatti).

Attività 2 - Il barista ordinato: si lavorerà sul trovare e confrontare diverse strategie per ordinare oggetti (es. cannucce di lunghezze diverse), potendo solo effettuare confronti e scambi tra due oggetti.

PROBLEMA, ALGORITMO, STRATEGIA

Come visto nel Capitolo 2, in informatica si vogliono spesso risolvere problemi generali. La soluzione non è un numero (o più in generale un oggetto), ma un procedimento, un algoritmo. Con strategia intendiamo, in modo informale, la scelta di quali istruzioni costituiscono un algoritmo.

Due strategie diverse possono dare la stessa risposta ma una delle due può essere più efficiente (meno passi, meno confronti, meno errori), più chiara, più elegante, più semplice da capire.

In generale, una buona strategia usa le informazioni che otteniamo passo dopo passo e ci fa eliminare le possibilità inutili.

ATTIVITÀ 1: Trova il sacchetto più pesante (gruccia-bilancia)

→ Vedi Quaderno di Informatica pp. 24-25

**SUL QUADERNO DI INFORMATICA**

Puoi far lavorare le alunne e gli alunni, in classe o a casa, sulla scheda TROVARE IL PIÙ PESANTE (pag. 24 e 25 del Quaderno di Informatica).

Obiettivo

Sviluppare una strategia per trovare il sacchetto più pesante tra quattro sacchetti, rispettando vincoli precisi e usando un *Manuale delle istruzioni* per esprimerla come algoritmo.

Che cosa serve

- Una gruccia robusta da usare come “bilancia”. La gruccia deve dare la possibilità di appendere ai due lati, in modo bilanciato, due sacchetti (ad esempio, usa una gruccia con le pinze).
- 4 sacchetti identici che non permettano di intravedere il contenuto.
- Materiale per riempirli con pesi diversi (riso, pasta, sassolini, monete, frutta...), in modo che i pesi dei sacchetti siano tutti diversi, ma abbastanza simili tra loro.
- 2 scatole (o spazi delimitati) etichettate:
 - SACCHETTI DA PESARE
 - SACCHETTI ELIMINATI

Preparazione

- Prepara quattro sacchetti che siano all'apparenza “quasi uguali”, ma che abbiano pesi tutti diversi.
- Verifica prima tu che la gruccia funzioni bene: appesa a un filo o tenuta con il gancio appoggiato a un dito deve restare in perfetto equilibrio. Con due sacchetti appesi, deve “pendere” in modo chiaro dal lato di quello più pesante.

**ASPETTO A CUI FARE ATTENZIONE**

I sacchetti devono apparire il più possibile simili: se si distinguono “a occhio” o “a tatto” l'attività perde senso.

Regole del gioco (da esplicitare in classe)

- All'inizio, tutti e 4 i sacchetti sono nella scatola “SACCHETTI DA PESARE”.
- Esiste anche la scatola “SACCHETTI ELIMINATI”.
- In ogni momento, puoi avere in mano un solo sacchetto.
- Alla fine, sulla gruccia deve restare solo il sacchetto più pesante.

Fase 1 – Capire il Manuale delle istruzioni

Leggi e commenta in classe il Manuale delle istruzioni. Le uniche istruzioni che l'esecutore potrà compiere sono le seguenti:

PRENDI IN MANO UN SACCHETTO DAI SACCHETTI DA PESARE

APPENDI IL SACCHETTO CHE HAI IN MANO ALLA GRUCCIA

PRENDI IN MANO IL SACCHETTO PIÙ LEGGERO TRA I DUE SULLA GRUCCIA

METTI IL SACCHETTO CHE HAI IN MANO NEI SACCHETTI ELIMINATI
--

**ASPETTO A CUI FARE ATTENZIONE**

Qui è utile ribadire un'idea già incontrata: l'esecutore deve seguire solo queste istruzioni, nell'ordine scelto dal programmatore.

Fase 2 – Ideazione e test di una strategia

In piccoli gruppi:

- un'alunna o un alunno pensa a una strategia e dà le istruzioni (programmatore);
- un'altra o un altro le esegue materialmente (esecutore);
- una terza o un terzo controlla che le regole siano rispettate (un solo sacchetto in mano, scatole ecc.).

Quando si arriva alla fine, il gruppo verifica (per esempio usando una bilancia elettronica, oppure se era stato annotato in precedenza dall'insegnante) se:

- sulla gruccia è rimasto un solo sacchetto;
- è davvero il più pesante.

**ASPETTO A CUI FARE ATTENZIONE**

È normale che alcuni bambini provino strategie diverse (ad esempio togliere e rimettere sacchetti "per controllare"). Non è un problema: anzi, è un ottimo punto di discussione su cosa significa "avere una strategia" e su come semplificarla.

Quando il gruppo è soddisfatto, potrà completare la scheda disegnando le icone corrispondenti alle azioni.

Fase 3 – Verifica con quattro sacchetti nuovi

Se c'è tempo, è utile verificare che la strategia funzioni anche con sacchetti di pesi diversi da quelli originariamente usati durante l'ideazione.

Fase 4 – Discussione: perché funziona?

In questa fase è importante favorire una riflessione sulla strategia naturale che sarà probabilmente emersa.

**DOMANDE STIMOLO**

- Dopo la prima pesata, quanti sacchetti potrebbero ancora essere il più pesante?
- Il sacchetto eliminato può essere il più pesante? Perché?
- Se ripeti sempre lo stesso tipo di ragionamento, cosa succede dopo 3 eliminazioni?

Una strategia corretta

Nella scheda, a pag. 25 del sussidiario, sono previsti 14 passi. Una strategia molto naturale (e coerente con i vincoli) è questa:

- Metto sulla gruccia due sacchetti.
- Elimino quello più leggero.
- Metto sulla gruccia un nuovo sacchetto contro quello rimasto.
- Elimino di nuovo il più leggero.
- Ripeto finché ne resta uno.

IDEA DI SPIEGAZIONE

Sulla gruccia resta sempre un “campione” (quello più pesante tra quelli visti finora). Ogni nuovo sacchetto viene sfidato contro il campione.

Sequenza di istruzioni (1–14):

1. PRENDI IN MANO UN SACCHETTO DAI SACCHETTI DA PESARE
2. APPENDI IL SACCHETTO CHE HAI IN MANO ALLA GRUCCIA
3. PRENDI IN MANO UN SACCHETTO DAI SACCHETTI DA PESARE
4. APPENDI IL SACCHETTO CHE HAI IN MANO ALLA GRUCCIA
5. PRENDI IN MANO IL SACCHETTO PIÙ LEGGERO TRA I DUE SULLA GRUCCIA
6. METTI IL SACCHETTO CHE HAI IN MANO NEI SACCHETTI ELIMINATI
7. PRENDI IN MANO UN SACCHETTO DAI SACCHETTI DA PESARE
8. APPENDI IL SACCHETTO CHE HAI IN MANO ALLA GRUCCIA
9. PRENDI IN MANO IL SACCHETTO PIÙ LEGGERO TRA I DUE SULLA GRUCCIA
10. METTI IL SACCHETTO CHE HAI IN MANO NEI SACCHETTI ELIMINATI
11. PRENDI IN MANO UN SACCHETTO DAI SACCHETTI DA PESARE
12. APPENDI IL SACCHETTO CHE HAI IN MANO ALLA GRUCCIA
13. PRENDI IN MANO IL SACCHETTO PIÙ LEGGERO TRA I DUE SULLA GRUCCIA
14. METTI IL SACCHETTO CHE HAI IN MANO NEI SACCHETTI ELIMINATI

Possibile estensione: generalizzazione

La sequenza numerata qui sopra è una soluzione immediata e molto esplicita: funziona bene per una prima discussione, perché rende visibili tutti i passaggi e aiuta gli alunni a “fidarsi” del procedimento.

Però, appena ci si sposta da quattro sacchetti a tanti sacchetti, la stessa idea diventa lunga da scrivere. Si crea così un’ottima occasione per introdurre (in modo leggero) il concetto di ripetizione: quando un blocco di istruzioni si ripete “uguale” molte volte, possiamo scriverlo una volta sola e dire quando ripeterlo. Seguono due esempi.

Versione con “salto” (torna indietro)

Qui l’idea è: faccio il confronto tra due sacchetti, elimino quello più leggero, poi ricomincio dal confronto con il sacchetto “campione” rimasto sulla gruccia finché non finiscono i sacchetti da pesare.

1. PRENDI IN MANO UN SACCHETTO DAI SACCHETTI DA PESARE
2. APPENDI IL SACCHETTO CHE HAI IN MANO ALLA GRUCCIA
3. PRENDI IN MANO UN SACCHETTO DAI SACCHETTI DA PESARE
4. APPENDI IL SACCHETTO CHE HAI IN MANO ALLA GRUCCIA
5. PRENDI IN MANO IL SACCHETTO PIÙ LEGGERO TRA I DUE SULLA GRUCCIA
6. METTI IL SACCHETTO CHE HAI IN MANO NEI SACCHETTI ELIMINATI
7. SE CI SONO ANCORA SACCHETTI DA PESARE: TORNA ALL’ISTRUZIONE 3

Versione con “ripeti finché”

Ai programmatori di oggi non piace molto utilizzare istruzioni di salto come TORNA ALL'ISTRUZIONE 3.

Si può allora utilizzare una ripetizione con condizione (vedi Capitolo 2).

PRENDI IN MANO UN SACCHETTO DAI SACCHETTI DA PESARE

APPENDI IL SACCHETTO CHE HAI IN MANO ALLA GRUCCIA

FINCHÉ CI SONO ANCORA SACCHETTI DA PESARE:

PRENDI IN MANO UN SACCHETTO DAI SACCHETTI DA PESARE

APPENDI IL SACCHETTO CHE HAI IN MANO ALLA GRUCCIA

PRENDI IL SACCHETTO PIÙ LEGGERO TRA I DUE SULLA GRUCCIA

METTI IL SACCHETTO CHE HAI IN MANO NEI SACCHETTI ELIMINATI

Ulteriore estensione

E se due sacchetti avessero esattamente lo stesso peso? Qui è interessante discutere del fatto che l'algoritmo debba prevedere cosa fare in quel caso, altrimenti l'esecutore non sa come comportarsi.

Contenuto informatico

La ricerca del massimo in una sequenza è un problema tipico dell'informatica e contiene alcuni elementi ricorrenti in algoritmi e programmi: scorrere una sequenza, su ciascun elemento compiere un'azione se si verifica una certa condizione.

ATTIVITÀ 2: Il barista ordinato

→ Vedi Quaderno di Informatica pp. 26-27

**SUL QUADERNO DI INFORMATICA**

Puoi far lavorare le alunne e gli alunni, in classe o a casa, sulla scheda IL BARISTA ORDINATO (pag. 26 e 27 del Quaderno di Informatica).

Obiettivo

Inventare strategie per ordinare 5 elementi secondo un criterio (dalla cannuccia più corta, che dovrà stare a sinistra, alla più lunga, che dovrà stare a destra) avendo a disposizione solo la possibilità di confrontare ed eventualmente scambiare due cannucce alla volta.

Che cosa serve

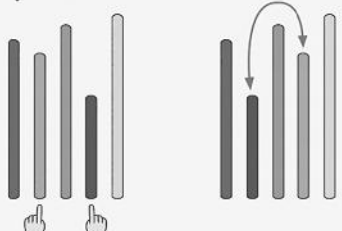
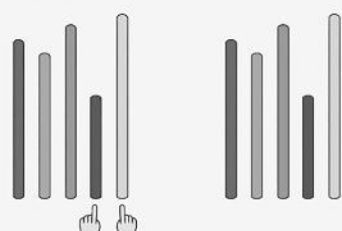
- 5 cannucce di carta di 5 colori diversi.
- Forbici.
- Un foglio di carta (per la Fase 2 “cannucce coperte”).

Preparazione

- Tagliare 5 cannucce di lunghezze tutte diverse.
- Scegliere lunghezze abbastanza distanti tra loro, così il confronto è netto.

Capire il Manuale delle istruzioni

Il Manuale delle istruzioni in questo caso presenta una sola istruzione, che deve essere compresa molto bene dalle alunne e dagli alunni. In questa attività, l'insegnante avrà il ruolo dell'aiutante.

MANUALE DELLE ISTRUZIONI PER ORDINARE LE CANNUCCE		
INDICHI DUE CANNUCCE	Se la cannuccia alla tua sinistra è più lunga di quella alla tua destra, l'aiutante le scambia di posto.	Se la cannuccia alla tua sinistra è già più corta di quella alla tua destra, l'aiutante non fa nulla.
		

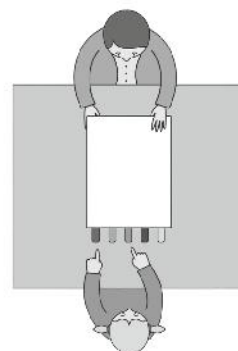
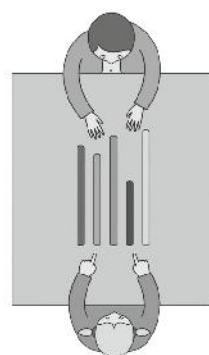
Svolgimento

Fase 1 - Cannucce scoperte

1. Un adulto mette le 5 cannucce in un ordine casuale.
2. I bambini e le bambine indicano due cannucce alla volta.
3. L'adulto applica la regola di scambio.
4. Si prosegue finché le cannucce risultano ordinate.

Fase 2 – Cannucce coperte

1. Mentre le alunne e gli alunni non guardano, un adulto taglia 5 cannucce (tutte diverse) e le dispone in ordine casuale.
2. Le allinea e copre la parte finale con un foglio o un qualsiasi altro oggetto adeguato: le alunne e gli alunni vedono i colori e le posizioni, ma non la lunghezza.
3. I bambini indicano due cannucce: l'adulto decide se scambiarle o meno seguendo il *Manuale delle istruzioni*, ovviamente senza mostrare la lunghezza.
4. Quando pensano che siano ordinate, chiedono di scoprirle.
5. Se non sono ordinate, continuano.



ASPETTO A CUI FARE ATTENZIONE

Problemi tipici potrebbero essere:

- supporre che dopo la prima passata, le cannucce siano già tutte ordinate;
- non riuscire a trovare una regola per decretare che le cannucce sono ordinate e non è più necessario confrontarne oltre.

Modalità di lavoro

Importanza dell'aiutante adulto

L'aiutante deve essere un adulto per vari motivi:

- alunne e alunni potrebbero confondere sinistra e destra, specialmente se messi uno di fronte all'altro;
- nella Fase 2, l'aiutante deve contemporaneamente tenere coperte le cannucce e scambiarle.

Per questo motivo, se l'attività viene svolta in classe, è opportuno che, ad esempio, l'insegnante lavori con ciascun alunno a turno, magari mentre il resto della classe osserva le scelte effettuate dalla compagna o dal compagno che sta giocando in quel momento.

A differenza dell'Attività 1, questa attività è pensata perché gli studenti inizino a esplorare l'idea che esistono molte strategie diverse per risolvere un problema o raggiungere un obiettivo.

Differenza tra Fase 1 e Fase 2

Nella Fase 1 le cannucce sono visibili: i bambini vedono subito quale cannuccia è più lunga e possono basare la strategia sul confronto “a colpo d'occhio”, per esempio cercando la più corta e spostandola all'inizio (tramite uno scambio tra essa e quella in prima posizione), poi la seconda più corta, e così via. In altre parole, la strategia può essere **globale**: si osserva l'insieme e si prende una decisione generale. Questa fase è importante per iniziare a ideare una strategia, che però dovrà essere raffinata con i vincoli successivi.

Nella Fase 2, infatti, le cannucce sono coperte. Non si può più scegliere “la più corta” o “la più lunga” guardandole: l'unica informazione disponibile proviene dal confronto tra due cannucce alla volta, e l'adulto decide se scambiarle. Trovare una strategia globale è impossibile e dobbiamo affidarci a un approccio **locale**: si prova uno scambio, si osserva l'effetto, e si decide passo per passo come continuare. Questo cambia profondamente il modo di ragionare sul problema: non si tratta più di individuare subito l'ordine corretto, ma di usare le informazioni che deduciamo dai confronti per avvicinarsi gradualmente alla soluzione.

Possibili strategie e aspetti a cui fare attenzione (soprattutto per la Fase 2)

La ricerca mostra che le bambine e i bambini tra i 4 e i 10 anni riescono a scoprire da soli strategie efficienti per ordinare gli oggetti, anche quando l'informazione sulla loro lunghezza è nascosta. La loro accuratezza aumenta con l'età [Yang et al., 2025].

Mostriamo due delle possibili strategie che potrebbero essere proposte dagli alunni e dalle alunne, descrivendole in modo volutamente informale, lasciando così alla classe la libertà di scegliere le modalità linguistiche per esprimerle.

Una prima strategia consiste nel fare più “passate” da sinistra a destra, scorrendo le coppie di cannucce vicine, confrontandole ed eventualmente scambiandole. Per 5 cannucce, ogni passata consiste nelle istruzioni seguenti:

- Si confronta la prima cannuccia con la seconda, scambiandole se la seconda è più corta della prima.
- Ora si confronta la seconda con la terza cannuccia, eventualmente scambiandole.
- Ora si confronta la terza con la quarta cannuccia, eventualmente scambiandole.
- Ora si confronta la quarta con la quinta cannuccia, eventualmente scambiandole.

Si ripete la passata più volte; quando durante una passata non avvengono scambi, le cannucce sono ordinate e l'algoritmo termina.

La ricerca suggerisce che spesso, finita una passata “da sinistra a destra”, bambini e le bambine tenderanno a farne una seconda “da destra a sinistra”, e poi di nuovo “da sinistra a destra”. È una variante perfettamente sensata.

Perché funziona? Dopo una passata completa, la cannuccia più lunga sarà finita completamente a destra (perché, se è più lunga, viene spostata verso destra ogni volta che la confronti con una più corta). Ripetendo, anche le altre trovano la loro posizione corretta.

Un'alternativa altrettanto naturale è la seguente strategia. In questo caso, una passata consiste nel considerare una certa posizione e nel confrontare (ed eventualmente scambiare) la

cannuccia in quella posizione con le cannucce nelle posizioni seguenti (potenzialmente, durante la passata, la cannuccia nella posizione fissata cambierà per via degli eventuali scambi, ma si confronta comunque sempre la cannuccia in posizione fissata con le successive).

