Dal pensiero computazionale al CODING

3 DAL PENSIERO COMPUTAZIONALE AL CODING

La scuola e la vita di tutti i giorni ti pongono continue situazioni e problemi da affrontare: trovare delle soluzioni può essere anche divertente. In questo capitolo vedremo che:

- per risolvere i problemi può essere utile l'intuito, ma occorre anche metodo;
- esistono problemi mal formulati, problemi con dati superflui, problemi irrisolvibili e problemi risolvibili ma con tempi di risoluzione enormi;
- la tecnica del *problem solving* è un metodo che probabilmente già applichi;
- ci sono problemi che hanno come soluzione un risultato e altri problemi che hanno come soluzione un algoritmo;
- affrontare un problema con gli strumenti del pensiero computazionale aiuta a trovare una strategia risolutiva.

Vedremo anche che fare delle ipotesi di soluzione e analizzarne i risultati per correggere eventualmente il tiro è una strategia molto utilizzata per risolvere i problemi: è quello che si fa per esempio nei videogiochi!

E osserveremo che anche quando il problema non ha a che fare con l'informatica, «pensare come un computer», o meglio pensare come un informatico, ci può far avvicinare alla soluzione di un problema; *programmare* infatti non significa soltanto scrivere codice per un computer, ma anche organizzare un'attività in modo razionale.

In questo capitolo, infine, giocheremo! Risolvere puzzle e schemi con i personaggi di **Code Studio**, far loro raccontare delle storie, realizzare disegni geometrici fantasiosi saranno modi divertenti ed efficaci di capire i meccanismi che sono alla base della programmazione dei computer.

3 DAL PENSIERO COMPUTAZIONALE AL CODING

Problemi e soluzioni

A volte esiste più di un metodo per arrivare alla soluzione di un problema: in certi casi, i metodi sono tra loro equivalenti; in altri, uno è preferibile in termini di rapidità, di efficienza o di eleganza. Per esempio, possono esserci tante strade che portano nello stesso luogo: una può essere preferibile perché più veloce da percorrere, un'altra perché più breve, una terza perché... più panoramica!

Un problema necessita di una strategia risolutiva che parte dall'analisi dei dati a disposizione. Sono sufficienti a risolvere il problema? Ce ne sono di inutili? Ogni problema ben formulato contiene tutti gli elementi necessari per poter progettare un metodo risolutivo, però può contenerne di superflui.

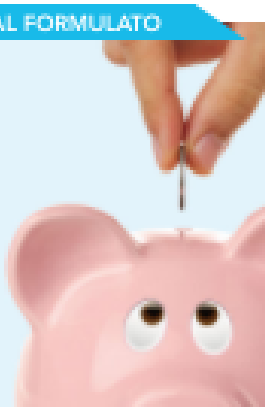
Ma non tutti i problemi sono risolvibili: ne esistono anche di impossibili. E infine ci sono problemi risolvibili, ma la cui risoluzione implicherebbe un tempo enorme di esecuzione: devi saper valutare anche queste possibilità.

ESEMPIO: PROBLEMA MAL FORMULATO

Un bambino ogni giorno mette una moneta nel salvadanaio.

Quanti euro ha dopo dieci giorni?

Se non si conosce il valore delle monete e quanti euro conteneva all'inizio, non sarà possibile calcolare il contenuto finale del salvadanaio: il problema è mal formulato.



ESEMPIO: PROBLEMA CON DATI SUPERFLUI

Una torta ha una base rotonda di superficie pari a 72 cm^2, è alta 5 cm e pesa 2 kg. Le decorazioni su di essa sono alte 8 cm.

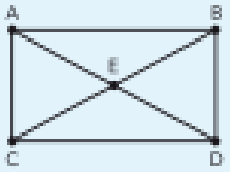
Quali sono le dimensioni minime della scatola con base di forma quadrata che la deve contenere?

È necessario calcolare il diametro della torta: il lato della scatola dovrà essere almeno di questa misura. L'altezza minima è data dalla somma dell'altezza della torta e delle decorazioni. Il dato relativo al peso della torta, in questo caso, è superfluo.

ESEMPIO: PROBLEMA IMPOSSIBILE

Da quale vertice devi partire per disegnare la figura a destra senza mai staccare la matita dal foglio e senza passare due volte sopra lo stesso segmento?

Come dimostrò il matematico svizzero Eulero nel 1736, si possono disegnare figure di questo tipo alle condizioni date soltanto quando ogni nodo è collegato a un numero pari di nodi (come per il nodo E, che è collegato ad A, B, C e D) oppure se soltanto due nodi della figura sono collegati a un numero dispari di nodi. Nella nostra figura, quattro nodi (A, B, C e D) sono collegati ciascuno a tre altri nodi: il problema è dunque impossibile (cioè non ha una soluzione), ma puoi divertirti a proporlo ai tuoi amici!



ESEMPIO: PROBLEMA RISOLVIBILE, MA IN UN TEMPO ENORME

Come puoi fare per scoprire la combinazione da 8 cifre di una cassaforte?

Per aprire una cassaforte a combinazione numerica di 8 cifre si deve individuare una sequenza del tipo 73016449: indovinarla con pochi tentativi senza avere indizi è impossibile. Potremmo allora provare con tutte le sequenze di cifre pensabili, iniziando da 00000000, poi 00000001 e così via fino a 99999999: sicuramente arriveremmo al risultato. Ma anche supponendo di essere in grado di eseguire un tentativo al secondo, occorrerebbero più di 1000 giorni per provare tutte le possibilità! Il problema è quindi risolvibile in teoria, ma richiede un tempo eccessivo.

Il *problem solving*

Per affrontare con sistematicità un problema di qualsiasi tipo, può essere molto utile applicare un metodo conosciuto come *problem solving*, il cui impiego prevede la ripetizione di specifici passi fino ad arrivare alla soluzione del problema:

Come vedi dalla figura, il *problem solving* è un'attività ciclica, che dopo il passo 5 prevede il ritorno al passo 1 per definire nuovamente il problema alla luce dei risultati ottenuti. Si applica di nuovo il metodo in due casi:

- per ottenere una soluzione valida, quando la precedente non lo sia;
- per valutare l'esistenza di una soluzione migliore della precedente.