Per 5 cannucce:

- Si fa una passata fissando la prima posizione.
- Si fa una passata fissando la seconda posizione.
- Si fa una passata fissando la terza posizione.
- Si fa una passata fissando la quarta posizione.

Perché funziona? Dopo la prima passata, la cannuccia più corta sarà in prima posizione. Per cui, non serve più controllare quella posizione. Dopo la seconda passata, la seconda cannuccia più corta sarà in seconda posizione. Dopo quattro passate, tutte saranno nella posizione corretta. Non è necessario confrontare la cannuccia in quinta posizione perché non ha cannucce che la seguono.

Queste due sono solo le più comuni strategie osservate negli studi. Le alunne e gli alunni potrebbero proporre molte strategie diverse, miste, eccetera. A questo livello, l'obiettivo non è insegnare strategie migliori (ne esistono altre, più efficienti ma più complicate), ma solo far sperimentare ai ragazzi la costruzione di strategie e la necessità di misurarne la bontà secondo un criterio (ad esempio il numero di scambi effettuati).

Esplorazione e riflessione

Prima di tutto, è importante che le alunne e gli alunni trovino un modo per “formalizzare” la propria strategia, cioè descriverla (a parole), anche con l'idea di comunicare la strategia ad altri, di modo che essi la possano seguire fedelmente e applicare in casi diversi (cioè con cannucce di lunghezza diversa o, più avanzato, con un numero di cannucce diverso). Spesso, nei capitoli precedenti, le istruzioni erano già formalizzate nei Manuali delle istruzioni: qui si chiede alle alunne e agli alunni di iniziare a proporre le loro istruzioni.

Inoltre, è possibile aprire a riflessioni informatiche più profonde:

- come capire se la strategia funziona sempre?
- come capire se una strategia è meglio di un'altra?

A questo livello scolastico, la principale possibilità è quella di provare ciascuna strategia con cannucce di lunghezze diverse e con disposizioni iniziali diverse (le cannucce sono già ordinate, oppure ordinate ma in ordine decrescente di lunghezza, oppure quasi ordinate, oppure in ordine totalmente casuale), contando quanti confronti vanno effettuati con una certa strategia nei diversi casi.



ASPETTO A CUI FARE ATTENZIONE

NB: Nel Quaderno di Informatica (pag. 27) viene chiesto di tenere traccia del numero degli scambi, ma in realtà è più efficace tenere traccia del numero di confronti richiesti (cioè quante volte si indica una coppia di cannucce, indipendentemente dal fatto che poi questo risulti o meno in uno scambio).

Contenuto informatico

La progettazione di questa attività è ispirata alla ricerca di Yang e colleghi [2025] che mostra che le bambine e i bambini mettono in atto strategie riconducibili ad algoritmi noti dell'informatica.

Lo studio degli algoritmi di ordinamento è un argomento classico in Informatica¹, perché didatticamente possono servire da esempio per capire concetti fondamentali della disciplina che verranno esplorati più nel dettaglio nei fascicoli rivolti a classi successive della scuola primaria e secondaria di primo grado.

Da “unplugged” a “plugged”

Spesso è utile, per consolidare i concetti appresi attraverso attività senza computer, metterli in pratica usando il computer.

Alle pagine <https://q3.hubscuola.it/y66> e <https://q3.hubscuola.it/j8w> le alunne e gli alunni potranno sperimentare digitalmente quanto appreso durante le due attività proposte. I simulatori, con la loro natura multimodale e interattiva, possono costituire un supporto a una pratica didattica inclusiva. Dal punto di vista metodologico, potrebbero essere introdotti dopo aver concluso le rispettive attività senza computer, incoraggiando alunne e alunni a mettere in relazione significativa le due esperienze.

Soprattutto per quanto riguarda l'attività con le cannuce, il simulatore costituisce anche un valido strumento per scoprire nuove strategie, potendole testare più velocemente (specialmente quando non è disponibile un adulto).

RELAZIONE CON L'INFORMATICA NELLE INDICAZIONI NAZIONALI

Relativamente alle Indicazioni Nazionali 2025, queste attività contribuiscono allo sviluppo delle seguenti competenze.

Competenze (alla fine della scuola primaria) nella disciplina Matematica:

- *Descrivere procedure per lo svolgimento di compiti pratici mediante algoritmi.*
Questa competenza era già coperta dalle attività della sezione “Algoritmi quotidiani”. Nelle attività proposte in questa sezione i compiti non sono più quelli della vita quotidiana, ma sono problemi che richiamano esplicitamente elaborazioni dell'informazione classiche come trovare il massimo o ordinare elementi.
- *Scrivere e comprendere semplici programmi, espressi in elementari linguaggi di programmazione a scopo didattico, e valutarne l'adeguatezza rispetto al compito che si vuole automatizzare.*
Sebbene queste attività siano unplugged (senza computer), i linguaggi utilizzati mirano comunque ad essere rigorosi (*Manuale delle istruzioni* limitato e definito). Per questo sono una buona preparazione a questa competenza, poiché sono orientate alla progettazione e scrittura di programmi, e alla loro valutazione (controllo di correttezza, debug, ...).

Tenendo presenti le stesse specifiche appena fatte per le competenze, le attività proposte contribuiscono ai seguenti obiettivi.

Obiettivi (alla fine della terza primaria) nella disciplina Matematica

- *Descrivere a parole attività della vita quotidiana tramite sequenze di passi precisi e non ambigui e saperle eseguire.*

¹ La prima strategia suggerita (confrontare a coppie adiacenti) è simile al Bubble sort, nella variante Shaker sort se si fa una passata “in avanti” e una “indietro”. La seconda strategia è simile al Selection sort, nella variante “con scambi immediati” (Exchange/Interchange sort), dovuta al fatto che il confronto e l'eventuale scambio avvengono unitamente nell'unica istruzione disponibile. Esistono algoritmi più efficienti, ma più complicati da esprimere in questo contesto.

- *Scrivere semplici programmi e verificare, mediante la loro esecuzione, se svolgono il compito previsto ed eventualmente correggerli.*

In questo caso, l'esecutore è umano.

Infine, lavorano sulle seguenti conoscenze (inserite nelle Indicazioni Nazionali 2025 a titolo di esempio, non prescrittive)

- *concetto di algoritmo e sua esecuzione rigorosa*
- *controllo di correttezza dei programmi*

VALORE COSTRUZIONISTA

Queste attività funzionano particolarmente bene se:

- si lascia spazio a tentativi, errori e correzioni;
- si valorizza la discussione su come si è arrivati alla soluzione;
- si confrontano strategie diverse.

Le attività unplugged (come quelle di questo capitolo) possono essere efficaci perché materializzano idee anche “profonde” dell'informatica, senza semplificarle troppo: le alunne e gli alunni le incontrano attraverso l'esperienza, non attraverso definizioni astratte [Bell e Lodi, 2019]. In molti casi si lavora con corpo e oggetti (muoversi, manipolare, confrontare), e questo rende naturale parlare di procedure, confronti, eliminazioni e ripetizioni anche prima che i bambini abbiano strumenti formali per farlo.

La lezione diventa uno spazio di gioco-esplorazione: gli alunni provano, sbagliano, correggono e imparano a dare un senso ai risultati dei loro tentativi. Il ruolo dell'insegnante è decisivo: non serve “spiegare tutto subito”, ma guidare con domande mirate, esempi ben scelti e piccoli vincoli che rendono l'esplorazione produttiva. In pratica: si lascia libertà, ma con una traccia in mente (un percorso possibile) e con una buona impalcatura di domande che aiuta la classe a costruire significato passo dopo passo.

Inoltre, come abbiamo già visto, in informatica spesso il problema da risolvere (l'obiettivo da raggiungere) è nuovo: non si tratta di applicare una procedura nota, ma, al contrario, esattamente di inventarne una nuova (almeno per chi non la conosce già) che risolve quel problema (o, idealmente, quella classe di problemi). Si tratta dunque di un importante modo di allenare le competenze, che richiedono appunto l'applicazione di conoscenze e abilità in contesti inediti e realistici.

SUGGERIMENTI PER ULTERIORI ATTIVITÀ E RIFERIMENTI

La prima attività è una variante “leggera” di un'idea classica dell'unplugged: trovare il massimo con una bilancia a due piatti. L'attività è descritta:

- nella versione classica di CS Unplugged <https://q3.hubscuola.it/tmk> (Attività 7, pagina 93).
- in una forma più strutturata, che include la scrittura di un algoritmo, in Michael Lodi, Renzo Davoli, Rebecca Montanari, Simone Martini [2020]. Informatica senza e con computer nella Scuola Primaria. Sottosezione “La ricerca del massimo con le bilance”. Capitolo del volume “Coding e oltre: Informatica nella scuola”, a cura di Enrico Nardelli, edito da Lisciani. Una versione libera del capitolo è disponibile a <https://q3.hubscuola.it/4aa>

La seconda attività è ispirata agli esperimenti descritti in Huiwen Alex Yang, Bill D. Thompson, Celeste Kidd. Children spontaneously discover efficient solutions to a difficult sorting task. *Nature Human Behaviour* (2025). <https://q3.hubscuola.it/iu4>

Una sfida Bebras (vedi fine del Capitolo 3) legata alle strategie di risoluzione dei problemi potrebbe essere questa:

<https://q3.hubscuola.it/pci> (per leggere la soluzione e la spiegazione informatica, seleziona “Mode: spiega”)

I principi alla base delle attività unplugged e il loro rapporto con gli approcci costruzionisti sono esplorati nel seguente articolo (in inglese): Tim Bell e Michael Lodi (2019). Constructing computational thinking without using computers. *Constructivist Foundations* 14(3): 342–351. Una versione liberamente scaricabile si trova qui: <https://q3.hubscuola.it/ztn>

CAPITOLO 7: CODE.ORG

Dopo aver esplorato concetti fondamentali dell'Informatica (come gli **algoritmi** e la **rappresentazione delle informazioni**) e aver compreso come dare **istruzioni** a un **esecutore**, è il momento di passare alla **programmazione** vera e propria usando il computer o il tablet.

In questa sezione ti guideremo passo passo nell'utilizzo dei corsi proposti da Code.org, un'associazione non profit che da anni promuove l'accesso universale all'informatica, con risorse gratuite e attività pensate specificamente per bambini e bambine.

Per la scuola primaria, Code.org propone percorsi per insegnare la programmazione usando un linguaggio visuale a blocchi, cioè in cui le istruzioni sono blocchetti grafici da trascinare e collegare come pezzi di LEGO®.

Grazie a personaggi familiari (come Angry Birds, Plants vs Zombies, Frozen, Star Wars ecc.), l'attività risulta motivante e vicina all'immaginario di bambini e bambine. Le sfide proposte sono strutturate in modo progressivo (partono dall'imparare a trascinare i blocchi e progrediscono in maniera molto graduale) e forniscono un alto livello di scaffolding (supporto guidato). Ad ogni livello il sistema dà feedback immediato, suggerimenti e obiettivi chiari da raggiungere. Anche se può sembrare un approccio limitato in termini di creatività, questa modalità basata su puzzle è estremamente utile per i principianti, perché riduce la paura di sbagliare e accompagna passo dopo passo nella scoperta dei concetti fondamentali. I bambini e le bambine imparano così ad apprezzare gli errori come parte del processo (**debugging**) e a sviluppare strategie di problem solving in un contesto giocoso e controllato.

TRAGUARDI E OBIETTIVI GENERALI

- Orientarsi nell'ambiente di programmazione Code.org Studio (interfaccia, comandi base, esecuzione e reset del programma).
- Creare programmi in sequenza, disponendo le istruzioni in un ordine logico per muovere un personaggio sullo schermo.
- Riconoscere il ruolo della ripetizione per eseguire più volte una stessa istruzione o sequenza di istruzioni in modo efficiente.
- Creare disegni e percorsi geometrici, sviluppando precisione nell'uso delle sequenze e delle ripetizioni e collegando la programmazione ai concetti spaziali.
- Sperimentare la strategia di prova ed errore, interpretando i feedback dell'ambiente per correggere gli errori (debugging).

PREREQUISITI

Le attività di questo capitolo risultano più efficaci se studenti e studentesse hanno già avuto qualche esperienza di base con i concetti informatici introdotti nei capitoli precedenti. In particolare è consigliabile, ma non obbligatorio, che abbiano lavorato sui movimenti su griglia (Capitolo 4).

Dal punto di vista strumentale, avere un po' di dimestichezza con il mouse o il touch (cliccare, trascinare, rilasciare) è un vantaggio. Tuttavia, i primi livelli dei corsi (in particolare il Corso A) sono pensati proprio per esercitare queste abilità di base: se per alcuni è la prima esperienza al computer o tablet, Code.org può diventare un'ottima occasione guidata per imparare a usare il mouse o il touch mentre si lavora sui concetti informatici.

PREPARAZIONE

Programma il Futuro

In Italia, le attività di Code.org sono state diffuse grazie al progetto del Laboratorio CINI Informatica e Scuola, patrocinato dal MIM, chiamato *Programma il Futuro* (PiF).

Il progetto fornisce da oltre 10 anni un ampio supporto agli insegnanti italiani nell'introduzione dell'Informatica nella scuola. Ti consigliamo di visitare il sito

<https://q3.hubscuola.it/rwf>.

Iscrizione dell'insegnante e della classe a Code.org

Il modo più semplice per iscriversi a Code.org è seguire le istruzioni sul sito di *Programma il Futuro*: <https://q3.hubscuola.it/czb>. In questo modo ti iscriverai contemporaneamente sia a PiF sia a Code.org.

Una volta completato questo passo, confermando la registrazione a *Programma il Futuro* cliccando sul link presente nella mail ricevuta all'indirizzo e-mail usato per la registrazione, è possibile accedere a Code Studio (<https://q3.hubscuola.it/83g>) usando lo stesso indirizzo e-mail e la stessa password.

Successivamente si può procedere con la creazione di una classe. Per creare una classe, bisogna accedere alla sezione Controllo (che è comunque la pagina predefinita al login su studio.code.org), dedicata alla gestione delle classi esistenti e alla creazione di nuove.

Si può indicare anche il livello della classe e il corso seguito (scelte non obbligatorie, ma consigliate per agevolare studenti e studentesse).

Suggeriamo di compiere le seguenti scelte:

- Tipi di login → Immagine password.
- Nome della classe → attribuire un nome significativo.
- Grado → indicare la classe (es. 3 per la terza primaria).
- Assegna curriculum → Fondamenti di informatica - Corso A - Versione 2021 (o il corso da cui vuoi partire: vedi più avanti).
- Impostazioni avanzate:
 - Esercizi Supplementari (Lesson Extras) → Sì
 - Programmazione in coppia → Sì. Abilitare questa opzione consente agli studenti di lavorare in due o più sullo stesso dispositivo, mantenendo aggiornati i progressi dei profili di tutti.

Salvando queste informazioni (che possono comunque essere tutte successivamente modificate), la classe viene identificata da un codice di 6 caratteri, ad esempio ABCDEF.

Una volta creata la classe, devono essere aggiunti studenti e studentesse. L'unica informazione obbligatoria è un nome/nickname univoco per ciascuno. Consigliamo di non inserire dati personali (es. usare un numero e/o un nomignolo). Una volta che la classe sarà completa¹, si potrà consegnare a ciascuno un foglio autogenerato con le istruzioni e la loro immagine-password per accedere alla classe.

Assicurati che tutti abbiate il testo visualizzato in italiano. Se così non fosse, scorri la pagina fino in fondo e seleziona "Italiano" dal menu a tendina delle lingue.

Dalla sezione Controllo, puoi monitorare le tue classi e i progressi di studenti e studentesse.

¹ Potrebbe capitare che venga richiesto di accettare le condizioni relative alla privacy e all'uso dei cookie.

Accesso di studenti e studentesse a Code.org

Tutte le lezioni al computer o sul tablet iniziano con gli studenti e le studentesse che effettuano l'accesso alla classe su Code.org, raggiungendo il sito studio.code.org, inserendo il codice della classe (es. ABCDEF) nella parte destra ("Inserisci il codice di 6 lettere della tua classe") e poi premendo su "Go".

E se non voglio creare nessun account?

Puoi accedere direttamente ai corsi, ma ovviamente i progressi non verranno salvati e non sarà possibile creare una classe.

Link diretto alle attività interattive del Corso A:

<https://q3.hubscuola.it/9xj>

Link diretto alle attività interattive del Corso B:

<https://q3.hubscuola.it/2mv>

Link diretto alle attività interattive del Corso C:

<https://q3.hubscuola.it/7ja>

ATTIVITÀ 1: Introduzione a Code.org

→ Vedi Quaderno di Informatica pp. 28-29

I corsi di "Introduzione all'Informatica" di Code.org forniscono ambienti molto guidati; una volta entrati, studenti e studentesse impareranno velocemente a muoversi all'interno di essi. Specialmente per chi non ha dimestichezza con l'uso del mouse (o, meno probabile, del tablet) per trascinare i blocchi, ti consigliamo di iniziare dal Corso A (2021).

Come insegnante, puoi guardare qui la panoramica sulle lezioni del Corso A:

<https://q3.hubscuola.it/v59>

Nota, ad esempio, che:

- Alcune lezioni (1, 3, 7, 11) sono unplugged (contraddistinte dal pulsante "Attività tradizionale")
- Per ogni lezione, puoi
 - Leggere il piano di lavoro di ciascuna lezione: contiene una panoramica, traguardi, obiettivi, consigli per lo svolgimento, una pianificazione precisa di ogni fase della lezione;
 - Decidere se mostrarla o nascondere ai tuoi studenti.

La Lezione 2 (la prima lezione "tecnologica") aiuta a prendere confidenza con il trascinamento dei blocchi.