Quando cerchi di risolvere un problema puoi arrivare a una soluzione errata, che nel *problem solving* non è inutile, anzi: l'errore ha un valore positivo, perché le soluzioni non corrette possono aiutare a trovare quelle giuste e a restringere il campo delle possibilità. La fase successiva del processo risolutivo di un problema può partire dall'analisi del procedimento che ti ha portato a una conclusione errata.

L'attività tipica del *problem solving* è applicata, a volte inconsapevolmente, quando si cerca una soluzione a un problema complesso, quando si risolve un indovinello e, spesso, quando si lavora in gruppo. Davanti a un problema non arrenderti mai: studialo, cerca una soluzione, applicala e poi controlla il risultato; se non è corretta, il procedimento impiegato può esserti comunque utile per trovare la soluzione giusta. Così fanno anche i grandi scienziati!

HELP iCub

Per far sì che un robot riesca ad afferrare degli oggetti senza romperli o farli cadere, i ricercatori adottano la tecnica del *problem solving*: valutano se il robot afferra l'oggetto nel modo corretto, se usa troppa forza o troppo poca, se riesce a valutarne peso e dimensioni; poi utilizzano i dati ricavati per cercare nuove soluzioni e migliorare il sistema ideato in precedenza.

iCub, sviluppato dall'Istituto italiano di tecnologia di Genova.

ADESSO TOCCA A TE!

Un atleta, un ragazzo, un bambino e un anziano vogliono attraversare un ponte che può sopportare al massimo il peso di due persone alla volta. È notte, per attraversare occorre usare una lampada e loro ne hanno una sola. L'atleta può attraversare in 1 minuto, il ragazzo in 2, il bambino in 5 e l'anziano in 10.

Qual è il tempo minimo che possono impiegare per raggiungere tutti l'altra sponda?

In gruppo con altri compagni prova a risolvere il problema. Vedrai che arriverete prima a soluzioni errate, poi a soluzioni valide ma non ottimali e infine, applicando in modo naturale il metodo del *problem solving*, alla soluzione corretta, che è... 17 minuti! Non ci credi? Prova!

3 DAL PENSIERO COMPUTAZIONALE AL CODING

Soluzioni: risultati e procedimenti

Dal punto di vista della loro soluzione, esistono due tipi di problemi.

1. **I problemi per i quali la soluzione, se esiste, è una risposta (o un *risultato*).**
   In questo tipo di problemi la soluzione è un dato, non necessariamente di tipo numerico: può essere anche un insieme di numeri, una parola o una frase, un percorso, un colore e così via, ma è comunque un **risultato**, oppure la constatazione che il problema è irrisolvibile o mal formulato.

ESEMPI:

- **Quanti litri di vernice occorrono per una parete di 4x3 metri, se ogni litro copre 2 metri quadri?**
- **Nel labirinto qui a fianco, quale percorso deve effettuare l'uomo per arrivare al furgone?**

- **Che colore ottieni mescolando uguali quantità di pittura gialla e blu?**
- **Qual è la parola palindroma (ossia che risulta identica anche se letta da destra a sinistra) più lunga che riesci a trovare?**
- **Quante monete puoi disporre sul fondo di una scatola quadrata di 10 cm di lato senza che si sovrappongano?**

- Se tre ragazzi salgono su una bilancia, questa segna 120 kg. Se ne scende uno, la bilancia segna 80 kg; se lui risale e ne scende un altro, la bilancia segna 82 kg.

  Quanto pesano i tre ragazzi?

[Soluzione: siano A, B e C i pesi dei tre ragazzi; la somma è 120.
Se A + B + C = 120 e A + B = 80, vuol dire che C = 120 − 80 = 40.
Se A + B + C = 120 e A + C = 82, vuol dire che B = 120 − 82 = 38
Se A + B + C = 120, B = 38 e C = 40, vuol dire che A = 120 − 38 − 40 = 42
La soluzione è A = 42, B = 38, C = 40]

2. **I problemi per i quali la soluzione, se esiste, è un *procedimento*.**
   Le soluzioni a problemi di questo tipo possono essere un metodo, una serie di confronti, una sequenza di azioni condizionate da particolari eventi; in generale la soluzione non è costituita da un semplice risultato ma da un **procedimento**, oppure dalla constatazione che il problema è irrisolvibile o mal formulato.

ADESSO TOCCA A TE!

Con i tuoi compagni cerca le soluzioni a tutti questi problemi e scrivi, se esiste, il procedimento risolutivo.

ESEMPI

- **Come puoi moltiplicare due numeri interi utilizzando soltanto operazioni di somma?**
- **Come puoi scoprire se un numero è primo, cioè divisibile soltanto per 1 e per se stesso?**
- Hai 27 palline simili, ma una pesa leggermente di più delle altre.

  Come puoi individuarla con tre pesate su una bilancia a bracci?

Markus Gann/Shutterstock

- Davanti a te hai tre interruttori posizionati su spento e una porta chiusa. Solo uno dei tre interruttori accende la lampadina che c'è nella stanza oltre la porta. Non puoi premere gli interruttori mentre la porta è aperta, e una volta entrato non puoi tornare indietro.

  Come fai a capire quale interruttore accende la lampadina?

Stocky Marmary/Shutterstock

[Soluzione: non toccare il primo interruttore; attiva il secondo interruttore per circa un minuto, poi disattivalo, attiva il terzo ed entra subito nella stanza. Tocca la lampadina: se è spenta e fredda la soluzione è il primo interruttore, se è spenta ma calda è il secondo, se è accesa è il terzo]

Problemi e algoritmi

Abbiamo visto che ci sono problemi la cui soluzione non è un risultato ma un procedimento. La soluzione al problema, cioè, non è una risposta, ma *un metodo per trovare una risposta* quando la situazione reale si presenterà. Quindi il metodo deve funzionare ogni volta che la situazione si presenta, al variare delle possibili condizioni. Questo significa che utilizzando il medesimo procedimento avremo in generale risultati diversi al presentarsi di condizioni iniziali diverse.