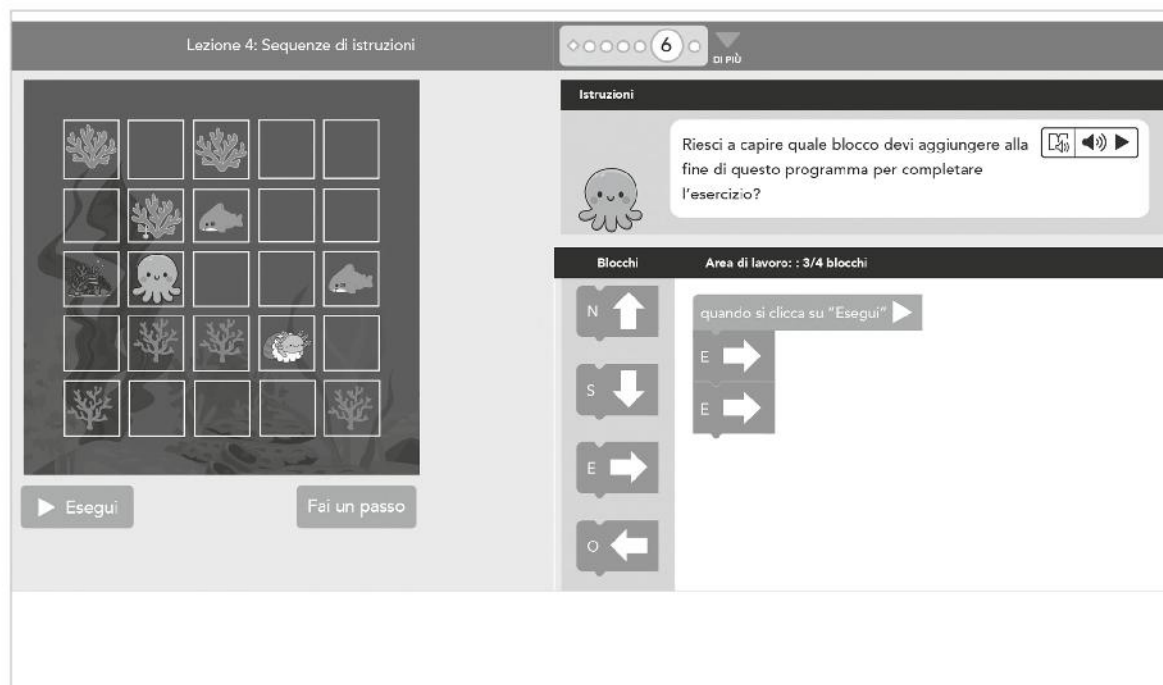
La Lezione 4 (seconda lezione "tecnologica") è la prima in cui ci si trova nell'ambiente del "labirinto". Ci sono dei livelli da superare, come in un normale videogioco, ma studenti e studentesse dovranno programmare un personaggio affinché raggiunga i suoi obiettivi. Fai loro sperimentare alcuni livelli (almeno fino al Livello 4 compreso) e, nel frattempo, comincia a far osservare loro le varie componenti dell'interfaccia che si trovano davanti.



SUL QUADERNO DI INFORMATICA

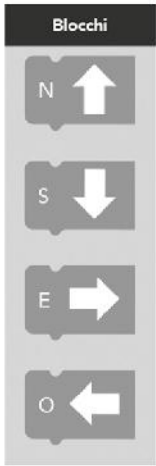
Mentre riflettete sulla funzione delle varie parti della schermata, potete tenere aperte le pagg. 28 e 29 del Quaderno di Informatica.

La schermata è sempre di questo tipo:



Ecco la spiegazione dei singoli elementi:

	Area di gioco: dove si muovono i personaggi del gioco.
	Istruzioni: quello che il personaggio deve fare per superare il livello. Si possono anche ascoltare le istruzioni premendo

	Blocchi: sono le istruzioni che il computer capisce e che puoi usare (anche più volte) per indicare al personaggio cosa fare per raggiungere l'obiettivo. Nelle attività unplugged dei capitoli precedenti abbiamo chiamato questo elenco Manuale delle istruzioni.
	Area di lavoro: qui c'è il programma, cioè la sequenza di istruzioni che vuoi dare al personaggio. I blocchi devono essere attaccati tra loro come mattoncini LEGO®. I numeri indicano rispettivamente il numero di blocchi attualmente usati (cioè quelli presenti nell'area di lavoro) e il numero di blocchi sufficienti a risolvere il puzzle e superare il livello. In alcuni casi, il numero di blocchi necessari per risolvere il puzzle è inferiore al numero di blocchi sufficienti.
	Pulsante Esegui: premilo per eseguire il programma nell'area di lavoro e vedere l'effetto che fa nell'area di gioco!
	Pulsante "Fai un passo": premilo per eseguire il programma nell'area di lavoro un'istruzione alla volta. Usalo se il personaggio non fa quello che ti aspetti!
	I livelli di questa lezione: quando i pallini saranno tutti verdi, passa alla prossima lezione.

Queste considerazioni valgono ugualmente anche per il Corso B (2021).

<https://q3.hubscuola.it/alg>

Il Corso B è allineato al Corso A, ma si distingue per attività unplugged più approfondite e una selezione più varia di esercizi al computer².

² <https://q3.hubscuola.it/l1x>
<https://q3.hubscuola.it/713>
<https://q3.hubscuola.it/d31>

Al momento della scrittura di questo manuale (dicembre 2025) i corsi del 2021 sono quelli più recenti tradotti in italiano. Da qui in poi, quando ci riferiamo ai Corsi A, B, C, intendiamo sempre la versione del 2021 in italiano.

ATTIVITÀ 2: Labirinti (Corsi A e B)

→ Vedi Quaderno di Informatica pp. 30-31

Le lezioni:

- 4, 5, 6, 8, 9 del Corso A
- 3, 4, 5, 7, 8 del Corso B

sono tutte collocate nell'ambiente "Labirinti".

In questo tipo di livelli, ci sarà un personaggio (spesso dei videogiochi o dei film conosciuti) che si muove su una griglia/scacchiera (analogamente a quanto già visto nel Capitolo 4 - Movimenti sulla griglia) e deve

- raggiungere altri personaggi (spesso nemici);
- raccogliere oggetti.

Chi programma deve immaginare di guardare la scena dall'alto e quindi potrà indicare al personaggio come muoversi seguendo i punti cardinali (Nord, Sud, Ovest, Est) per raggiungere il suo obiettivo. Si tratta quindi di movimenti descritti rispetto a un sistema di riferimento assoluto.



SUL QUADERNO DI INFORMATICA

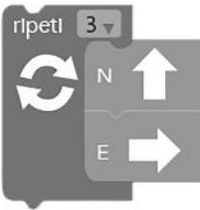
Una guida sintetica sull'effetto di ciascun blocco (istruzione) che i bambini possono incontrare nei livelli del Labirinto è disponibile alle pagine 30 e 31 del Quaderno di Informatica. Qui sotto ne riportiamo una versione estesa:

	Il personaggio si muove in alto (verso Nord) di una casella.
	Il personaggio si muove in basso (verso Sud) di una casella.
	Il personaggio si muove a destra (verso Est) di una casella.
	Il personaggio si muove a sinistra (verso Ovest) di una casella.

Oltre a muoversi i personaggi possono raccogliere oggetti:

 	Se si trova su una casella che contiene l'oggetto raffigurato, il personaggio ne raccoglie uno.
---	---

Per evitare di usare tante volte lo stesso blocco, è possibile usare il blocco Ripeti:

	Ripeti fa eseguire più volte quello che c'è dentro. Dal menu a tendina si sceglie il numero di volte.
	Ripete 3 volte l'istruzione "Spostati di una casella verso Nord". È come fare: 
	Ripete 3 volte le due istruzioni "Spostati di una casella verso Nord" e poi "Spostati di una casella verso Est". È come fare: 



SUL QUADERNO DI INFORMATICA

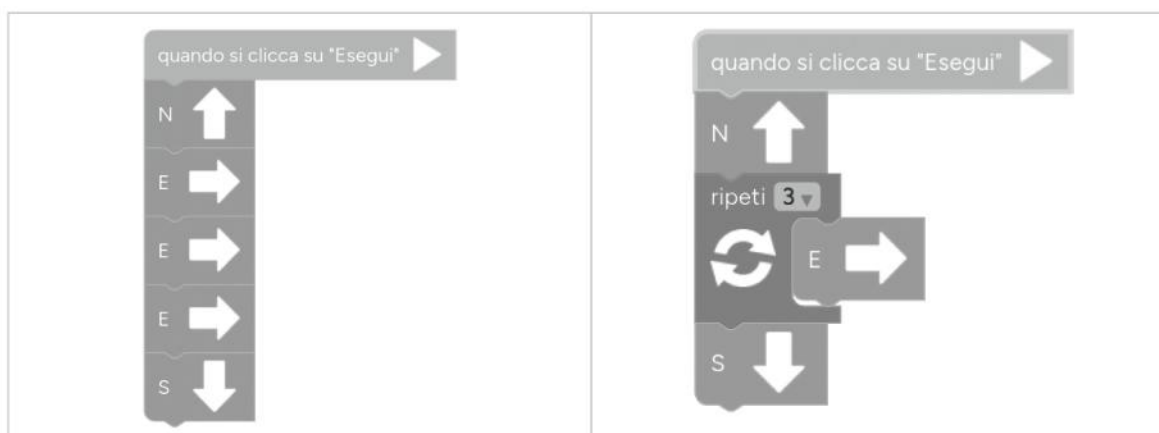
Il viaggio del polpo

Se non puoi lavorare con computer o tablet, o se vuoi assegnare un compito unplugged, puoi usare la scheda IL VIAGGIO DEL POLPO a pagina 31 del Quaderno di Informatica.

Se dopo che studenti e studentesse l'hanno svolto sul libro/quaderno vuoi farli provare sul computer, si tratta di un livello non presente nei Corsi A o B, ma accessibile a questo link: <https://q3.hubscuola.it/6ge>

Su Code.org l'ambientazione è diversa, ma le soluzioni restano le stesse.

Soluzioni (NB: il blocco “quando si clicca su “Esegui”” non è presente né conteggiato nel Quaderno di Informatica, mentre è conteggiato su Code.org).



ATTIVITÀ 3: Artista (Corsi A e B)

→ Vedi Quaderno di Informatica pp. 32-33

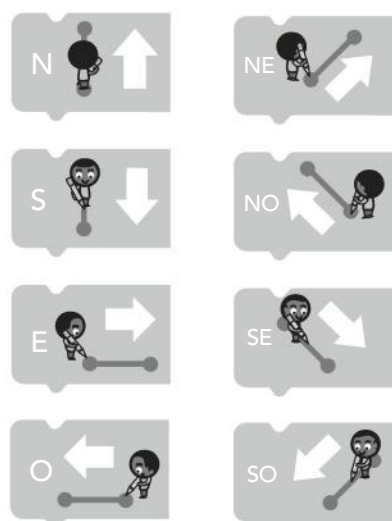
La Lezione 10 del Corso A e la Lezione 9 del Corso B si svolgono nell'ambiente dell'Artista. In questo tipo di livelli, si controlla un artista che, muovendosi sul foglio (esprimendo i movimenti sempre rispetto a un sistema di riferimento assoluto con i punti cardinali), potrà realizzare disegni geometrici.



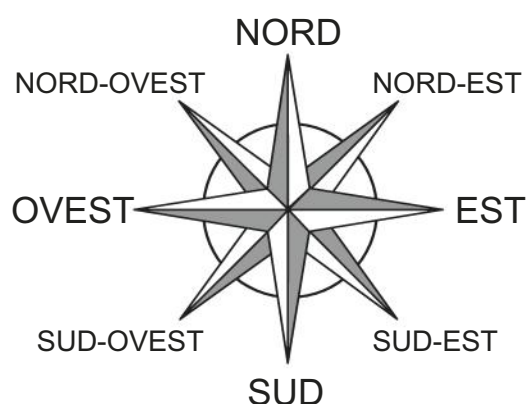
SUL QUADERNO DI INFORMATICA

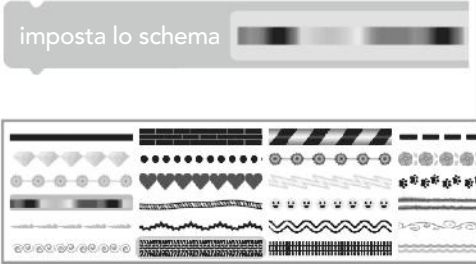
Una guida sintetica sull'effetto di ciascun blocco (istruzione) che i bambini possono incontrare nei livelli dell'artista è disponibile a pagina 32 del Quaderno di Informatica.

Di seguito ne riportiamo una versione estesa.



L'artista si muove e disegna un segmento nella direzione indicata, anche in diagonale



	L'artista si muove come nei blocchi sopra, ma senza lasciare traccia!
	Da questa istruzione in poi, l'artista disegna usando lo schema scelto fra quelli proposti.
	Da questa istruzione in poi, l'artista disegna usando una penna del colore scelto.
	L'artista stampa, nel punto in cui si trova, l'adesivo (come un timbro).

Si può cambiare la velocità con cui l'artista disegna.

L'artista disegna più lentamente se la freccia è vicina alla tartaruga, e più velocemente se è vicina alla lepre:





SUL QUADERNO DI INFORMATICA

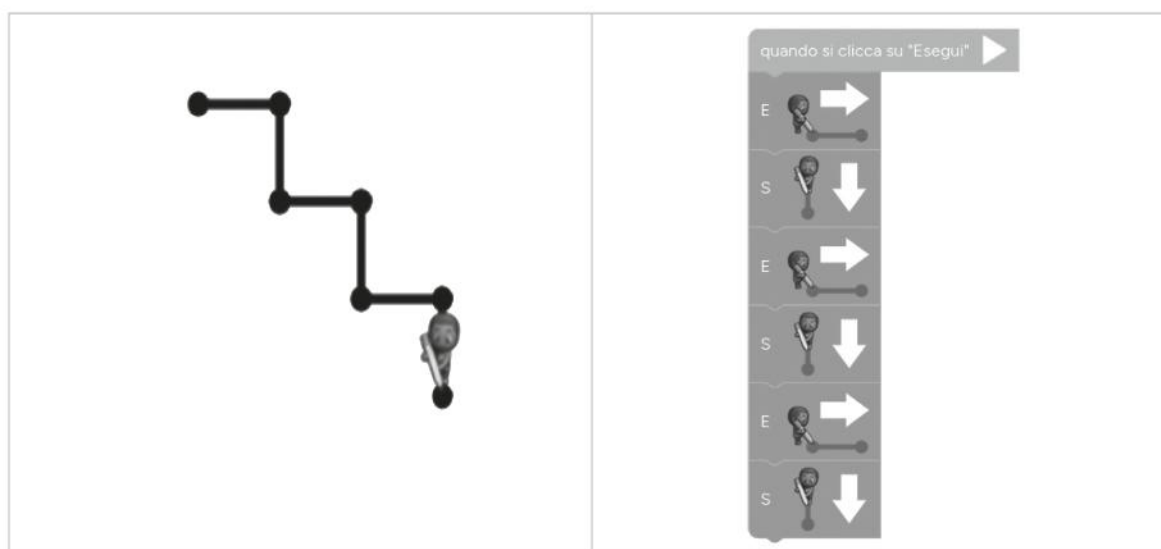
Se non puoi lavorare con computer o tablet, o se vuoi assegnare un compito unplugged, puoi assegnare gli esercizi a pagina 33 del Quaderno di Informatica.

Se dopo che studenti e studentesse li hanno svolti sul libro/quaderno vuoi farli provare sul computer, puoi farli lavorare sul LABORATORIO ARTISTA PRE-SCOLARE, disponibile al link: <https://q3.hubscuola.it/zdl>

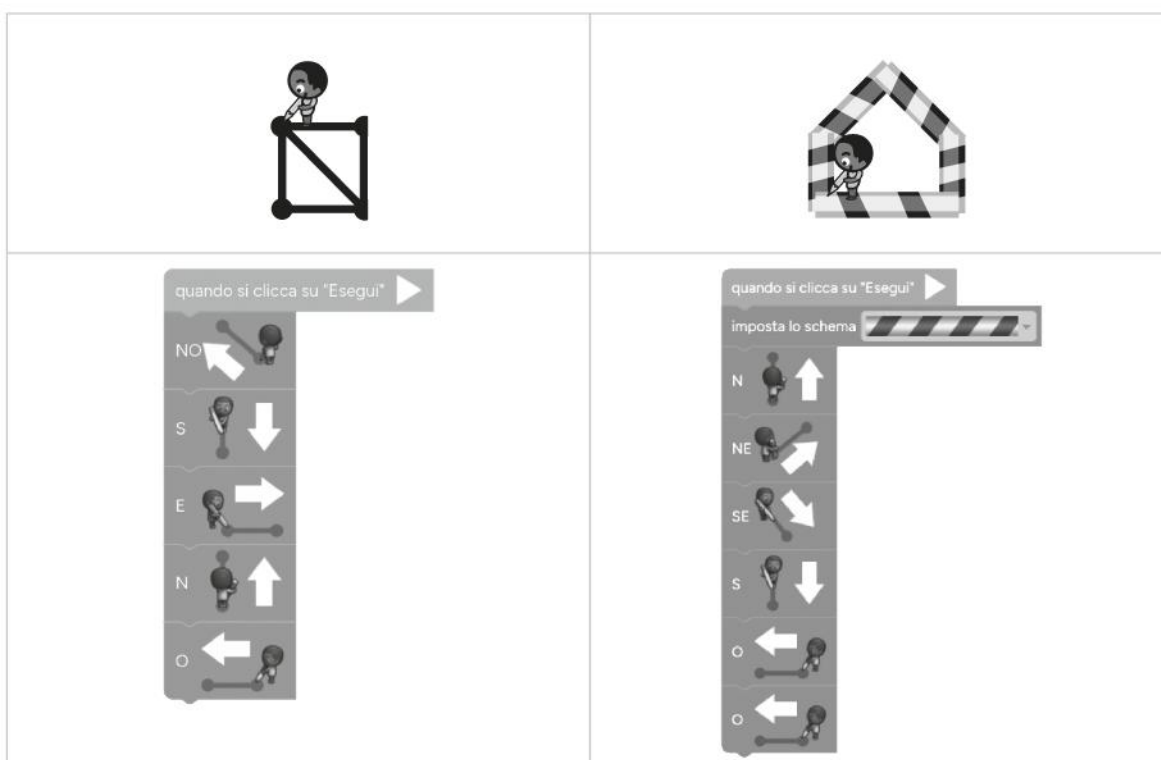
Qui possono realizzare liberamente i disegni che vogliono, senza un compito preassegnato. Possono anche condividere le loro opere tramite link.