ESEMPIO: PROBLEMA LA CUI SOLUZIONE È UN PROCEDIMENTO

Ho una sola sfera di metallo; so che se la lascio cadere dal primo piano di un grattacielo non si romperà, ma so che si romperà se la lancio dall'ultimo.

Qual è il piano più basso dal quale la sfera lanciata si romperà?

Non ho gli elementi per ricavare una risposta, quindi non posso sapere qual è la soluzione, ma posso fornire un metodo per trovarla.
So che la sfera non si rompe **se** la lascio cadere dal primo piano; **quindi** inizio dal secondo piano. Ogni volta lancio la sfera dal piano in cui mi trovo: **se si rompe**, ho trovato la soluzione; **se non si rompe**, salgo di un piano e riprovo.

Come vedi, questo non è un *risultato* ma un *procedimento*: lo posso applicare senza modificarlo lanciando oggetti diversi, e ogni volta potrò ottenere risultati diversi. I procedimenti sono chiamati **algoritmi** dagli informatici.

Un algoritmo è una sequenza finita di istruzioni chiare, eseguibili, non ambigue, la cui esecuzione risolve un problema. Può essere scritto in italiano (come quello precedente) o in una qualsiasi altra forma che ne garantisca la chiarezza. Le singole istruzioni che compongono un algoritmo ben descritto devono essere:

- **chiare**: chi sta risolvendo il problema deve capire ogni singolo passo;
- **eseguibili**: chi sta risolvendo il problema deve poter eseguire ogni singolo passo;
- **non ambigue**: per ogni singolo passo ci deve essere univocità di interpretazione.

RIPASSIAMO INSIEME

Indica con una crocetta la risposta corretta.

1 **La soluzione di un problema risolvibile è:**

- [A] un risultato.
- [B] un algoritmo.
- [C] un risultato o un algoritmo.

2 **Nel *problem solving* se arrivi a una soluzione errata:**

- [A] ti fermi.
- [B] continui sfruttando anche la soluzione errata.
- [C] vuol dire che il problema è impossibile.

3 **Se un problema ha dati superflui:**

- [A] può essere ugualmente risolto.
- [B] non può essere risolto.
- [C] ha sempre come risultato un algoritmo.

4 **Un'istruzione è ambigua se:**

- [A] può essere eseguita in un solo modo.
- [B] può essere eseguita in più di un modo.
- [C] non è eseguibile.

3 DAL PENSIERO COMPUTAZIONALE AL CODING

ESEMPIO: ALGORITMO PER IL CALCOLO APPROSSIMATIVO DI π (PI GRECO)

1. Prendi un barattolo cilindrico (per esempio una lattina);
2. prendi un nastro (o una striscia di carta);
3. arrotola il nastro intorno al barattolo e segna il punto dove i due lembi si sovrappongono;
4. stendi il nastro e misuralo con il righello; è la circonferenza: segna il valore su un foglio;
5. poni il barattolo tra due libri;
6. misura la distanza tra i due libri; è il diametro del barattolo: segna il valore su un foglio;
7. dividi la misura della circonferenza per la misura del diametro;
8. ciò che ottieni è il valore (approssimato) di π.

Prova a eseguire questo algoritmo e rispondi alle seguenti domande.

- **Tutte le istruzioni sono chiare? Capisci che cosa devi fare a ogni passo?**
- **Tutte le istruzioni sono eseguibili? Sei in grado di portarle a termine?**
- **Le istruzioni sono tutte non ambigue? C'è il rischio che un'istruzione possa essere eseguita in modi diversi?**

ADESSO TOCCA A TE!

Quando fai eseguire un compito a un computer o a un robot, questi non sa (e non può sapere) che cosa vuoi ottenere. Prova a fare lo stesso con qualcuno che conosci, cioè fagli eseguire la sequenza di azioni che portano a ricavare π senza spiegargli qual è lo scopo e, se non ottieni il risultato voluto, correggi l'algoritmo adattandolo all'esecutore.

Per descrivere un algoritmo puoi usare lo strumento che preferisci; il più semplice è il linguaggio naturale, cioè la lingua parlata: nel tuo caso l'italiano. Ricorda però che un buon algoritmo deve avere istruzioni *chiare*, *eseguibili* e *non ambigue*.

ADESSO TOCCA A TE!

Per questa esperienza dovete essere almeno in tre e dare sfogo alla vostra creatività:

1. il primo compagno dovrà disegnare un percorso, come una pista, su carta a quadretti, con un punto di partenza e uno di arrivo, ostacoli (che danno punteggio negativo) e premi (che danno punteggio positivo);
2. il secondo compagno dovrà scrivere un algoritmo indicando le istruzioni per arrivare dalla partenza all'arrivo, senza mai far staccare la penna dal foglio e facendo accumulare il punteggio massimo possibile;
3. il terzo compagno dovrà seguire le istruzioni contenute nell'algoritmo per segnare il percorso su un foglio bianco, ma senza mai vedere il percorso originale.

Alla fine i fogli saranno sovrapposti e si dovrà valutare la bontà dell'algoritmo fornito in termini di correttezza (Quante volte si è «usciti di strada»? È stato raggiunto il traguardo?), numero di istruzioni fornite e punteggio ottenuto.

Il pensiero computazionale

L'informatica ha introdotto non solo nuovi strumenti (computer, *smartphone*, *tablet*, navigatori satellitari e così via), ma anche nuovi metodi per affrontare i problemi: la mentalità con la quale si affrontano i problemi razionalizzandone il processo risolutivo è detta **pensiero computazionale**.

Affrontare un problema in questo modo non vuol dire che tu lo debba risolvere impiegando un computer: il pensiero computazionale è una capacità dell'essere umano; il computer è solamente uno degli strumenti a sua disposizione.

D'altra parte, usare un computer può aiutarti a risolvere un problema che per un essere umano potrebbe essere troppo lungo, noioso o ripetitivo. L'uso di un robot, che è un tipo di computer che interagisce con l'ambiente esterno, permette di risolvere problemi stressanti, che necessitano di estrema precisione o che risultano pericolosi o impossibili per un essere umano.

foto NASA

Curiosity, il robot della NASA atterrato su Marte nel 2012 alla ricerca di possibili forme di vita. L'autoscatto è stato realizzato componendo diverse foto scattate sul pianeta con il suo braccio telescopico.

In generale, affrontare un problema mediante il pensiero computazionale significa:

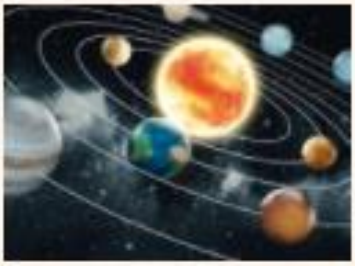
Analizzare	1. cogliere gli aspetti importanti del problema distinguendoli da quelli superflui; 2. individuare le risorse necessarie alla sua risoluzione;
Schematizzare	3. rappresentare il problema in maniera schematica;
Scomporre	4. suddividere il problema in una serie di sottoproblemi più semplici da risolvere;
Progettare	5. ideare la successione di passi elementari che risolvono ciascuno dei sottoproblemi;
Verificare	6. esaminare la soluzione trovata in termini di correttezza ed efficienza*;
Generalizzare	7. fare in modo che la soluzione trovata sia applicabile a una vasta gamma di problemi dello stesso tipo.

HELP Robot

Un robot non è altro che un computer dotato di sensori (audio, video, di luminosità, forza, posizione) che gli consentono di percepire l'ambiente esterno, e di attuatori (come motori, riproduttori audio, pinze) che gli consentono di interagire con esso.

HELP Telecomandare Curiosity?

Perché Curiosity è un robot che prende decisioni in autonomia e non è invece telecomandato dalla Terra? Perché per inviare segnali radio dalla Terra a Marte occorrono da un minimo di 3 a un massimo di 22 minuti. Il robot è quindi programmato per essere autonomo, in particolar modo per quanto riguarda i suoi spostamenti.



Secondo te, come mai il tempo che occorre per inviare dei segnali tra i due pianeti non è costante?

* Spesso esistono più algoritmi che risolvono correttamente il medesimo problema; un algoritmo è più efficiente di un altro se a parità di esecutore impiega un tempo inferiore.

ESEMPI

Un giardino di forma rettangolare ha un lato di 5 metri.
La recinzione che lo racchiude è lunga 30 metri e alta 1 metro.

Qual è la superficie del giardino?

1. Analizzare: cogliere gli aspetti fondamentali del problema distinguendoli da quelli superflui.

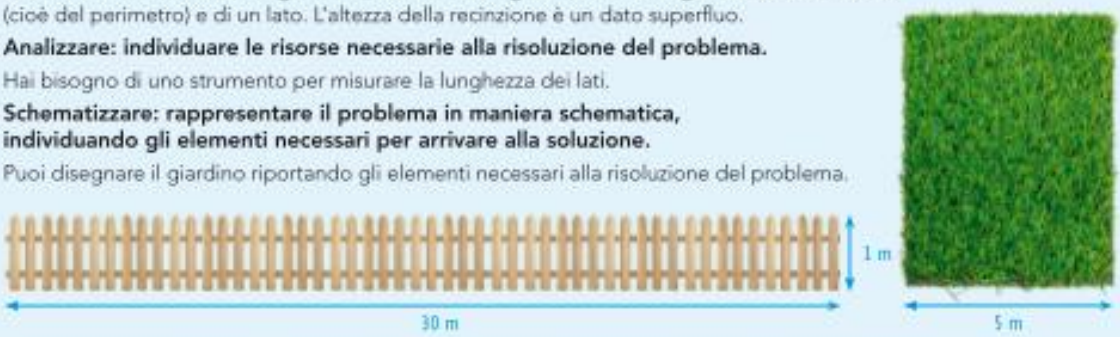
Si tratta di ricavare l'area del giardino (di forma rettangolare) data la lunghezza della recinzione (cioè del perimetro) e di un lato. L'altezza della recinzione è un dato superfluo.

2. Analizzare: individuare le risorse necessarie alla risoluzione del problema.

Hai bisogno di uno strumento per misurare la lunghezza dei lati.

3. Schematizzare: rappresentare il problema in maniera schematica, individuando gli elementi necessari per arrivare alla soluzione.

Puoi disegnare il giardino riportando gli elementi necessari alla risoluzione del problema.



4. Scomporre: suddividere il problema in una serie di sottoproblemi più semplici da risolvere.

Per trovare l'area del rettangolo devi prima trovare il secondo lato; per trovare il secondo lato devi prima trovare il semiperimetro.

5. Progettare: ideare la successione di passi elementari che risolvono ciascuno dei sottoproblemi.

- Se il perimetro di un rettangolo è 30 metri, vuol dire che la somma dei due lati diversi (il semiperimetro) è la metà, 15 metri.
- Se il primo lato è 5, l'altro sarà di 15 – 5 = 10 metri.
- La superficie sarà quindi 10 x 5 = 50 metri quadri; schematizzando:

1. semiperimetro: 30 m / 2 = 15 m
2. $lato_2$: 15 m – 5 m = 10 m
3. area: 10 m x 5 m = 50 m^2

6. Verificare: esaminare la soluzione trovata in termini di correttezza ed efficienza.

Per valutare la correttezza del metodo utilizzato, in questo caso, puoi effettuare la seguente verifica: sommando i quattro lati (10 m + 10 m + 5 m + 5 m) ottieni la lunghezza delle recinzione; il risultato è corretto.
Per valutare l'efficienza devi porti le domande: «*Potevo ottenere lo stesso risultato in un altro modo? L'eventuale altro metodo prevedeva un numero inferiore di passaggi?*».

7. Generalizzare: fare in modo che la soluzione trovata sia applicabile a una vasta gamma di problemi dello stesso tipo.

Hai risolto lo specifico problema, ma il metodo impiegato è valido esclusivamente per rettangoli di perimetro 30 metri e con un lato di 5 metri. Generalizzare la soluzione di questo problema significa fare in modo che il procedimento usato ti consenta di calcolare la superficie di un qualsiasi rettangolo, dati il perimetro e un solo lato.