Soluzioni alle attività di pagina 33

Con il primo programma, l'artista realizza una scaletta dall'alto in basso:



Per realizzare i disegni indicati, l'artista può seguire queste istruzioni:



Se qualcuno non sa cosa disegnare, ecco altri due suggerimenti che puoi dare loro (il secondo è il più sfidante).

	
quando si clicca su "Esegui" ▶ - N - NE - SE - S - O - O	quando si clicca su "Esegui" ▶ ripeti 3 salta O imposta lo schema - E - NE - E - SE - E - N - O - SO - O - NO - O - S

ATTIVITÀ 3: Labirinti (Corsi C)

→ Vedi Quaderno di Informatica pp. 34-35

Le Lezioni 3, 4, 5, 8, 9 sono tutte collocate nell'ambiente "Labirinti". L'ambiente è del tutto analogo a quelli dei Corsi A e B, ma, da questo corso in poi, i movimenti del personaggio saranno descritti rispetto a un sistema di riferimento soggettivo centrato sul personaggio, così come abbiamo visto in tutti gli esercizi unplugged (vedere Capitolo 4 - Movimenti sulla griglia).

Quindi, in questa attività, chi programma vive la scena dal punto di vista del personaggio: quando c'è un'istruzione di movimento, è sempre "rispetto a dove sta guardando il personaggio". Se dici al personaggio "vai avanti", lui cammina nella direzione in cui è rivolto il suo sguardo.



SUL QUADERNO DI INFORMATICA

Se non puoi lavorare con computer o tablet, o se vuoi assegnare un compito unplugged, puoi usare la scheda a pagina 34-35 del Quaderno di Informatica.

Se dopo che studenti e studentesse l'hanno svolto sul libro/quaderno vuoi farli provare sul computer, si tratta di un livello non presente nel Corso C, ma accessibile a questo link: <https://q3.hubscuola.it/bex>

Su Code.org l'ambientazione è diversa, ma le soluzioni restano le stesse.

Soluzioni (NB: il blocco "quando si clicca su "Esegui"" non è presente né conteggiato nel Quaderno di Informatica, mentre è conteggiato su Code.org).

<pre> quando si clicca su "Esegui" vai avanti gira a destra ↻ vai avanti vai avanti vai avanti vai avanti gira a destra ↻ vai avanti </pre>	<pre> quando si clicca su "Esegui" vai avanti gira a destra ↻ ripeti 4 volte fai vai avanti gira a destra ↻ vai avanti </pre>
---	---

ALTRE ATTIVITÀ

Eventi (Corsi A, B, C)

Nella Lezione 12 del Corso A e del Corso B, e nelle lezioni 12, 13 e 16 del Corso C è possibile costruire videogiochi, storie e animazioni facendo parlare personaggi, facendoli muovere e facendoli reagire ad eventi (es. click del mouse, pressione dei tasti freccia sulla tastiera).

Lascia che studenti e studentesse sperimentino con questi livelli. L'introduzione alla creazione di giochi, storie, animazioni è la parte centrale di Scratch, che verrà introdotto nel prossimo capitolo

Artista (Corso C)

Nel Corso C, anche nei livelli dedicati all'artista i movimenti vengono espressi rispetto a un sistema di riferimento soggettivo, centrato sull'Artista, ed è presente anche la possibilità di orientarlo di un certo angolo. Puoi lasciare che i tuoi studenti sperimentino intuitivamente e poi, al momento opportuno, collegare quanto emerso dalle loro esplorazioni ai concetti geometrici più formali.

L'Ora dell'InformaticA (l'Ora del Codice)

Code.org è diventata famosa per l'iniziativa "L'ora del Codice", dal 2025 rinominata "L'ora dell'InformaticA" (per maggiormente rimarcare da un lato l'aspetto disciplinare, e dall'altro l'introduzione di elementi di Intelligenza Artificiale). Si tratta di un'iniziativa in cui, più o meno nello stesso periodo (es. la "Settimana Internazionale dell'Insegnamento dell'Informatica", che solitamente cade a dicembre - <https://q3.hubscuola.it/46g>), le bambine e i bambini di tutto il mondo svolgono almeno un'ora di introduzione alla programmazione.

Puoi consultare le attività proposte sul sito:

<https://q3.hubscuola.it/b0n>

MODALITÀ DI LAVORO

L'approccio guidato dalla piattaforma dovrebbe permettervi di far sì che, una volta avviata l'attività, studenti e studentesse procedano in modo autonomo e ciascuno al proprio ritmo. Tuttavia, è importante facilitare e vigilare costantemente sul processo.

Debugging

Le attività proposte in questo capitolo della guida offrono un contesto autentico per sperimentare la pratica del debugging. Infatti, se escludiamo le attività con i robot educativi, questa è probabilmente la prima volta che gli studenti hanno a che fare realmente con un esecutore automatico, "meccanico".

Come abbiamo visto, in informatica, più che applicare una procedura già nota, si tratta spesso di progettare una che risolva il problema in questione, ma non sia già conosciuta e che, idealmente, possa funzionare anche per altri casi simili appartenenti alla stessa classe di problemi. Sarà quindi normale procedere, almeno in parte, con prove ed errori. Se da un lato all'inizio può essere frustrante vedere che il programma non fa ciò che ci si aspetta, dall'altro – una volta convinti studenti e studentesse che il computer esegue sempre e solo la sequenza di istruzioni che gli vengono date, e in modo sempre "prevedibile" e "ripetibile" – si può accettare più serenamente che l'errore è dovuto a un programma che va rivisto e migliorato perché non aderente agli obiettivi.

Per abituare studenti e studentesse a riconoscere le cause degli errori e correggerli, può essere utile aiutarli a strutturare le fasi di debug, ad esempio proponendo di seguire le fasi spiegate nel box che segue.

Fasi di debugging

1. Assicurarsi di aver capito qual è l'obiettivo del protagonista.
2. Eseguire il programma: cosa c'è di sbagliato nel comportamento del protagonista?
3. Cercare di determinare quali blocchi sono responsabili del comportamento sbagliato (per capirlo meglio può essere utile usare "Fai un passo" per eseguire un'istruzione alla volta, vedendo che effetto ha sul programma; oppure far disegnare l'Artista più lentamente).

4. Modificare il programma (aggiungere o togliere istruzioni, o modificare i numeri presenti in esse) e poi rieseguirlo.
5. Se ancora non funziona, valutare se con le modifiche fatte ci si sta avvicinando all'obiettivo oppure se è meglio annullarle.
6. Ritornare al passo 3 fino a che non si è raggiunto l'obiettivo.

Programmazione in coppia

Un altro modo di fare debugging è raccontare a un compagno o una compagna che cosa fa il proprio programma. Spesso il solo fatto di spiegare ad alta voce le istruzioni del proprio programma fa capire a chi l'ha creato dove ha sbagliato. Nell'industria informatica, questa pratica è stata istituzionalizzata come metodologia chiamata “pair programming”, cioè “programmazione in coppia” - due programmatori programmano insieme su un solo computer. È una metodologia estremamente utile anche a scuola.

Nella programmazione in coppia, pensiamo la coppia come a un equipaggio da rally: da una parte c'è il **pilota**, dall'altra il **navigatore**. Come in un vero rally, il pilota ha in mano il “volante”, cioè il mouse: il pilota materialmente guida il programma, clicca, trascina i blocchi, scrive il codice. Il suo obiettivo è arrivare a un programma funzionante, proprio come il pilota vuole portare l'auto al traguardo. Mentre lavora, però, non lo fa in silenzio: il pilota racconta ad alta voce al navigatore che cosa sta facendo, passo dopo passo, spiegando perché sceglie un certo blocco o lo mette in un certo punto. Il compagno o la compagna con il ruolo di navigatore potrà dare dei suggerimenti ma alla fine sarà il pilota a prendere le decisioni finali: ha il volante/mouse in mano.

Il navigatore, invece, è come il copilota nel rally: osserva con attenzione ciò che fa il pilota e ascolta ciò che dice, ma non tiene il volante. Il suo compito è guardare bene la “strada” che hanno davanti, in questo caso il puzzle di programmazione, e aiutare il pilota a non perdersi. Può notare errori, proporre correzioni, suggerire strade alternative: “E se invece provassimo a mettere questo blocco qui?”, “Secondo me così il personaggio non arriverà dove vogliamo”. Mentre il pilota si concentra sui dettagli, il navigatore mantiene la visione d'insieme: verifica se il programma sta davvero procedendo nella direzione giusta per risolvere il problema.

Ogni tanto si scambiano i ruoli: chi guidava diventa il navigatore e chi faceva il navigatore passa al volante. All'inizio sarà l'insegnante a guidare questo scambio e a ricordare i compiti di ciascuno, ma l'obiettivo è che studenti e studentesse imparino pian piano a collaborare in modo autonomo.

Come insegnante dovrai fare attenzione nel formare (e, se necessario, separare) le coppie e, soprattutto, nel far rispettare i ruoli: un pilota che non considera il navigatore o un navigatore prepotente sono da evitare!

In questo video viene spiegata la metodologia: <https://q3.hubscuola.it/32w>

Su Code.org è possibile cliccare sulla casella accanto alla scritta “Sto lavorando in coppia” che appare una volta effettuato il login, quindi cliccare prima sul nome del compagno o della compagna con cui si sta lavorando e poi sul pulsante “Aggiungi partner”, per essere sicuri che vengano tracciati i progressi di entrambi i membri della coppia (o anche di gruppi più numerosi).

Evitare l'effetto “basta superare il livello”

La modalità con cui progrediscono le lezioni di Code.org (piccoli puzzle da superare) dovrebbe essere congeniale perché studenti e studentesse li considereranno livelli da superare in un videogioco. Tuttavia, è importante evitare che questo generi l'idea che basti “far venire gli

esercizi” (senza aver capito perché la soluzione proposta è corretta, oppure proponendo soluzioni molto lunghe e ripetitive da scrivere), magari addirittura gareggiando per chi li completa prima. Di seguito discutiamo alcune strategie per superare questo approccio ingenuo.

Esistono più programmi corretti!

Uno degli aspetti più belli dell'informatica è che lo stesso problema può avere molte soluzioni diverse, tutte valide. In Code.org, questo diventa subito evidente: programmi differenti possono portare il personaggio allo stesso obiettivo, pur utilizzando blocchi o strategie non identiche. È importante valorizzare questo aspetto: non si tratta di trovare “la” risposta giusta, ma di scoprire che esistono modi diversi per arrivare allo stesso risultato! Questa varietà permette attività ricchissime in classe:

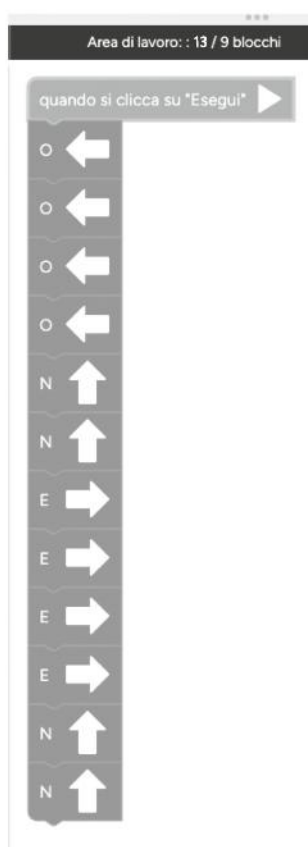
- confrontare soluzioni diverse;
- raccontare come ciascuno ha ragionato;
- osservare cosa cambia se si sposta un blocco;
- scoprire nuove idee guardando il lavoro degli altri.

In questo modo la programmazione diventa un terreno di dialogo e creatività: ogni soluzione è un punto di vista, e il confronto tra programmi aiuta a capire meglio sia il problema sia i concetti informatici in gioco.

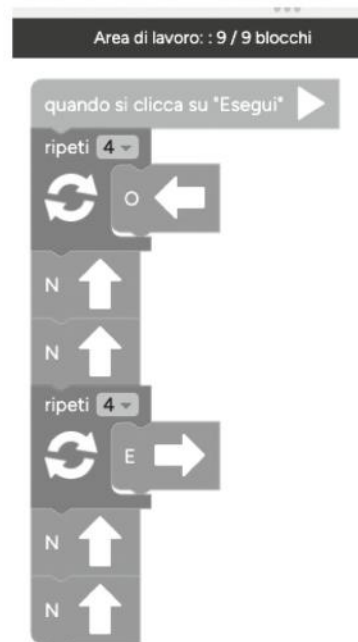
La sfida è usare meno blocchi possibile

Nell'Area di lavoro viene sempre indicato il numero di blocchi attualmente utilizzati nel programma. Solitamente, se si risolve un puzzle usando un numero elevato di blocchi, Code.org suggerisce che sarebbe stato possibile farlo con meno. È importante che, piano piano, studenti e studentesse comincino ad apprezzare programmi che fanno uso di meno blocchi, in quanto più chiari e semplici da comprendere (e anche da realizzare, ad esempio, senza dover trascinare lunghe sequenze di blocchi tutti uguali). Dunque, è importante suggerire di non superare il numero di blocchi indicato nel titolo dell'Area di lavoro.

L'esempio del blocco Ripeti è lampante: non è necessario per risolvere i semplici problemi proposti in questi corsi, ma consente di realizzare programmi decisamente più brevi (cioè con meno blocchi) e quindi più leggibili.



Ad esempio, il programma di sinistra usa 13 blocchi ma se ne poteva realizzare un altro con lo stesso effetto usando solo 9 blocchi (immagine a destra)!



A volte compare un numero rosso che indica il numero massimo di volte in cui si può usare un blocco.



Questo è un altro modo su Code.org per costringere gli studenti a utilizzare, ad esempio, il blocco Ripeti.

VALORE COSTRUZIONISTA

Da un punto di vista metodologico, in Code.org la scoperta di nuovi concetti è inizialmente molto vincolata da un elevato scaffolding, cioè un insieme di *sostegni temporanei* che aiutano bambini e bambine a procedere passo passo fino a diventare autonomi. Pian piano viene concessa loro maggiore libertà creativa [Resnick e Rusk, 2020]. Tuttavia, in ogni puzzle da risolvere viene spesso richiesto di usare nuovi blocchi lasciando a studenti e studentesse la possibilità di esplorarne il funzionamento per capire come utilizzarli per raggiungere gli obiettivi e superare i livelli.

Per poter aumentare la libertà di esplorazione anche quando si programma su Code.org è nato, in direzione costruzionista, l'approccio metodologico chiamato *Usa-Modifica-Crea*, che descriviamo di seguito.

Usa, Modifica, Crea

In questo approccio metodologico (o meglio, famiglia di approcci metodologici - poiché è stato declinato in molti modi diversi), studenti e studentesse imparano usando programmi già scritti, modificandoli e, infine, creando programmi tutti loro [Caserio et al., 2022].

Le attività sono solitamente organizzate in tre fasi. Le fasi hanno un ordine logico, ma non sono rigide: si può andare avanti o indietro più volte.

1. Fase “**Usa**”: “Provo il programma degli altri”

In questa prima fase si esegue un programma già pronto, costruito dall'insegnante o da altri, rispondendo poi a domande di riflessione.

- Il programma non è banale: deve essere un po' “curioso” o “bello da vedere” (es. un minigioco o una storia autoconclusiva) e, idealmente, contenere i nuovi concetti che si vogliono introdurre.
- Bambini e bambine eseguono il programma senza vedere le istruzioni di cui è composto: osservano semplicemente ciò che succede sullo schermo, e rispondono a domande di riflessione su ciò che hanno esperito. In questo modo studenti e studentesse si concentrano prima sul descrivere e interpretare il comportamento del programma e non sulle istruzioni.

2. Fase “**Modifica**”: “Metto mano al codice”

Nella seconda fase, studenti e studentesse possono vedere le istruzioni che costituiscono il programma. L'obiettivo è leggere le istruzioni (i blocchi), esplorarli e collegarli a ciò che accade sullo schermo quando il programma viene eseguito. L'insegnante in questa fase può proporre vari tipi di attività, quali:

- Richiedere di apportare modifiche alle istruzioni del programma, come ad esempio spostare o sostituire un blocco, o cambiare un numero dentro a un blocco. Studenti

e studentesse eseguiranno quindi di nuovo il programma osservando che cosa è cambiato nell'effetto.

- Richiedere di fare previsioni. Prima di apportare la modifica, l'insegnante invita studenti e studentesse a formulare esplicitamente un'ipotesi sull'effetto che avrà la modifica sul risultato dell'esecuzione del programma. Solo dopo che ciascuno avrà formulato la propria previsione si potrà procedere con la modifica del programma, confrontando le varie ipotesi con il risultato.
- Richiedere di apportare modifiche alle istruzioni del programma con un obiettivo dato. Viene dato un obiettivo esplicito e circoscritto, per esempio: cambiare un dettaglio grafico; variare in piccola parte il funzionamento del programma. Studenti e studentesse quindi proveranno, sbaglieranno, ritenteranno finché non raggiungono il risultato richiesto.

In questa fase si lavora molto sulla curiosità e sulla sperimentazione, orientando sempre le attività di modifica a comprendere meglio i nuovi concetti da introdurre.

3. Fase “Crea”: “Costruisco il mio programma”

Nella terza fase, studenti e studentesse sono invitati a creare un proprio programma personale, utilizzando anche i nuovi concetti appresi. La creazione può essere più o meno dirompente rispetto ai progetti delle fasi precedenti.

- Creazione come evoluzione delle modifiche. Studenti e studentesse partono dal programma proposto dall'insegnante, ma decidono di modificare in modo più profondo la grafica o il funzionamento. Il tuo ruolo di insegnante in questo caso sarà quello di controllare che, nelle loro creazioni, studenti e studentesse utilizzino i nuovi concetti introdotti.
- Creazione libera con alcuni vincoli. Studenti e studentesse possono progettare un programma anche molto diverso da quelli visti prima, ma con qualche vincolo chiaro, ad esempio: usare alcuni blocchi obbligatori; rispettare un certo comportamento generale dei personaggi.

In questo modo, anche nella creazione più libera, bambini e bambine devono comunque mettere in gioco i nuovi concetti, consolidandoli attraverso le proprie idee.

Un esempio di materiale didattico completo (che comprende sia le schede studenti che una guida passo-passo per l'insegnante) per introdurre il concetto di ciclo utilizzando l'Artista di Code.org in una classe di terza primaria è disponibile all'indirizzo:
<https://q3.hubscuola.it/efy>

ASPETTI INFORMATICI

In queste attività studenti e studentesse incontrano per la prima volta un vero linguaggio di programmazione. È fatto di blocchetti colorati, ma ha tutte le caratteristiche di un linguaggio formale: ad esempio, ha una sintassi (i blocchetti si incastrano solo in un certo modo) e una semantica (es. la sequenza di istruzioni incluse nella forma a C di un Ripeti viene eseguita il numero di volte indicato nel blocco stesso).