Soluzione del problema	Generalizzazione
1. *semiperimetro* = 30 m / 2 = 15 m 2. $lato_2$ = 15 m – 5 m = 10 m 3. *area* = 10 m x 5 m = 50 m^2	1. *semiperimetro* = ***perimetro*** / 2 2. $lato_2$ = *semiperimetro* – $\boldsymbol{lato_1}$ 3. *area* = $\boldsymbol{lato_1}$ × $lato_2$

Puoi applicare il metodo generalizzato per trovare l'area di un qualsiasi rettangolo, dati il perimetro e un lato, sostituendo i due valori ai nomi in grassetto nel procedimento a destra.

Generalizzare un processo risolutivo

La generalizzazione della soluzione di un problema è una fase fondamentale nell'ambito del pensiero computazionale, perché ti consente di applicare lo stesso procedimento a una gamma più ampia di problemi.

Una volta trovata la soluzione a un problema, devi sempre chiederti se è possibile modificarla affinché possa risolvere altri problemi dello stesso tipo.

Progettare la successione di passi elementari che risolvono il problema

Uno degli aspetti fondamentali del pensiero computazionale consiste nel progettare la successione di **passi elementari** che risolvono il problema.

I passi elementari sono le singole istruzioni che l'esecutore (un computer, un cuoco, una persona cui stai fornendo l'algoritmo eccetera) è in grado di eseguire; il metodo con il quale componi tra loro le istruzioni è fondamentale affinché esse siano eseguite nel giusto ordine. In generale, l'applicazione di un algoritmo di qualsiasi tipo può prevedere:

- azioni da svolgere in **sequenza**;
- azioni da eseguire in **cicli definiti**, cioè un numero prestabilito di volte;
- azioni da eseguire in **cicli indefiniti**, cioè un numero imprecisato di volte, fino al verificarsi di una certa condizione;
- **scelte** da effettuare in base a determinate condizioni;
- **eventi** che possono causare altri eventi;
- **attività** che possono o devono essere svolte **in contemporanea**.

Questo modo di affrontare alcuni problemi lo applichi già, magari senza farci caso. Per esempio, nella vita di tutti i giorni avrai:

- **sequenze:** *prima* mi spoglio, *poi* indosso il pigiama, *poi* vado a letto;
- **cicli definiti:** *fino a che* non ho percorso 10 giri *continuo a* correre;
- **cicli indefiniti:** *fino a che* non ho capito *continuo a* studiare;
- **scelte:** *se* domani ho il compito in classe *allora* studio, *altrimenti* gioco;
- **eventi:** *quando* sono le cinque *inizio a* preparare la borsa;
- **attività in contemporanea:** faccio merenda *e nel frattempo* ascolto la musica.

ADESSO TOCCA A TE!

La ricetta per una piccola crostata alla marmellata è la seguente: *mescola 100 g di farina, 50 g di zucchero, 50 g di burro e un uovo fino a ottenere un impasto omogeneo. Stendilo in uno stampo imburrato di 10 cm di diametro e ricoprilo con 30 g di marmellata. Inforna a 180 °C fino alla doratura del bordo.* Generalizza questo procedimento per fare in modo che la ricetta sia valida per un numero variabile n di crostate.

ADESSO TOCCA A TE!

Descrivi la tua giornata, la tua settimana, oppure le regole del tuo sport preferito, usando almeno una volta ciascuna delle tipologie di espressioni indicate per programmare le sequenze di passi elementari.

RIPASSIAMO INSIEME

Indica con una crocetta la risposta corretta.

1 **Generalizzare la soluzione di un problema significa:**

- [A] poter applicare lo stesso metodo ad altri problemi.
- [B] cercare una soluzione più precisa.
- [C] cercare un metodo più rapido.

2 **Il pensiero computazionale:**

- [A] può essere usato solo dai programmatori.
- [B] può essere usato soltanto dagli scienziati.
- [C] può essere usato da tutti.

3 **Suddividere un problema in sottoproblemi:**

- [A] ne aumenta la complessità.
- [B] può aiutare a risolverlo.
- [C] aumenta il tempo necessario alla sua risoluzione.

4 **«Tira fino a che non fai 10 canestri» prevede:**

- [A] una sequenza di azioni.
- [B] un ciclo definito di azioni.
- [C] un ciclo indefinito di azioni.

Dal pensiero al programma: Code Studio

Creare programmi per computer è un ottimo metodo per applicare i principi del pensiero computazionale. Il codice in cui si scrive un programma di solito è in linguaggio testuale, ma esistono anche linguaggi di programmazione *visuali* che consentono di creare programmi accostando tra loro blocchi grafici corrispondenti a istruzioni.

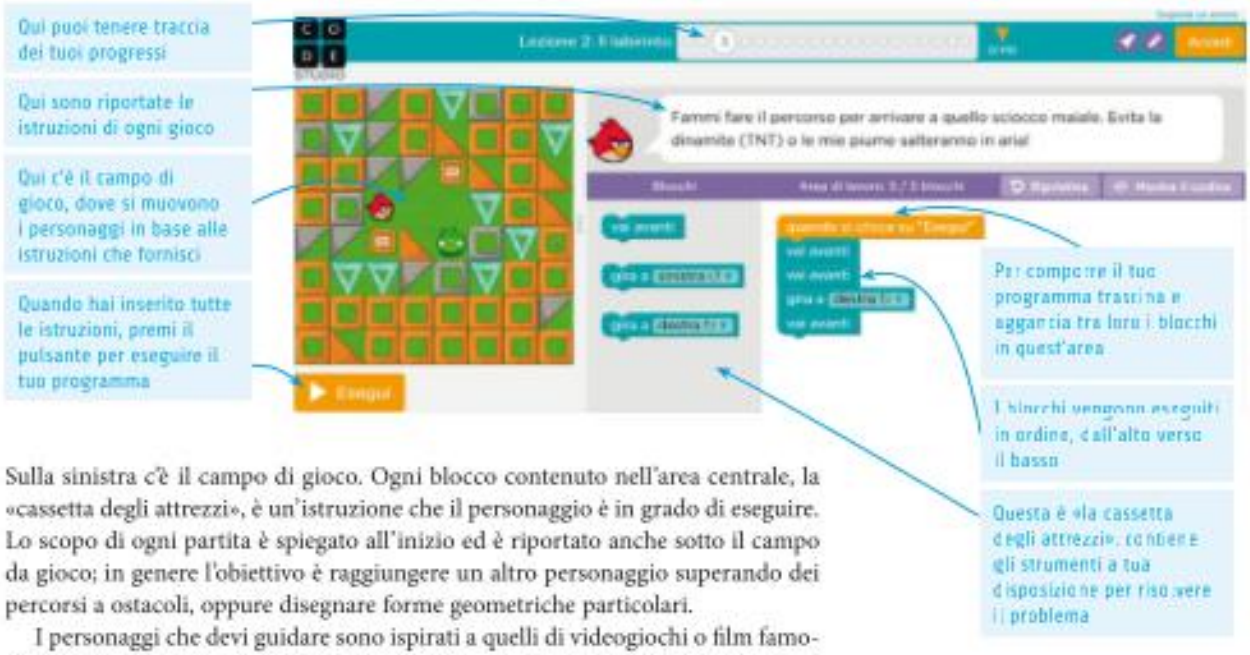
La scrittura di un programma per computer è detta *coding* (dall'inglese *to code*, scrivere in codice).

```
for (var count = 0; count < 4; count++)
 {
  turnLeft();
  moveForward();
}
```

Un blocco di codice testuale (JavaScript)

Il corrispondente codice visuale

Per risolvere i giochi interattivi presenti nell'ambiente **Code Studio** del sito **code.org** si usa il linguaggio visuale **Blockly**. L'ambiente di Code Studio è suddiviso in aree, e numerosi video ti spiegano come usare gli strumenti a tua disposizione per risolvere i labirinti e le altre sfide proposte.



Sulla sinistra c'è il campo di gioco. Ogni blocco contenuto nell'area centrale, la «cassetta degli attrezzi», è un'istruzione che il personaggio è in grado di eseguire. Lo scopo di ogni partita è spiegato all'inizio ed è riportato anche sotto il campo da gioco; in genere l'obiettivo è raggiungere un altro personaggio superando dei percorsi a ostacoli, oppure disegnare forme geometriche particolari.

I personaggi che devi guidare sono ispirati a quelli di videogiochi o film famosi. Non accade niente di negativo se commetti un errore e fai sbattere l'uccellino di *Angry Birds* contro il muro oppure se conduci un personaggio di *Plants vs. Zombies* su una pianta carnivora, o se non fai fare il disegno corretto a Elsa e Anna di *Frozen*. Se sbagli, osserva che cosa succede per risalire all'errore e individuare il comportamento corretto al tentativo successivo.

HELP Programma il futuro

programmailfuturo è il sito creato dal Ministero dell'Istruzione, dell'Università e della Ricerca in collaborazione con il CINI (Consorzio Interuniversitario Nazionale per l'Informatica) per promuovere il pensiero computazionale. Per gli esperimenti online fa riferimento al sito code.org, ma è anche ricco di materiali aggiuntivi.

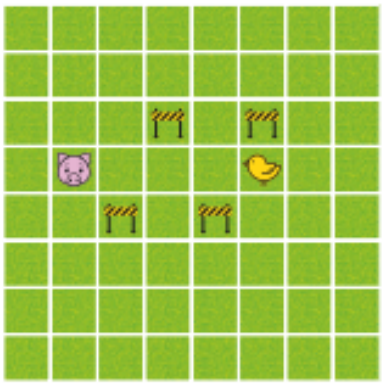
HELP code.org

Prima di iniziare a giocare con code.org effettua l'accesso **creando un tuo *account***: sono sufficienti un indirizzo di posta elettronica e una password. In questo modo il sito terrà traccia dei tuoi progressi.

Le sequenze in Code Studio

Le azioni descritte negli algoritmi visti finora sono – e devono essere – eseguite in **sequenza**: solamente se eseguite nell'ordine descritto portano al risultato corretto. La sequenza di azioni è alla base di ogni algoritmo.

Per esempio, per risolvere il seguente labirinto, cioè per portare l'uccellino sulla stessa casella del maialino facendogli evitare le caselle ostacolo, quale tra le due sequenze di azioni proposte bisogna eseguire?

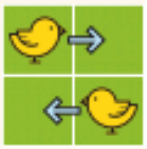


1
- Vai avanti
- Vai avanti
- Vai avanti

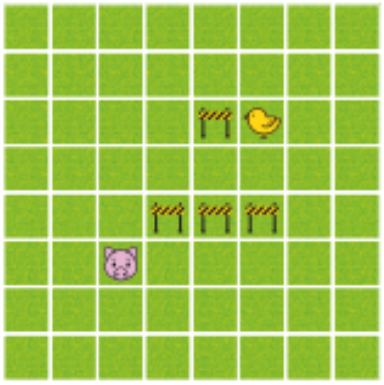
2
- Vai avanti
- Vai avanti
- Vai avanti
- Vai avanti

HELP Movimento

I personaggi di Code Studio si muovono nella direzione verso la quale sono girati.



Le sequenze possono essere composte da blocchi di differente tipologia, eseguiti in un determinato ordine affinché possano risolvere un determinato problema. Che sequenza di azioni devi eseguire tra le due proposte a fianco per portare l'uccellino sulla casella del maialino?



1
- Gira a sinistra di 90°
- Vai avanti
- Gira a destra di 90°
- Vai avanti
- Vai avanti
- Vai avanti
- Gira a sinistra di 90°
- Vai avanti
- Vai avanti

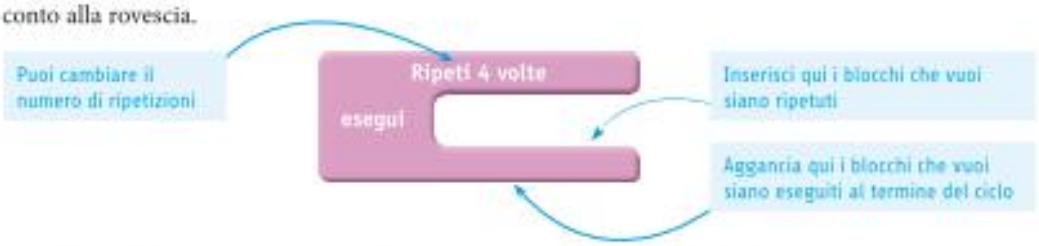
2
- Gira a sinistra di 90°
- Vai avanti
- Gira a destra di 90°
- Vai avanti
- Vai avanti
- Gira a sinistra di 90°
- Vai avanti
- Vai avanti
- Vai avanti

Anche per risolvere uno qualsiasi dei labirinti in Code Studio devi agganciare i blocchi appropriati nel corretto ordine. Se commetti un errore, il sistema visualizza il messaggio per aiutarti a capire come correggere il programma; dopo puoi ripartire premendo il tasto Riprova.