In queste attività studenti e studentesse non si limitano a “giocare con i blocchi”, ma incontrano alcune idee chiave di informatica.

- **Algoritmo vs programma.** I puzzle dei Labirinti e dell'Artista rendono visibile il passaggio dall'algoritmo (la sequenza di istruzioni pensata a voce o su carta) al programma (cioè l'algoritmo tradotto in un linguaggio di programmazione eseguibile dal computer). La stessa

strategia usata negli esercizi unplugged (sequenze di frecce o istruzioni) deve ora essere tradotta in blocchi, rispettando la sintassi del linguaggio.

- **Sequenza e ciclo come prime strutture di controllo.** La maggior parte dei puzzle può essere risolta con semplici sequenze; l'introduzione del blocco Ripeti permette di “compattare” programmi lunghi e ripetitivi, facendo emergere l'idea che un buon programma non è solo quello che “funziona”, ma anche quello più leggibile e manutenibile (quindi ad esempio ha meno blocchi ed è più chiaro).
- **Parametri e generalizzazione.** Nei livelli dell'Artista, trovi alcuni blocchi che richiedono di specificare dei numeri. In informatica, questi numeri sono chiamati parametri: indicano quanto deve muoversi o ruotare l'Artista. Cambiando il parametro, puoi ottenere disegni diversi: non stai più programmando “questo quadrato preciso”, ma stai programmando “un quadrato di lato di lunghezza x”, semplicemente modificando il numero. È una prima forma di generalizzazione, perché studenti e studentesse capiscono che un programma può funzionare anche quando cambiano i valori.
- **Esecutore meccanico.** Il personaggio sullo schermo si comporta come un esecutore “ottuso”: fa esattamente ciò che c'è scritto nel programma, né più né meno. Insistere su questo aspetto aiuta studenti e studentesse a costruirsi una prima immagine mentale del calcolatore come sistema deterministico che esegue alla lettera una sequenza di istruzioni.
- **Previsione, esecuzione e debugging.** In questo capitolo, nella sezione relativa alla Modalità di lavoro, il debugging è già stato presentato come pratica centrale per accogliere e affrontare gli errori in modo sereno e sistematico. In questo modo studenti e studentesse imparano che l'errore non è un fallimento, ma un'informazione preziosa sul funzionamento del proprio programma.

RELAZIONE CON L'INFORMATICA NELLE INDICAZIONI NAZIONALI

Relativamente alle Indicazioni Nazionali 2025, queste attività contribuiscono allo sviluppo delle seguenti competenze.

Competenze (alla fine della scuola primaria) nella disciplina Matematica:

- *Scrivere e comprendere semplici programmi, espressi in elementari linguaggi di programmazione a scopo didattico, e valutarne l'adeguatezza rispetto al compito che si vuole automatizzare.*

Le attività proposte contribuiscono ai seguenti obiettivi:

Obiettivi (alla fine della terza primaria) nella disciplina Matematica

- *Scrivere semplici programmi e verificare, mediante la loro esecuzione, se svolgono il compito previsto ed eventualmente correggerli.*

Obiettivi (alla fine della terza primaria) nella disciplina Tecnologia

- *Creare, selezionare e usare semplici testi o disegni, usando strumenti informatici.*

Infine, lavorano sulle seguenti conoscenze (inserite nelle Indicazioni Nazionali 2025 a titolo di esempio, non prescrittive).

- *Concetto di algoritmo e sua esecuzione rigorosa.* In particolare qui si lavora sulla creazione ed esecuzione di programmi, cioè algoritmi scritti in un linguaggio di programmazione (qui, in particolare, il linguaggio a blocchi di Code.org).
- *Strutture di controllo fondamentali di un linguaggio di programmazione e reazione agli eventi.*
- *Controllo di correttezza dei programmi.*

SUGGERIMENTI PER ULTERIORI ATTIVITÀ E RIFERIMENTI

- Il primo luogo in cui trovare ulteriore materiale è il sito del *Programma il Futuro*. Contiene guide per gli insegnanti, videotutorial, eventi e iniziative legati al mondo dell'Informatica nella scuola italiana. <https://q3.hubscuola.it/ul5>
- Un libro che spiega passo passo come risolvere i puzzle di Code.org è:
Enrico Nardelli (2020). *La programmazione informatica per la scuola primaria*. Edizioni Themis. Scaricabile gratuitamente (CC BY-NC-SA 4.0) da <https://q3.hubscuola.it/qyh>.
- Per quanto riguarda l'utilizzo della metodologia Usa, Modifica, Crea:
 - La già citata guida e relativi materiali per le attività in classe. Silvia Caserio, Michael Lodi, Violetta Lonati, Marco Sbaraglia. Guida didattica UMC. <https://q3.hubscuola.it/g29>
 - Alcuni riferimenti scientifici sul design e la sperimentazione del percorso.
 - Sara Capecchi, Michael Lodi, Violetta Lonati e Marco Sbaraglia. 2023. *Castle and Stairs to Learn Iteration: Co-designing a UMC Learning Module with Teachers*. ITICSE 2023. <https://q3.hubscuola.it/gbv>
 - Sara Capecchi e altri. 2025. *Puzzle-based vs UMC approaches for learning iteration in grade 3*. Computer Science Education. <https://q3.hubscuola.it/pl8>
- Un confronto tra gli approcci guidati di Code.org e quelli più liberi e creativi si trova in:
Mitchel Resnick e Natalie Rusk (2020). *Coding at a Crossroads*, Communications of the ACM, vol. 63, no. 11, pp. 120-127. <https://q3.hubscuola.it/ak9>
Una traduzione italiana dell'articolo è disponibile qui:
<https://q3.hubscuola.it/vu6>

CAPITOLO 8: PROGRAMMAZIONE CREATIVA CON SCRATCH

Dopo le attività unplugged e l'uso di ambienti guidati come Code.org, **Scratch rappresenta il successivo passo naturale per introdurre gli alunni e le alunne alla programmazione.** Si tratta di un ambiente gratuito, progettato per permettere di esprimere le proprie idee attraverso l'informatica, costruendo giochi, animazioni e storie interattive mediante un linguaggio a blocchi semplice e intuitivo. Sviluppato dal gruppo Lifelong Kindergarten del MIT Media Lab e **ispirato alla teoria costruzionista dell'apprendimento**, Scratch offre un contesto di programmazione accessibile e ricco di possibilità. Come su Code.org, le istruzioni sono presentate sotto forma di blocchi colorati che si incastrano tra loro, eliminando le difficoltà legate alla sintassi e permettendo alle alunne e agli alunni di concentrarsi sulla logica, sulla progettazione e sulla creatività.

La filosofia educativa alla base di Scratch si riassume nelle "4 P": **Peers**, la collaborazione tra pari; **Projects**, l'apprendimento orientato a un progetto significativo; **Passion**, il coltivare interessi autentici; e **Play**, il valore del gioco come spazio di esplorazione [Resnick, 2018]. Ed è proprio questa filosofia a segnare la differenza principale rispetto ad ambienti più strutturati e sequenziali come Code.org: mentre questi ultimi guidano gli alunni e le alunne attraverso percorsi molto guidati, Scratch offre uno spazio aperto in cui esplorare, sperimentare e costruire in autonomia. Questa libertà progettuale consente di mettere in gioco le proprie idee, provare soluzioni alternative e sviluppare competenze creative e collaborative che vanno oltre la semplice risoluzione di esercizi.

Sebbene non sia l'unico strumento disponibile per introdurre la programmazione a scuola, Scratch è uno dei pochi ambienti progettati in modo pedagogicamente solido per le fasce d'età che vanno dalla primaria ai primi anni di secondaria di secondo grado. È gratuito, accessibile da qualsiasi dispositivo e nasce esplicitamente per supportare percorsi educativi basati sulla creatività, sulla collaborazione e sull'apprendimento attivo. Il suo design user-friendly lo rende adatto non solo agli alunni e alunne, ma anche a docenti e genitori che desiderano accompagnarli nelle attività. La versatilità dell'ambiente permette di utilizzarlo sia in contesti didattici sia in situazioni ricreative, poiché si adatta con facilità a diverse finalità e livelli di esperienza.

La combinazione di semplicità e potenzialità è uno dei motivi principali per cui Scratch è così diffuso: è possibile iniziare con progetti essenziali, come piccole animazioni o giochi molto semplici, e sviluppare progressivamente competenze più avanzate, esplorando concetti di logica, progettazione e pensiero computazionale in modo naturale e graduale. **Questa progressione fluida non è casuale**, ma riflette alcune scelte di design fondamentali, riassumibili in tre principi chiave [Resnick, 2018]:

- **Low floor** (pavimento basso): l'ambiente è facile da approcciare e permette di ottenere risultati soddisfacenti fin dai primi tentativi, grazie anche all'interfaccia grafica intuitiva.
- **High ceiling** (soffitto alto): una volta acquisite le basi, è possibile realizzare progetti sempre più complessi, aumentando gradualmente le proprie competenze.
- **Wide walls** (pareti ampie): lo stesso ambiente può essere utilizzato per creare progetti molto diversi tra loro (ad esempio giochi, storie, animazioni, arte interattiva, quiz...), favorendo la creatività, la sperimentazione e la produzione di artefatti originali.

PREREQUISITI

Le attività di questo capitolo sono veramente efficaci se alunni e alunne hanno già avuto qualche esperienza di base con i concetti informatici affrontati nelle schede precedenti, in particolare quelle dedicate alla programmazione visuale di Code.org. Avere già familiarità con idee come *sequenza di istruzioni*, *movimento di un personaggio* e *ripetizione* aiuta a comprendere più rapidamente il funzionamento dei blocchi di Scratch per poter sviluppare progetti più ricchi.

TRAGUARDI E OBIETTIVI GENERALI

- Orientarsi nell'ambiente di programmazione Scratch (stage, sprite, Area dei blocchi, Area del codice), riuscendo a selezionare personaggi e sfondi e ad avviare l'esecuzione di un programma.
- Creare programmi in sequenza, disponendo i blocchi in un ordine logico per far compiere azioni a un personaggio.
- Utilizzare le ripetizioni per eseguire più volte una stessa istruzione o sequenza di istruzioni, rendendo i programmi più compatti.
- Utilizzare in modo consapevole le istruzioni di movimento per controllare posizione, direzione e spostamenti degli sprite.
- Sincronizzare le azioni di più sprite, gestendo dialoghi, turni di parola mediante attese temporizzate.
- Sperimentare la strategia di prova ed errore, osservando il comportamento dello sprite, interpretando i feedback dell'ambiente ed effettuando correzioni (debugging) per migliorare il programma.
- Progettare e realizzare piccole animazioni o storie, combinando sprite, costumi, sfondi e blocchi di codice per dare forma a idee personali.

PREPARAZIONE

L'accesso alla piattaforma di Scratch è completamente libero, gratuito e immediato. È possibile utilizzare Scratch:

- **online**, all'indirizzo <https://q3.hubscuola.it/qno>;
- **offline**, scaricando l'applicazione Scratch Desktop (gratuita per Windows, Mac e Linux).

Sul sito ufficiale sono disponibili una vasta documentazione, una community internazionale e numerose risorse didattiche pensate appositamente per la scuola.

Iscrizione dell'insegnante e della classe a Scratch

Per utilizzare Scratch, è possibile utilizzare un account normale oppure un **account docente**, che permette di creare le proprie classi come in Code.org. A differenza degli account normali che sono attivi da subito, gli account docente richiedono circa 24 ore per essere approvati. Per registrare un account docente:

1. visita <https://q3.hubscuola.it/g02>;
2. seleziona "Richiedi un account docente";
3. compila il breve modulo con i tuoi dati;
4. conferma la registrazione tramite il link ricevuto via e-mail.

L'**account docente** di Scratch permette di creare e gestire classi virtuali, aggiungere alunni e alunne senza richiedere le loro e-mail, monitorare i loro progetti e controllare le impostazioni di privacy e condivisione. Offre quindi un ambiente più sicuro e organizzato, ideale per lavorare in classe in modo strutturato e per seguire facilmente i progressi di studenti e studentesse. Ulteriori informazioni sono disponibili in questa guida (in inglese).

<https://q3.hubscuola.it/2n8>

E se non vogliamo creare nessun account?

Esattamente come in Code.org, è possibile utilizzare Scratch anche **senza effettuare l'accesso**. In questo caso:

- i progetti possono comunque essere creati;
- ma **non verranno salvati** online o nella classe; è tuttavia possibile scaricarli come file locali.

Per attività singole (es. esercitazioni brevi), questa soluzione è pienamente utilizzabile. Basta accedere al sito di Scratch (<https://q3.hubscuola.it/5g2>) e cliccare su **Crea**.

ATTIVITÀ 1: Benvenuti in Scratch!

→ Vedi Quaderno di Informatica pp. 36-37

In questa prima attività è importante illustrare agli alunni e le alunne le aree dell'interfaccia.

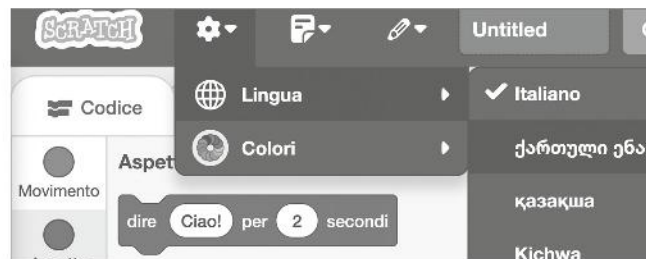
! MAESTR* SI È BUGGATO.. è tutto in inglese!

Scratch può essere un valido alleato per l'inclusione di chi non è italofono.

L'ambiente Scratch è infatti disponibile in circa 70 lingue: questo permette a ciascuno di lavorare nella lingua che conosce meglio, riducendo il carico cognitivo e favorendo la comprensione delle attività.

Per modificare la lingua è sufficiente aprire il menu **Impostazioni** → **Lingua** e selezionare l'opzione desiderata. L'interfaccia si aggiornerà immediatamente, rendendo i blocchi e i comandi più accessibili e consentendo ai bambini di partecipare con maggiore sicurezza alle attività di programmazione.

Dal menu Impostazioni è possibile scegliere anche colori ad elevato contrasto per facilitare la distinzione dei blocchi.



Ecco come appare la finestra di Scratch dopo averla aperta sul proprio pc.



L'interfaccia di Scratch è organizzata in aree ben distinte, pensate per aiutare gli alunni e le alunne a costruire i propri progetti in modo semplice e intuitivo. Nell'immagine sono visibili

le quattro zone principali: Lista dei blocchi, Area del codice, stage, Area degli sprite, Area degli sfondi. Comprendere la funzione di ciascuna parte è fondamentale per accompagnare alunni e alunne nei primi passi della programmazione.

- **Lista dei blocchi** (a sinistra): l'elenco dei blocchi disponibili, come quello presente nei Manuali delle istruzioni, divisi per categorie e riconoscibili grazie al codice colore.
- **Area del codice** (al centro): lo spazio per costruire i programmi, trascinando nell'area i blocchi presi dalla lista dei blocchi e incastrandoli tra loro. Ogni personaggio (chiamato **sprite**) avrà il **proprio codice**: quando si seleziona uno sprite, l'Area del codice mostra quello relativo a quel personaggio.
- **Stage** (in alto a destra): la scena in cui si svolge l'animazione o il gioco che si sta costruendo. Nello stage si vedono i personaggi in movimento, gli sfondi scelti e gli effetti delle azioni che hanno programmato. In alto compaiono anche i **pulsanti start** (bandierina verde) e **stop** (cartello stradale rosso).
- **Area degli sprite** (in basso a destra): qui sono mostrati gli sprite presenti nel progetto, lo sprite selezionato in quel momento (con il contorno azzurro) e le proprietà dello sprite selezionato (posizione con coordinate x e y, direzione, visibilità, dimensione). Da qui è possibile aggiungere nuovi sprite, cancellarli e duplicarli.
- **Area degli sfondi** (accanto all'Area degli sprite), dove è possibile cambiare o aggiungere sfondi allo stage.

In sintesi, l'immagine mostra **dove si trovano i blocchi** (colonna sinistra), **dove si costruisce il programma** (zona centrale), **dove si vede il risultato** (lo stage in alto a destra), **dove si gestiscono i personaggi e gli sfondi** (in basso a destra).



SUL QUADERNO DI INFORMATICA

Mentre riflettete sulla funzione delle varie parti della schermata, si possono tenere aperte le pagine 36 e 37 del Quaderno di Informatica.

ATTIVITÀ 2: Diventa un regista con Scratch!

→ Vedi Quaderno di Informatica pp. 38-39

In questa attività viene chiesto a bambini e bambine di diventare registi creando un'animazione con un dialogo, in cui il nostro personaggio potrà parlare tramite fumetti.

Prima di lavorare sul codice, vediamo come inserire uno sprite e uno sfondo.

Per creare un nuovo sprite bisogna premere sull'icona con la faccia del gatto e scegliere uno dei modi possibili (dall'alto verso il basso):

- caricare un file;
- farlo scegliere a Scratch in maniera casuale;
- disegnarlo;
- sceglierlo dalla libreria di Scratch.

Per creare un **nuovo sfondo** bisogna premere sull'icona in basso a destra dello schermo e scegliere uno dei modi possibili (dall'alto verso il basso):

- caricare un file;
- farlo scegliere a Scratch in maniera casuale;
- disegnarlo;
- sceglierlo dalla libreria di Scratch.



E ora, crea il codice!

Un programma è composto da una serie di blocchi colorati (tra quelli disponibili nella lista dei blocchi, a sinistra dello schermo) agganciati sequenzialmente uno sotto l'altro.

In Scratch ogni blocco ha un colore che identifica la categoria di comandi a cui appartiene: questo aiuta alunni e alunne a riconoscerne rapidamente la funzione e, con il tempo, a intuire che cosa fa un blocco ancora prima di usarlo; puoi decidere se presentarli passo passo oppure lasciare che li esplorino liberamente, sfruttando proprio il colore come guida pratica.

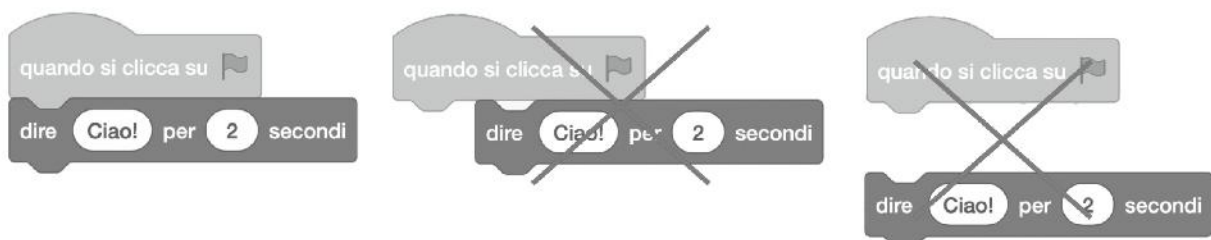
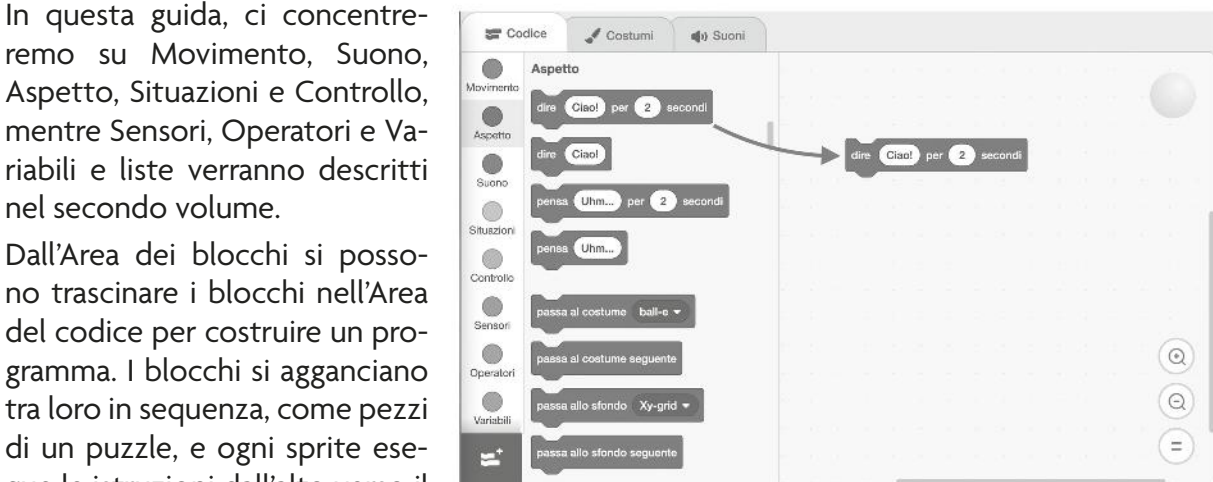
- **Movimento** (blu): istruzioni che fanno muovere lo sprite sulla scena.
- **Aspetto** (viola): istruzioni per far “parlare” e “pensare” uno sprite (con il meccanismo dei fumetti, quindi visualizzando del testo) e per cambiare il suo aspetto.
- **Suono** (rosa): istruzioni per riprodurre dei suoni.
- **Situazioni** (giallo): istruzioni per avviare una sequenza di istruzioni come reazione ad un evento, come il click sulla bandierina verde o la pressione di un tasto.
- **Controllo** (arancione): istruzioni che gestiscono il flusso del programma, permettendo di ripetere azioni, prendere decisioni, attendere, fermare l'esecuzione.
- **Sensori** (azzurro): istruzioni che permettono allo sprite di ottenere informazioni dall'ambiente (per esempio posizione del mouse, tasti premuti o contatto con altri oggetti) e usarle per guidare il comportamento del programma.
- **Operatori** (verde): istruzioni per eseguire operazioni aritmetiche e logiche, cioè verifiche di tipo vero/falso utili per decidere come far agire lo sprite.
- **Variabili e liste** (arancione chiaro): istruzioni per memorizzare informazioni e riutilizzarle nel programma, ad esempio per salvare il punteggio di un gioco.

In questa guida, ci concentreremo su Movimento, Suono, Aspetto, Situazioni e Controllo, mentre Sensori, Operatori e Variabili e liste verranno descritti nel secondo volume.

Dall'Area dei blocchi si possono trascinare i blocchi nell'Area del codice per costruire un programma. I blocchi si agganciano tra loro in sequenza, come pezzi di un puzzle, e ogni sprite esegue le istruzioni dall'alto verso il basso.

È possibile spostare, aggiungere o togliere blocchi in qualsiasi momento, modificando così il comportamento dello sprite.

Perché Scratch possa eseguire correttamente le istruzioni, i blocchi devono essere incastrati tra loro. Nella figura qui sotto il blocco a sinistra è agganciato correttamente, mentre gli altri due non lo sono: se i blocchi restano separati, verranno ignorati da Scratch.



Il programma, in questo caso, è composto da due istruzioni:

-  (**Situazioni**): indica che il programma inizierà quando l'utente cliccherà sul pulsante start (la bandiera verde che si trova sopra allo stage).
-  (**Aspetto**): fa comparire sullo sprite un fumetto con la scritta "Ciao!" per 2 secondi. Il numero di secondi da inserire dipende dalla lunghezza del testo e dal tempo necessario a chi guarda per leggerlo con calma: per frasi più lunghe può essere utile aumentare l'attesa.

In Scratch alcuni blocchi contengono parti già definite e parti "in bianco" che possono essere modificate. Queste parti modificabili si chiamano **parametri**: basta cliccarci sopra per cam-

biare il valore. Ad esempio, nel blocco  è possibile scrivere il testo che comparirà nel fumetto e decidere per quanti secondi mostrarlo.



SUL QUADERNO DI INFORMATICA

Fai scegliere a ciascuno un personaggio e due o tre frasi da fargli pronunciare con i blocchi "dire". Puoi proporre di seguire i passi a pagina 38 e 39 del Quaderno di Informatica.

ATTIVITÀ 3: Dialogo tra due personaggi

→ Vedi Quaderno di Informatica pp. 40-41

Sincronizzazione

Quando ci sono più personaggi, ciascuno può avere il proprio blocco . In questo caso, tutti gli sprite inizieranno ad eseguire il loro codice nello stesso momento.

Se, ad esempio, abbiamo un blocco  in ognuno di loro, tutti mostreranno il fumetto contemporaneamente. Se però vogliamo che parlino uno alla volta, o che uno sprite inizi a muoversi solo dopo che un altro ha finito, dobbiamo **sincronizzare** le loro azioni, cioè decidere l'ordine e i momenti in cui ogni sprite deve attivarsi.

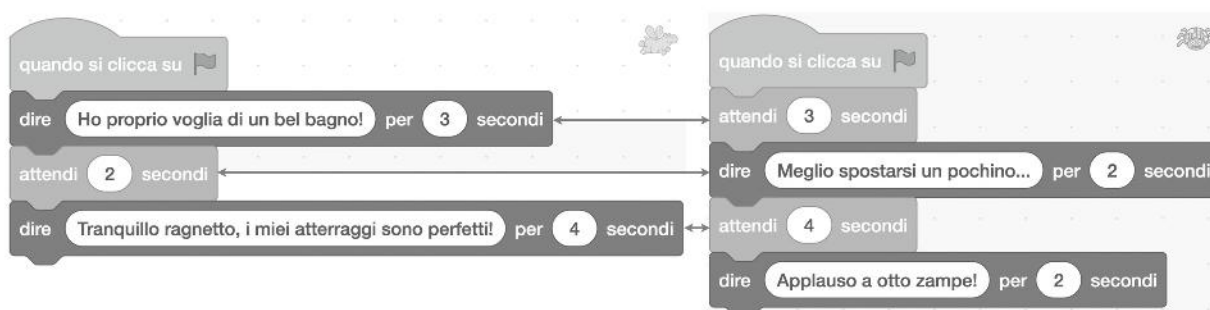


SUL QUADERNO DI INFORMATICA

A pagina 40 del Quaderno di Informatica, l'ippopotamo pronuncia la prima battuta e solo dopo alcuni secondi il raghetto¹ risponde. Per ottenere questo comportamento ordinato non basta mettere i blocchi in sequenza: bisogna dire allo sprite **quando** deve iniziare la sua azione.

Utilizziamo il blocco  (**Controllo**), che permette allo sprite di fare una pausa prima di passare alle istruzioni successive. In questo esempio, il raghetto aspetta 3 secondi, cioè il tempo che serve per far finire la frase all'ippopotamo: come si vede nella figura, i blocchi attendi permettono ai personaggi di parlare uno dopo l'altro, senza sovrapporsi.

¹ Lo sprite scelto è Ladybug2, che, stando al nome, è una coccinella. Continueremo a chiamarlo "raghetto" per maggiore suggestività. Al lettore pignolo facciamo notare che l'ippopotamo ha due ali da insetto 😊.



Prima di programmare può essere utile creare un piccolo storyboard: una sequenza di vignette o appunti che raccontano, passo dopo passo, cosa succederà nella storia o nell'animazione. Questo passaggio permette a ciascuno di pianificare con chiarezza le scene, decidere chi parla, chi si muove e quando, e trasformare l'idea in un progetto ordinato. Lavorare con uno storyboard evita confusione in fase di programmazione e aiuta a mantenere una visione d'insieme del progetto.

Personaggio: Lupo		Personaggio: Cappuccetto Rosso	
Ciao bella bambina!	3	ATTENDI	5
Come ti chiami?	2		
ATTENDI	4	Mi chiamo Cappuccetto Rosso!	4
Ma che bel nome!	2	ATTENDI	2

Ripetizione

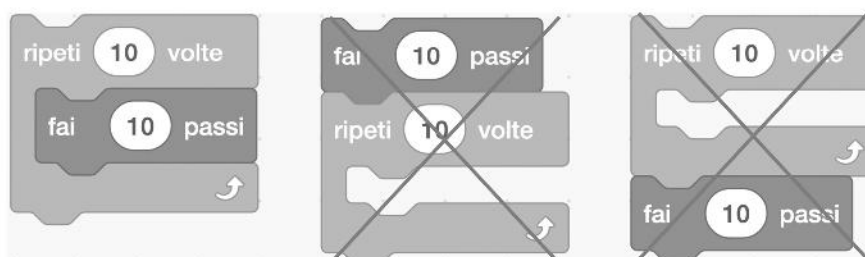
Supponiamo che il ragnetto non si fidi delle doti dell'ippopotamo e voglia spostarsi sullo schermo. Per farlo potrebbe usare più volte il blocco **fai 10 passi**, ma il programma diventerebbe lungo, difficile da leggere e da modificare.

Per rendere il codice più semplice possiamo usare il blocco **ripeti 10 volte** (Controllo). Questo blocco ha la forma di una "C" aperta: la sequenza delle istruzioni inserite al suo interno verranno ripetute per il numero di volte indicato. Le istruzioni che si trovano fuori dalla "C" non verranno ripetute.



! MAESTR* SI È BUGGATO... non ripete niente!

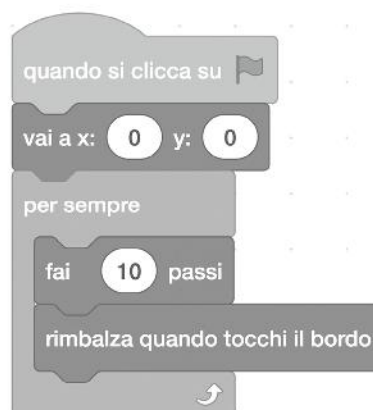
È utile far notare che, soprattutto all'inizio, alunni e alunne incastrano i blocchi nel punto sbagliato: ad esempio, può capitare che il blocco **fai 10 passi** venga messo prima o dopo la "C", pensando che venga comunque ripetuto. Mostrare loro l'effetto dell'errore aiuta a comprendere meglio la logica della ripetizione e l'importanza della posizione dei blocchi.



Ripetizione infinita

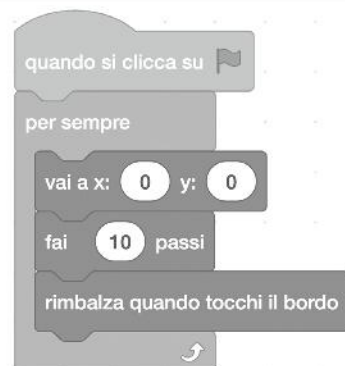
Possiamo anche ripetere una sequenza di blocchi all'infinito con il blocco **per sempre** (Controllo). A differenza di **ripeti volte**, questo blocco non ha l'incastro sul fondo: significa che il programma continuerà a eseguire all'infinito le istruzioni contenute al suo interno, senza arrivare mai a un blocco successivo. Come nel caso di **ripeti volte**, verranno ripetuti solo i blocchi inseriti dentro la "C".

Nel codice a destra, lo sprite viene prima posizionato al centro dello stage (x: 0, y: 0) e poi, **per sempre**, fa 10 passi e rimbalza quando tocca il bordo: si muoverà quindi in continuazione avanti e indietro.



MAESTR* SI È BUGGATO... non si muove!

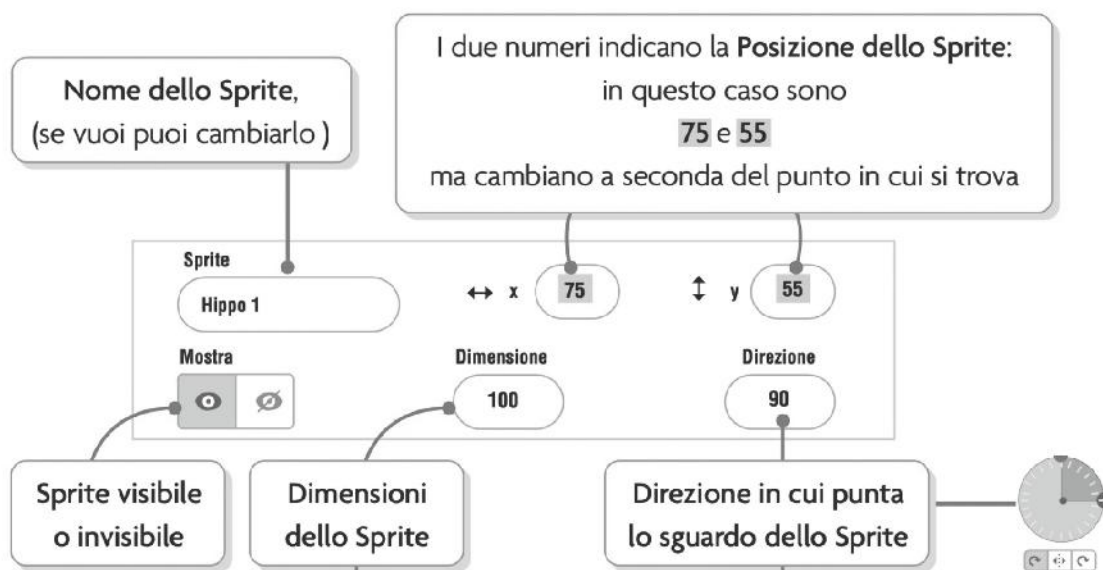
Nel codice a destra, invece, anche il blocco **vai a x: 0 y: 0** è inserito dentro la "C": in questo caso lo sprite torna al centro dello stage ad ogni ripetizione, fa solo pochi passi e poi ricomincia subito da capo. Il movimento è impercettibile e può sembrare che lo sprite "non si muova". In questi casi, può essere utile chiedere al proprio alunno/a di leggere il codice ad alta voce, passo per passo ("cosa succede per primo? e poi? e poi?") per facilitare il ragionamento di studenti e studentesse sul comportamento dei propri blocchi.



ATTIVITÀ 4: Movimenti precisi

→ Vedi Quaderno di Informatica pp. 42-43

Ogni sprite, oltre al nome, ha diverse proprietà visibili nell'Area degli sprite. In questa sezione è possibile vedere e modificare informazioni come posizione, direzione, dimensione e visibilità, come mostrato nell'immagine.



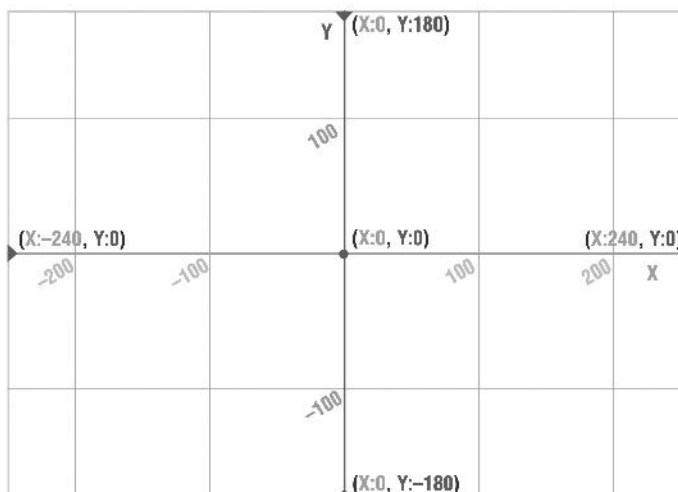


SUL QUADERNO DI INFORMATICA

Incoraggiare alunni e alunne ad esplorare cosa succede cambiando la proprietà **dimensione** e la proprietà **direzione** nell'Area degli sprite, completando i box a pagina 42 del Quaderno di Informatica.

La posizione

Lo sfondo visibile nell'immagine è uno degli sfondi predefiniti di Scratch (**XY-grid**). Mostra allo stesso tempo lo stage e il sistema di coordinate cartesiane che lo struttura. La larghezza misura 480 unità (240 a destra e 240 a sinistra del centro) e l'altezza 360 unità (180 sopra e 180 sotto il centro). Il punto centrale dello stage è indicato dalle coordinate $x=0$ e $y=0$: la linea arancione (orizzontale) rappresenta l'asse X, la linea blu (verticale) l'asse Y.