Se invece il tuo programma è corretto, appare il messaggio di conferma e puoi proseguire con lo schema successivo.

I cicli definiti in Code Studio

Le attività che ripeti per un numero prestabilito di volte si definiscono **cicli definiti**. Puoi usare un ciclo definito se devi porre una domanda a ciascuno dei tuoi compagni di classe, se devi percorrere una pista dieci volte, se stai facendo un conto alla rovescia.



Un ciclo definito ti permette di semplificare una sequenza di istruzioni che contiene al suo interno delle ripetizioni, e di utilizzare un numero inferiore di blocchi. Per esempio, invece di usare per cinque volte il blocco Vai avanti, puoi inserirne uno in un ciclo definito al quale puoi specificare di ripetere cinque volte i comandi al suo interno: il risultato finale sarà lo stesso.

Puoi inserire più blocchi all'interno del blocco Ripeti n volte: saranno eseguiti in ordine, dall'alto verso il basso, il numero di volte specificato. Nel labirinto qui sotto, quale ti sembra il modo più semplice per portare l'uccellino sulla casella del maialino?

1

2

I cicli indefiniti in Code Studio

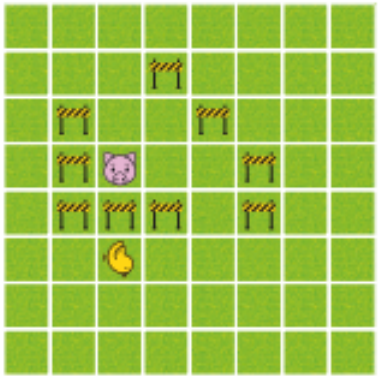
Quando devi ripetere un'azione o un gruppo di azioni, ma non sai quante volte, devi usare i **cicli indefiniti**. Semplici esempi di cicli indefiniti di azioni sono:

fino a che non raggiungi 100 euro *continua a* mettere una moneta nel salvadanaio
fino a che non raggiungi il traguardo *continua a* correre.

Ogni ciclo indefinito deve avere una condizione di uscita (*fino a che…*). Le espressioni usate in un algoritmo corretto non possono essere ambigue; anche la condizione di uscita da un ciclo indefinito deve rispettare questa regola.

Espressioni non ammesse	Espressioni ammesse
Fino a che non hai abbastanza denaro…	*Fino a che* non hai almeno 100 euro…
Fino a che non sei vicino al maialino…	*Fino a che* la distanza dal maialino non è inferiore a 5 cm…

Quale tra i seguenti programmi è corretto, cioè porta l'uccellino a raggiungere il maialino? Quale programma preferisci, e perché?



1

2

ADESSO TOCCA A TE!

Accedi a code.org con il tuo account, poi completa l'Ora del Codice *Labirinto Classico*.

RIPASSIAMO INSIEME

Indica con una crocetta la risposta corretta.

1 **Per compiere un percorso rettilineo nel labirinto:**

- [A] si può usare soltanto una sequenza di vai avanti.
- [B] si possono usare i blocchi ripeti e vai avanti.
- [C] si può usare soltanto un ciclo indefinito.

2 **Un ciclo definito viene eseguito:**

- [A] almeno una volta.
- [B] non si sa quante volte.
- [C] il numero di volte riportato nella condizione.

3 **Per compiere un percorso a "U" nel labirinto:**

- [A] si può anche usare una sequenza di blocchi.
- [B] si deve usare almeno un blocco ripeti.
- [C] occorrono almeno due blocchi gira.

4 **Un ciclo indefinito viene eseguito:**

- [A] almeno una volta.
- [B] non si sa quante volte.
- [C] il numero di volte riportato nella condizione.

Le scelte in Code Studio

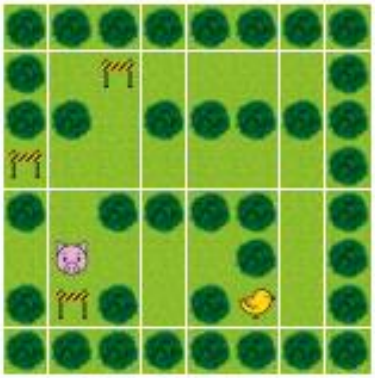
La possibilità di effettuare una **scelta** è alla base della capacità di un computer di eseguire compiti diversi in base a determinate condizioni. In realtà sei tu che devi prevedere i casi possibili e saper dare le opportune istruzioni in modo che il computer «faccia la scelta giusta» quando la situazione si presenta. Ci sono due tipi di blocchi scelta: *semplice* e *con alternativa*. Nel caso di scelta *semplice* devi specificare che cosa fare se si verifica una determinata condizione:

HELP Scelte semplici e con alternativa

Un esempio di scelta semplice è:
se al mercato hanno il mango
allora lo compro.
Una scelta con alternativa è:
se al mercato hanno il mango
allora lo compro
altrimenti compro le mele.

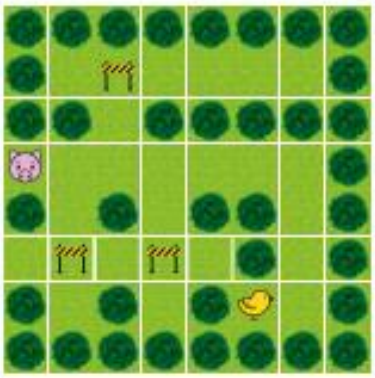
Nel caso di scelta *con alternativa* devi specificare che cosa fare se si verifica una determinata condizione e che cosa fare in caso contrario:

Quali blocchi portano l'uccellino a raggiungere il maialino in questi labirinti?