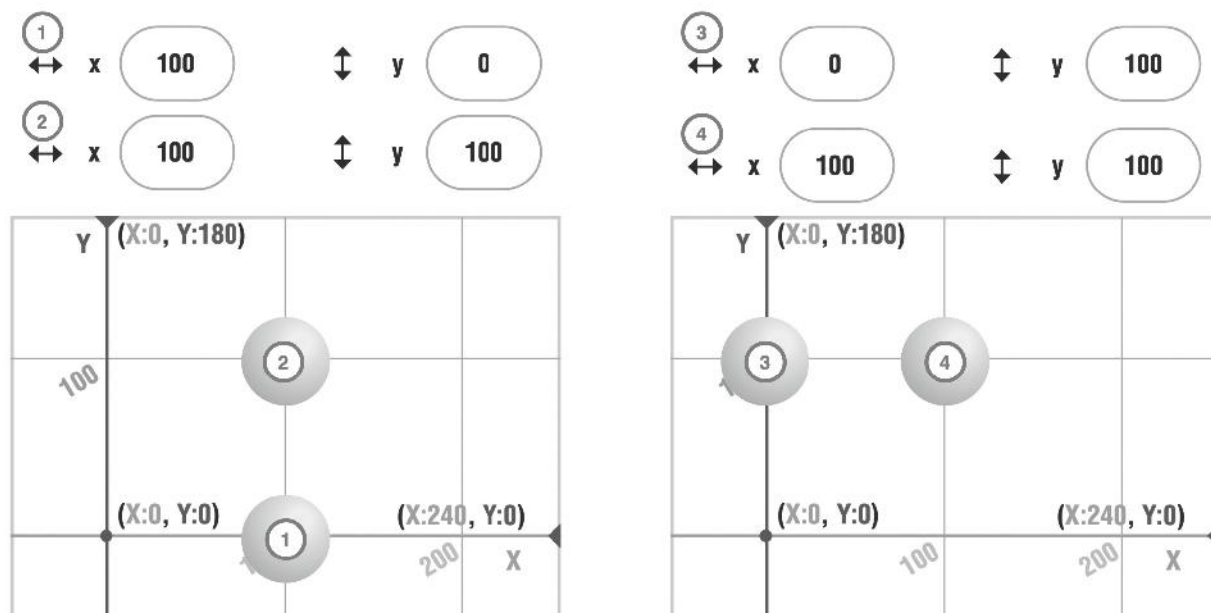


Sebbene in terza primaria è possibile che piano cartesiano e numeri negativi non siano ancora stati oggetto di insegnamento esplicito, Scratch offre la possibilità di incontrarli in modo naturale e graduale: il semplice fatto di copiare i valori indicati, provare a modificarli e osservare cosa succede ai personaggi può aiutare a orientarsi sullo stage e a prendere confidenza con l'idea di movimento sul piano.



SUL QUADERNO DI INFORMATICA

Incoraggiare alunni e alunne a esplorare cosa accade cambiando la posizione dello sprite, seguendo le attività proposte a pagina 43 del Quaderno di Informatica.



ATTIVITÀ 5: Posizionare un personaggio

→ Vedi Quaderno di Informatica p. 44

Scegliere una posizione

Il blocco  (**Movimento**) permette di posizionare lo sprite in una precisa posizione dello stage. Per trovare le coordinate da inserire, bisogna (1) spostare lo sprite con il mouse nella posizione desiderata, (2) leggere i valori x e y riportati nell'Area degli sprite e (3) copiarli dentro il blocco. In questo modo il personaggio si sposterà sempre nel punto scelto. Questa procedura aiuta alunni e alunne a scoprire che lo sprite può essere posizionato sia “a mano” nell'interfaccia, come se fosse telecomandato, sia usando il blocco di movimento, cioè **pianificando** in anticipo le azioni. Questo passaggio può aiutare a capire la differenza tra muovere direttamente il personaggio e farlo muovere da solo seguendo le istruzioni del programma.

**SUL QUADERNO DI INFORMATICA**

Proponete di copiare il programma mostrato a pagina 44 del Quaderno di Informatica. Incoraggiate bambini e bambine a provare a far partire il personaggio in diversi punti dello schermo, modificando le coordinate di partenza.

**MAESTR* SI È BUGGATO... perché non comincia come voglio io?**

Al termine dell'esecuzione del programma, può capitare che il personaggio resti in una posizione sullo stage non desiderata. Quando si clicca di nuovo sulla bandierina verde, lo sprite ricomincia a muoversi proprio da lì, rovinando l'effetto voluto.

Questo accade perché ogni personaggio possiede un proprio **stato**, cioè un insieme di informazioni che Scratch conserva: posizione, direzione, dimensione, visibilità, e così via. In pratica, lo sprite non “si resetta da solo”: si comporta come un attore un po' distratto, che riparte da dove era rimasto se non gli viene detto con precisione cosa fare.

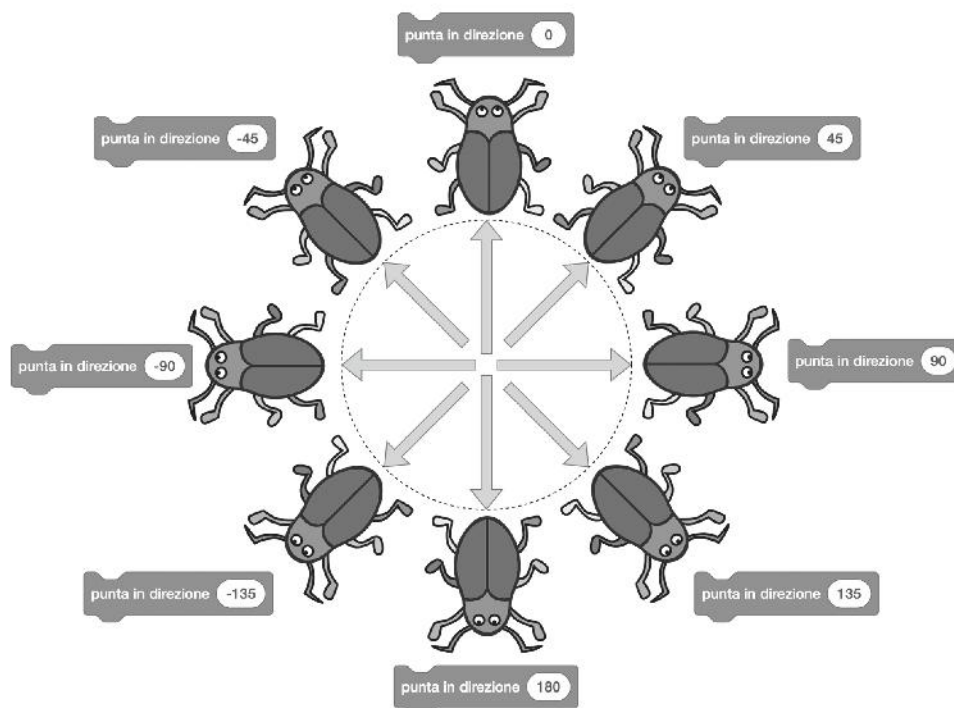
Per questo è importante inserire all'inizio dello script i blocchi che fissano la posizione di partenza (e, se serve, dimensione e direzione). Un buon trucco in classe è chiedere a ciascuno di leggere a voce alta il proprio codice passo per passo chiedendosi: “Dove sta il personaggio adesso? Dove gli ho detto di andare? Da dove sta ripartendo?”

Questo aiuta a capire che anche la posizione iniziale fa parte della programmazione.

Scegliere una direzione

Nell'attività di pagina 42 del Quaderno di Informatica abbiamo chiesto agli alunni e alle alunne di esplorare il concetto di direzione. La direzione indica “dove lo sprite sta guardando”: è come girarsi prima di iniziare a camminare. Come per la posizione, oltre a scegliere la direzione tramite l'interfaccia, è possibile utilizzare il blocco  (**Movimento**) per pianificare la direzione del movimento.

Questo blocco funziona con gli angoli misurati in gradi e il valore fa riferimento all'ampiezza dell'angolo centrato nell'origine del sistema di riferimento che struttura lo stage e che ha: un lato sul semiasse y positivo e il secondo lato dato dalla direzione in cui punta lo “sguardo dello sprite”. Quindi, inserendo il valore 0 lo sprite punterà verso l'alto, con 180 verso il basso, con -90 verso sinistra e con 90 verso destra, come mostrato nella figura seguente.

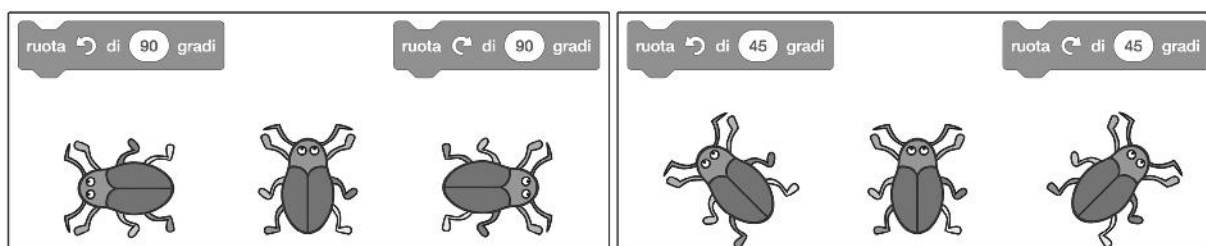


SUL QUADERNO DI INFORMATICA

Puoi proporre ai tuoi alunni e alunne l'attività di pagina 45 del Quaderno di Informatica, esplorando come la direzione di partenza possa cambiare il movimento.

Nel Quaderno di Informatica non compare, ma esiste un secondo modo per cambiare direzione: la **rotazione relativa**. È simile ai comandi “gira a sinistra” e “gira a destra” visti nel Capitolo 7 su Code.org, ma in Scratch si realizza tramite i blocchi **ruota** di gradi (Movimento), in senso orario o antiorario, indicato dalla freccia sul blocco.

Questi blocchi fanno ruotare lo sprite attorno al proprio centro e di un angolo dell'ampiezza in gradi indicata rispetto alla direzione in cui sta guardando in quel momento, e non verso una direzione assoluta. Richiedono familiarità con il concetto di angolo, che in terza primaria potrebbe non essere ancora stato affrontato formalmente, ma questi blocchi possono essere utili per comprendere meglio come funziona il movimento e per rispondere agli errori più comuni degli alunni.



Approfondimento: Spostamenti assoluti e relativi

I comandi di movimento disponibili in Scratch sono molti di più di quelli proposti nelle attività del Quaderno di Informatica. È possibile che alcuni vengano scoperti da alunni e alunne spontaneamente esplorando l'interfaccia: per questo motivo offriamo qui una breve spiegazione aggiuntiva, così da poter rispondere con sicurezza alle loro domande.

Prima di tutto, è utile distinguere due modalità di spostamento:

- **Movimento assoluto:** lo sprite raggiunge un punto preciso dello stage indicato dalle coordinate x e y ;
- **Movimento relativo:** lo sprite si sposta partendo dalla posizione in cui si trova, spostandosi nella direzione verso cui è orientato.

Questa distinzione sarà molto utile per interpretare il comportamento dello sprite e per accompagnare alunni e alunne a comprendere gradualmente come lo spazio e il movimento funzionano in Scratch.

Istruzione	Significato
	Sposta lo sprite del numero di passi desiderato, relativamente alla posizione e direzione corrente.
	Sposta lo sprite nel punto di coordinate (x, y) . Il movimento è istantaneo.
	Sposta lo sprite nel punto di coordinate (x, y) partendo dalla posizione attuale e impiegando il tempo richiesto. Aumentando o diminuendo il valore del tempo, cambia la velocità dello spostamento.
	Cambia la coordinata x / y della posizione dello sprite, sostituendola con il valore specificato.
	Cambia la coordinata x / y della posizione dello sprite, incrementandola del valore specificato.
	Quando lo sprite tocca il bordo, si allontana dal bordo come se avesse colpito una superficie rigida.
	Permette di selezionare lo stile di rotazione: sinistra-destra significa che lo sprite può guardare a sinistra e a destra, ma non si inclinerà.



MAESTR* SI È BUGGATO... non si muove!

Può capitare che, anche usando i blocchi di movimento corretti, lo sprite sembri non spostarsi dalla posizione iniziale, come nel programma a fianco (provatelo!). In realtà si sta muovendo eccome, ma così velocemente che l'occhio non fa in tempo a percepire il cambiamento. Per questo motivo, dal punto di vista grafico il programma qui a fianco è equivalente al solo ultimo blocco.

Per rendere visibile il movimento potete:

inserire un blocco tra un movimento e l'altro, così lo sprite farà una piccola pausa prima del passo successivo; oppure usare il blocco , che mostra il percorso sullo stage in modo fluido e chiaro.



**MAESTR* come faccio ad andare indietro?**

In Scratch il blocco  fa muovere lo sprite nella direzione in cui sta guardando. Per andare nella direzione opposta non serve un blocco nuovo: basta usare un numero negativo. Per esempio, il valore -10 fa muovere lo sprite di 10 passi nel verso opposto a dove sta guardando. Questo è un ottimo momento per far notare a bambini e bambine che anche i numeri con il segno davanti hanno un effetto preciso sul movimento. Non è necessario formalizzare i numeri negativi dal punto di vista matematico: si tratta di sperimentare e osservare cosa succede allo sprite quando si cambia il valore numerico nel blocco.

ATTIVITÀ 6A: Sperimentare programmi

→ Vedi Quaderno di Informatica p. 46

**SUL QUADERNO DI INFORMATICA**

Per far lavorare bambini e bambine a questa attività, potete chiedere di tenere aperta la pagina 46 del Quaderno di Informatica e copiare i programmi proposti.

Copiare e provare i programmi già pronti, come quelli proposti in questa pagina sul Quaderno di Informatica, è un modo efficace per imparare a programmare, soprattutto se accompagnati da domande che stimolano la riflessione. Mentre ricostruiscono i blocchi uno alla volta, gli alunni e le alunne osservano cosa succede sullo schermo, fanno ipotesi, notano differenze e scoprono nuovi concetti senza bisogno di spiegazioni astratte.

Questo approccio offre la possibilità di:

- imparare osservando cosa fa ogni blocco;
- capire la struttura del programma semplicemente ricreandolo;
- esplorare idee nuove in modo guidato ma creativo;
- aumentare il proprio senso di sicurezza riproducendo un codice che funziona.

È un po' come quando si impara a leggere copiando parole: prima si imita, poi si diventa capaci di crearne di nuove. Copiare programmi non è solo utile: è un passo fondamentale per costruire autonomia e padronanza nella programmazione.

Alcuni suggerimenti operativi:

- Dopo aver copiato un programma e prima di eseguirlo, invitate bambini e bambine a rispondere alla domanda: "Secondo te cosa succederà?"
- Chiedete agli alunni e alle alunne di modificarne un solo elemento: ad esempio un numero, un blocco, un costume o una direzione. Una variazione alla volta aiuta a capire quale blocco produce quale effetto.

ATTIVITÀ 6B: Solo 8 blocchi!

→ Vedi Quaderno di Informatica p. 47

**SUL QUADERNO DI INFORMATICA**

Per lavorare a questa attività, potete proporre di tenere aperta la pagina 47 del Quaderno di Informatica, aiutando gli alunni a cercare i blocchi nell'Area dei blocchi in base ai colori.

In questa sfida gli alunni e le alunne devono inventare un programma utilizzando tutti e soli gli 8 blocchi proposti nel Quaderno di Informatica.

A differenza della “tela bianca” di Scratch, che permette di usare subito tutti i comandi disponibili, qui si lavora entro un insieme ristretto di blocchi. Questo non è un limite: anzi, i vincoli diventano un motore per la creatività. Quando le possibilità sono definite, si è spinti a esplorare ogni blocco, anche quelli che normalmente verrebbero ignorati, e a combinarli in modi nuovi.

Per questo tipo di attività, l'insegnante può creare altre sfide scegliendo set diversi di blocchi (per esempio prendendone 7, 8 o 10). L'importante è che i blocchi selezionati permettano comunque di costruire programmi vari e significativi, possibilmente diversi da quelli già realizzati in classe, così da ampliare l'immaginario di ciascuno e stimolare nuove idee.

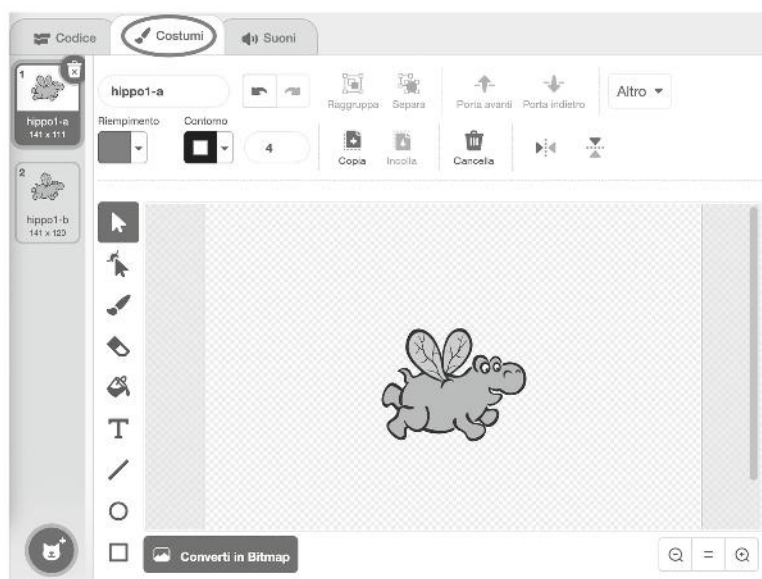
BONUS: COSTUMI E SUONI

Per concludere, parliamo di due funzionalità che non sono state incluse nel Quaderno di Informatica per motivi di spazio, ma possono essere interessanti da presentare in classe: costumi e suoni. Alcuni degli sprite di Scratch hanno a disposizione diversi **costumi**, cioè immagini alternative con aspetti o posizioni differenti. Allo stesso modo, ogni sprite può essere associato a uno o più **suoni**, che possono essere riprodotti dalle casse del computer.

È possibile vedere costumi e suoni associati allo sprite cliccando sulle linguette a fianco della linguetta codice, in alto sopra l'Area dei blocchi. Per tornare al codice, è sufficiente cliccare sulla linguetta corrispondente.

Nella scheda “Costumi” è possibile scegliere costumi, disegnarne di nuovi, duplicarli o modificarli. I costumi permettono di trasformare lo sprite in modo creativo, ad esempio cambiando espressione, postura, colori o dettagli del personaggio.

Sono anche uno strumento importante per creare **animazioni**: alternando due o più costumi e combinandoli con blocchi di movimento, si può simulare l'effetto di un personaggio che cammina, corre o sbatte le ali. Nello sprite dell'ippopotamo, per esempio, esistono due costumi che mostrano le ali in posizioni diverse (su/giù): utilizzando il blocco **passa al costume seguente** (Aspetto) dentro una ripetizione per sempre è possibile fare battere le ali al nostro ippopotamo.

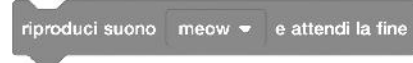


Per creare un **nuovo suono** bisogna premere sull'icona con l'altoparlante e scegliere uno dei modi possibili (dall'alto verso il basso):

- caricare un file audio dal computer;
- far scegliere a Scratch un suono dalla libreria in maniera casuale;
- registrarlo;
- sceglierlo dalla libreria di Scratch.



È poi possibile riprodurre il suono scelto con il blocco **(Suono)**.



MODALITÀ DI LAVORO

Programmazione in coppia

Nel Capitolo 7 abbiamo introdotto la metodologia della programmazione in coppia, evidenziando i ruoli di **pilota** e **navigatore**. Quel modello resta valido anche in Scratch: lavorare in coppia è un potente strumento per favorire attenzione, verbalizzazione, confronto, pianificazione e debugging.

In Scratch però il contesto cambia. Non ci sono puzzle da risolvere, né un pulsante dedicato a tracciare automaticamente il lavoro dei partner. Qui gli alunni e le alunne non si limitano a seguire un percorso guidato: progettano liberamente storie, dialoghi, animazioni e movimenti sullo stage. Questo rende la collaborazione ancora più preziosa, perché richiede decisioni condivise su cosa fare, come farlo e perché.

Alcuni consigli per aiutare le coppie a collaborare:

- invitali a discutere e concordare insieme la scena da realizzare prima di iniziare a programmare;
- chiedi al pilota di raccontare le proprie scelte;
- ricorda a ciascuna coppia di scambiarsi i ruoli regolarmente.

Usa-Modifica-Crea (UMC)

Anche in Scratch ritroviamo l'approccio UMC, già introdotto nel Capitolo 7: si parte da programmi esistenti, si esplorano piccole modifiche e poi si arriva alla creazione di progetti personali. La logica di fondo resta la stessa, ma in Scratch assume una forma più aperta e creativa.

- **Usa:** alunni e alunne provano programmi già costruiti dall'insegnante (per esempio animazioni o dialoghi) e osservano che cosa succede sullo stage, senza dover ricostruire tutto da zero.
- **Modifica:** alunni e alunne cambiano numeri, blocchi, direzioni, costumi o tempi, prima facendo previsioni e poi confrontando previsione e risultato. Anche un singolo cambiamento può trasformare in modo evidente il comportamento dello sprite.
- **Crea:** grazie alla libertà di movimento nello stage, ai costumi, ai suoni e ai dialoghi, la fase di creazione diventa naturalmente più ampia: ciascuno può progettare qualcosa di unico, non solo una variante del modello di partenza.

In Scratch, quindi, l'UMC mantiene la struttura per proporre i concetti, ma si arricchisce di un elemento nuovo: non si risolve un problema dato, si progetta un'idea personale. Questo valorizza il pensiero creativo e la progettazione autentica, mettendo al centro l'immaginazione di alunni e alunne.

ASPETTI INFORMATICI

In questo capitolo, alunni e alunne continuano a lavorare con un linguaggio di programmazione visuale, ritrovando molti concetti già incontrati con Code.org: **sintassi** (i blocchi si incastrano solo in certi modi), **semantica** (ogni blocco ha un significato preciso), **esecutore meccanico** (lo sprite fa solo ciò che è scritto), **algoritmi** e **programmi**, **sequenza** e **ripetizioni**, **parametri** e **debugging**. La logica di fondo non cambia: un programma è una sequenza di istruzioni che il computer esegue alla lettera.

Quello che cambia è l'ambiente: in Scratch il programmatore deve ragionare non solo su **cosa** fare, ma anche su **dove** farlo accadere e su **come** un personaggio può comportarsi nel tempo. Questo introduce alcune idee nuove e importanti.

- **Creatività e progettazione libera.** A differenza dei puzzle proposti in Code.org, qui non c'è un obiettivo già definito da raggiungere. Alunni e alunne devono decidere che cosa vogliono ottenere: una scena, un dialogo, un'animazione, una storia, un gioco. L'informatica non è più solo trovare la soluzione corretta a un problema dato, ma immaginare il problema stesso e costruire un percorso per risolverlo. La dimensione creativa non è un'aggiunta estetica: è parte centrale della competenza informatica, perché costringe a fare scelte, a definire scopi e a trasformarli in istruzioni chiare.
- **Debugging come ricerca del risultato desiderato.** In questo contesto il debugging assume un significato diverso rispetto ai puzzle: non si tratta più soltanto di far funzionare il programma nel modo "giusto", ma di capire perché il comportamento osservato è diverso da quello voluto. Bambini e bambine devono confrontare l'idea che avevano con ciò che accade sullo stage, e rivedere il codice fino a ottenere l'effetto desiderato. Questo rende il debugging un processo più aperto e riflessivo, dove si impara dai propri tentativi, invece di inseguire un'unica risposta esatta.

RELAZIONE CON L'INFORMATICA NELLE INDICAZIONI NAZIONALI

Relativamente alle Indicazioni Nazionali 2025, queste attività contribuiscono allo sviluppo delle seguenti competenze.

Competenze (alla fine della scuola primaria) nella disciplina Matematica:

- *Scrivere e comprendere semplici programmi, espressi in elementari linguaggi di programmazione a scopo didattico, e valutarne l'adeguatezza rispetto al compito che si vuole automatizzare.*

Competenze (alla fine della scuola primaria) nella disciplina Tecnologia:

- *Sviluppare un atteggiamento positivo nei confronti delle applicazioni informatiche riconoscendone le potenzialità come strumenti di espressione personale nella vita quotidiana.*

Le attività proposte contribuiscono ai seguenti obiettivi:

Obiettivi (alla fine della terza primaria) nella disciplina Matematica

- *Scrivere semplici programmi e verificare, mediante la loro esecuzione, se svolgono il compito previsto ed eventualmente correggerli.*

Obiettivi (alla fine della terza primaria) nella disciplina Tecnologia

- *Creare, selezionare e usare semplici testi o disegni, usando strumenti informatici.*

Infine, lavorano sulle seguenti conoscenze (inserite nelle Indicazioni Nazionali 2025 a titolo di esempio, non prescrittive):

Conoscenze nella disciplina Matematica

- *Concetto di algoritmo e sua esecuzione rigorosa.* In particolare qui si lavora sulla creazione ed esecuzione di programmi, cioè algoritmi scritti in un linguaggio di programmazione (qui, in particolare, il linguaggio Scratch);
- *Strutture di controllo fondamentali di un linguaggio di programmazione e reazione agli eventi.*
- *Controllo di correttezza dei programmi.*

Conoscenze nella disciplina Tecnologia

- *Creazione e modifica di semplici contenuti digitali e multimediali usando semplici applicazioni informatiche, per raccontare storie o esperienze personali.*

VALORE COSTRUZIONISTA

In Scratch l'apprendimento non nasce dalla ricerca della “risposta giusta”, ma dalla costruzione di qualcosa di personale e significativo. Ogni progetto è un artefatto che prende forma nel tempo attraverso tentativi, modifiche e nuove idee. Questo modo di lavorare rispecchia l'idea dell'**apprendimento creativo**, al centro della spirale proposta da Resnick (2018): immaginare → creare → giocare → condividere → riflettere, e di nuovo immaginare:

- **Immaginare:** ciascuno parte da un'idea propria, per esempio inventando una storia.
- **Creare:** si traduce l'idea in blocchi di codice, costumi, movimenti e posizioni, scoprendo strumenti nuovi mentre vengono usati.
- **Giocare:** si esegue il programma e si osserva cosa accade, divertendosi a modificare blocchi, numeri e costumi per vedere nuovi effetti.
- **Condividere:** ciascuno mostra il proprio lavoro agli altri, facendoglielo provare e ascoltando idee e punti di vista diversi.
- **Riflettere:** ci si accorge di cosa funziona e cosa no, e si cambia il progetto per migliorarlo o trasformarlo.

In classe questo processo appare molto chiaramente: il debugging non serve solo a correggere errori tecnici, ma a chiarire l'intenzione del progetto (“cosa volevo fare davvero?”), e ogni modifica apre possibilità nuove. L'attenzione si sposta dal risultato all'esperienza – una forma di apprendimento attivo che unisce creatività, logica e collaborazione.

SUGGERIMENTI PER ULTERIORI ATTIVITÀ E RIFERIMENTI

- Una guida che propone un percorso per insegnare l'informatica creativa e l'apprendimento creativo con Scratch è disponibile con licenza libera. Karen Brennan e colleghi. Creative Computing Curriculum Guide. Traduzione in italiano: <https://q3.hubscuola.it/4ca>
- I creatori di Scratch portano avanti la comunità di educatori *Learning Creative Learning*, che approfondisce i principi dell'apprendimento creativo. Sono presenti moderatori italiani e una discreta comunità di insegnanti. È possibile partecipare liberamente tramite il sito <https://q3.hubscuola.it/aor> (disponibile anche in italiano).
- Michael Lodi, Renzo Davoli, Rebecca Montanari, Simone Martini (2020). Informatica senza e con computer nella Scuola Primaria. Sezione “Informatica creativa con Scratch”.

Capitolo del volume “Coding e oltre: Informatica nella scuola”, a cura di Enrico Nardelli, edito da Lisciani. Una versione libera del capitolo è disponibile a <https://q3.hubscuola.it/cm6>

- Marco Giordano e Caterina Moschetti. 2016. “Coding e pensiero computazionale nella Scuola primaria”. ELI -La Spiga Edizioni. Prefazione a cura di M. Lodi, consultabile qui: <https://q3.hubscuola.it/bvn>.
- Una sequenza di percorsi didattici – in inglese – che applica una metodologia basata su Usa-Modifica-Crea è <https://q3.hubscuola.it/zkq>
- Mitchel Resnick (2018). Come i bambini. IMMAGINA, CREA, GIOCA e CONDIVIDI. Coltivare la creatività con il Lifelong Kindergarten del MIT. Erickson.

CAPITOLO 9: CONCLUSIONI

Al termine di questo percorso, pensiamo che la nostra visione sia chiara: l'informatica non è una semplice abilità tecnica né un insieme di strumenti da saper usare. È una **disciplina scientifica**, con i suoi concetti fondanti, i suoi linguaggi e i suoi metodi, che studia come rappresentare ed elaborare informazioni in modo automatico. Al pari della matematica o delle scienze, offre una lente interpretativa sul mondo e fornisce strumenti concettuali per comprenderlo e agire in modo consapevole.

Questa prospettiva è oggi più rilevante che mai. I bambini e le bambine che incontriamo nei primi anni della scuola primaria vivranno e lavoreranno in una società sempre più segnata dalla presenza dei sistemi digitali. Interagiranno quotidianamente con sistemi automatici che prendono decisioni, suggeriscono azioni, classificano informazioni e rispondono a istruzioni. Prepararli a comprendere questo futuro non significa insegnare loro a usare applicazioni specifiche, destinate a cambiare rapidamente, ma fornire basi solide per capire **come funzionano** questi sistemi e **come dialogare** con essi.

L'avvento e la diffusione dell'intelligenza artificiale rendono questa esigenza ancora più urgente. Se formiamo i nostri alunni e alunne ad essere bravi esecutori di istruzioni, li mettiamo in competizione diretta con macchine che, su quel terreno, sono destinate a superarli. Se invece li aiutiamo a diventare capaci di **pensare, formulare e valutare soluzioni e algoritmi**, allora l'intelligenza artificiale può diventare uno strumento potente al loro servizio.

Nella nostra visione, l'informatica, lungi da essere una disciplina tecnica, è strettamente legata allo sviluppo di competenze di **problem solving** consapevole e sistematico: non si tratta solo di trovare una risposta, ma di progettare una strategia, metterla alla prova, confrontarla con alternative e migliorarla progressivamente.

In questo processo, diventano centrali alcune competenze chiave. In primo luogo, la capacità di **esprimersi in modo chiaro e non ambiguo**: descrivere una procedura, scrivere un algoritmo o progettare un programma richiede precisione nel linguaggio e attenzione ai dettagli. In secondo luogo, la capacità di **immedesimarsi nell'altro**, che sia un compagno o una compagna di classe, un robot o un computer: bisogna saper anticipare come le istruzioni verranno interpretate, distinguendo ciò che è implicito per chi scrive da ciò che è esplicito per chi esegue. Infine, la capacità di **rivedere criticamente il proprio lavoro**, osservando gli effetti delle istruzioni date, individuando errori o ambiguità e correggendoli mediante pratiche di debugging. In questo quadro, l'errore diventa un comportamento da accogliere e osservare, assumendo un ruolo centrale e positivo.

Tutto questo non può essere rimandato ai gradi di istruzione superiori. Al contrario, richiede un lavoro paziente e progressivo che può e deve iniziare fin dai primi anni. Le attività unplugged, i giochi di movimento su griglia, la rappresentazione delle informazioni, la risoluzione di problemi e, più avanti, la programmazione visuale e creativa mostrano che è possibile introdurre concetti informatici profondi in modo accessibile, concreto e significativo già nella scuola primaria.

INVENTARE ISTRUZIONI E PROGRAMMI



1) Parte per il programmatore

Inventa delle istruzioni con cui creare tanti nuovi programmi!

Completa il *Manuale delle istruzioni*:

ISTRUZIONI DISPONIBILI	EFFETTO

CLASSE NOME: DATA:

E ora scrivi qualche programma da far eseguire a un compagno o una compagna:

Lo ha eseguito in maniera corretta? ☐ Sì ☐ No



2) Parte per l'esecutore

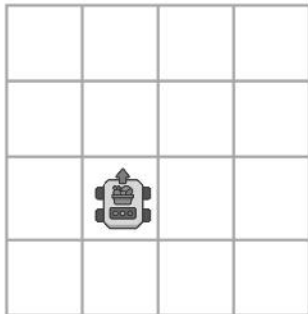
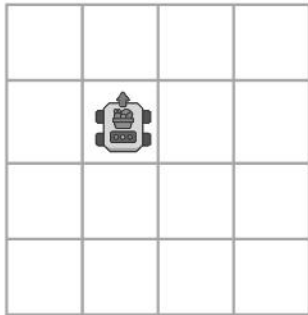
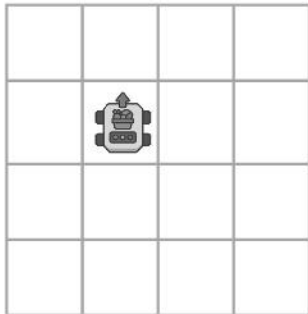
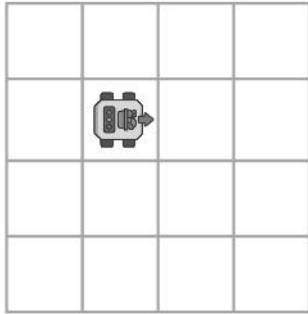
Quali istruzioni non erano chiare?

3) Parte da fare insieme

Modificate il *Manuale delle istruzioni* in modo da rendere la descrizione dell'effetto di ciascuna istruzione più chiara.

MANUALE DELLE ISTRUZIONI

Completa la tabella.

Istruzione	Cosa fa il personaggio?	Disegnalo sulla griglia
vai avanti di un passo	Si sposta di una casella nella direzione in cui "sta guardando" (è orientato)	- prima  - dopo 
gira a destra di un angolo retto		- prima  - dopo 

CLASSE NOME: DATA:

Istruzione	Cosa fa il personaggio?	Disegnalo sulla griglia																																
gira a sinistra di un angolo retto		• prima <table><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr></table> • dopo <table><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr></table>																																
		• prima <table><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr></table> • dopo <table><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr></table>																																
...	...	...																																

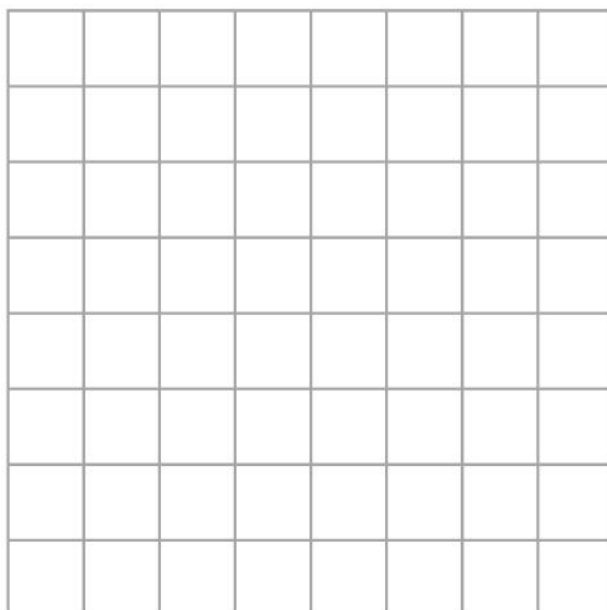
CLASSE NOME: DATA:

PROGRAMMA UNA MISSIONE!

Personaggio: _____

Missione: _____

Disegna nella griglia la situazione iniziale, l'obiettivo e, con la matita, la strada che vuoi far fare al personaggio. Poi completa la tabella.



ISTRUZIONI DISPONIBILI	PROGRAMMA PER COMPIERE LA MISSIONE

CAPIRE IL SISTEMA DI CODIFICA

1) Analizzare

Ecco due immagini: a sinistra c'è una griglia con una specie di L gialla; a destra una griglia con una specie di "farfallina" grigia. Di fianco a ciascuna immagine ci sono dei simboli che rappresentano le regole di colorazione:

	1	2	
1			(1,1) GO (1,2) BO
2			(2,1) GO (2,2) GO

	1	2	
1			(1,1) BO (1,2) GO
2			(2,1) GO (2,2) BO

Per scrivere l'immagine in simboli, da dove si parte?
E in che ordine si va?

Cosa significano i numeri tra parentesi?

Cosa significa GO?

CLASSE NOME: DATA:

Cosa significa BO?

Come si potrebbe indicare il colore blu?

2) Completare

Completa la tabella che segue scrivendo i simboli o colorando l'immagine:

	1	2	(,)		1	2	(1,1) GO
1			(,)		1		(1,2) GO
			(,)				(2,1) BO
2			(,)		2		(2,2) GO

Confrontati poi con un compagno o una compagna: avete completato la tabella nello stesso modo?