1

2



1

2

Gli eventi in Code Studio

Quando un particolare episodio determina l'avvio di una specifica sequenza di azioni si parla di **evento**. Se hai già fatto qualche esercitazione in Code Studio hai già visto un esempio di evento: è il blocco quando si clicca su Esegui al quale hai agganciato i programmi che hai creato finora. In tutti questi casi, quando hai fatto clic sul pulsante Esegui il programma è andato in esecuzione. Nei videogiochi, quando premi un pulsante, o quando fai clic, o quando due oggetti si toccano, spesso accade qualcosa: anche questi sono eventi.

Altri eventi tipici sono:

- quando premi il tasto Invio…
- quando il personaggio colpisce il bordo…
- quando raggiungi 1000 punti…

Sei tu che decidi che cosa deve accadere quando si verifica un determinato evento; in Code Studio puoi farlo agganciando delle sequenze di codice ai **blocchi evento**. Sono di colore verde e non possono essere agganciati ad altri blocchi: puoi solamente agganciare dei blocchi sotto di essi.

La contemporaneità in Code Studio

Mentre è in esecuzione il codice relativo a un evento, un altro evento può dare il via a un diverso gruppo di istruzioni da eseguire **in contemporanea**. In un videogame con due o più giocatori gli eventi relativi al gioco avvengono sempre in contemporanea: nessun giocatore deve attendere che un altro termini un'azione per poter giocare.

Ci sono anche sezioni di codice che devono essere eseguite costantemente durante l'esecuzione di un programma; pensa a un videogioco nel quale devi evitare degli oggetti che si muovono: il codice relativo al movimento degli oggetti sarà eseguito sempre e indipendentemente dalle tue azioni.

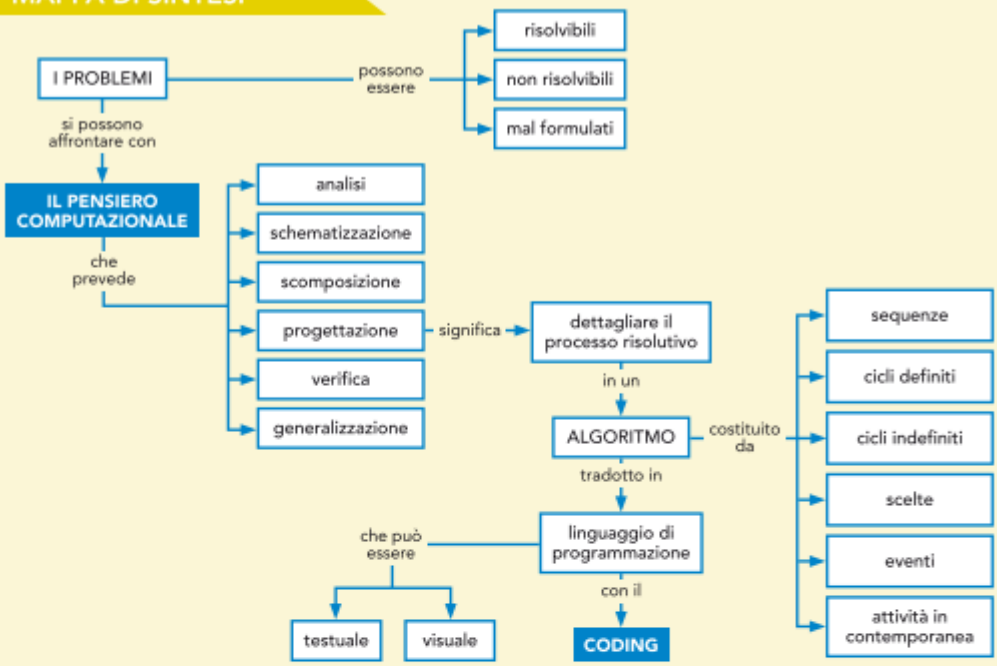
In Code Studio il blocco Ripeti per sempre consente di ripetere costantemente una o più azioni:

In Code Studio puoi combinare dei blocchi da eseguire «per sempre» (per esempio, per far muovere per sempre un personaggio lungo il bordo del campo di gioco) con altri blocchi da eseguire in seguito a un evento (per esempio, per far muovere un personaggio quando premi le frecce in alto o in basso). In questo modo puoi creare avventure o giochi animati.

ADESSO TOCCA A TE!

Accedi a code.org con il tuo account, poi completa l'Ora del Codice *Disney Infinity*.

MAPPA DI SINTESI



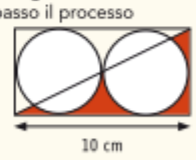
QUESITI

1 Hai una lista di 20 parole; descrivi un procedimento per disporle in ordine alfabetico. Confrontati con i tuoi compagni: quanti metodi diversi avete trovato? Valutate quali procedimenti sono preferibili e perché.

2 Quanto vale la somma di tutti i numeri da 1 a 10? Generalizza la soluzione: dato il numero N, quanto vale la somma dei numeri da 1 a N? Per aiutarti usa la figura a fianco, in cui N = 4.

3 Per riempire una vasca usando l'acqua del pozzo occorre 1 ora. Prelevando anche l'acqua da casa, si impiegano 40 minuti. Quanto tempo serve se usi solo l'acqua di casa? Descrivi il processo logico che hai seguito. (Suggerimento: se con l'acqua del giardino la vasca si riempie in 1 ora, in un minuto si riempie 1/60 di vasca.)

4 Nella seguente figura, due circonferenze tangenti tra loro sono inscritte in un rettangolo con un lato di 10 cm. Descrivi passo per passo il processo logico che devi usare per stimare l'area della sezione in rosso. Poi generalizza il procedimento, scrivendo una formula valida per ogni valore l del lato.



5 A partire da una misura di tempo espressa in secondi (per esempio 4445 secondi) trova l'equivalente in ore e in minuti. Descrivi il procedimento che hai usato per arrivare al risultato richiesto. Utilizza i costrutti che ritieni opportuni (scelta, ciclo indefinito, ...) per descrivere l'algoritmo risolutivo; generalizza il procedimento in modo da renderlo applicabile a una misura in secondi qualsiasi.