UNITÀ DIDATTICA 1

CONOSCIAMO GLI ALGORITMI E I LINGUAGGI

IN QUESTA UNITÀ IMPAREREMO...

- che cos'è un problema e come affrontarlo
- che cosa sono un algoritmo e un programma
- che cosa sono un linguaggio di programmazione e un linguaggio macchina

Il calcolatore, i problemi, i programmi e i linguaggi di programmazione

Come abbiamo già detto, il PC è una macchina complessa in grado di eseguire milioni di istruzioni al secondo, dotata di una memoria capace di contenere enormi quantità di dati, siano essi documenti, suoni, filmati o semplici numeri.

Queste sue caratteristiche ci consentono di utilizzarlo per svolgere compiti molto complicati, anche se, per far sì che ciò sia possibile, dobbiamo indicargli le operazioni da compiere attraverso una serie di **istruzioni** (**programma**) che consentono alla macchina di elaborare i **dati** in maniera appropriata.

Il compito che deve essere svolto dal calcolare è generalmente la soluzione di un **problema**, che può essere di diversa natura: matematica, scientifica, economica, finanziaria ecc.

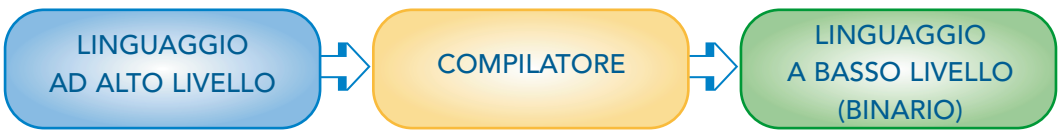
Il programmatore o, meglio, l'**analista**, studia il problema e ne individua un **algoritmo risolutivo**, cioè l'insieme delle operazioni (**istruzioni**) che devono essere eseguite per raggiungere il risultato.

L'insieme di queste istruzioni prende il nome di **programma**, e ogni singola istruzione deve essere scritta secondo rigorose regole (**sintassi**) affinché l'elaboratore possa comprenderla correttamente.

Il programma, dunque, viene scritto in un **linguaggio di programmazione** costituito da un insieme di regole sintattiche che forniscono al calcolatore le istruzioni necessarie per portare a termine un determinato compito.

Il linguaggio di programmazione usato dal programmatore (noto anche come **linguaggio ad alto livello**) è, di fatto, molto lontano dal codice binario (**linguaggio a basso livello**), cioè dal formato dei comandi che un PC è in grado di eseguire. Affinché l'elaboratore sia in grado di comprenderlo, è necessario che venga effettuata una traduzione da un linguaggio all'altro.

Questa operazione viene eseguita da particolari programmi, detti **compilatori** o **traduttori**, che trasformano il **codice sorgente**, scritto dal programmatore, in **codice macchina**, eseguibile dal calcolatore.



I problemi e la loro soluzione

Alla base della scrittura di un programma c'è quindi l'esigenza di risolvere un **problema**.
Il significato del termine "problema", che deriva dal greco *probléma* ("questione proposta", da *probállein*, "gettare davanti"), può essere ben illustrato dalla seguente definizione: "*Situazione difficile che si deve affrontare e risolvere e che presenta soluzioni alternative*".

Per essere risolto, un problema deve essere innanzitutto **compreso**.

La comprensione passa attraverso una fase di analisi, detta "studio della situazione reale" o, appunto, **fase di analisi**.

Si parte quindi dal **problema** per arrivare al **programma** individuando la **soluzione**: questo è il principale compito del programmatore, cioè la **ricerca della soluzione**.
Ed è anche il principale problema: **come** si fa ad arrivare alla risoluzione di un dato problema?

Se consideriamo il problema come *"una questione da risolvere partendo da* **elementi noti** *mediante il ragionamento"* ne deriva che il programmatore dapprima si pone una serie di

domande necessarie proprio per avere gli elementi della **conoscenza del problema** e solo successivamente può iniziare il lavoro di progetto per l'"individuazione di un algoritmo" risolutivo.
Ma molte domande attendono una risposta.
– In che cosa consiste il problema?
– Come si costruisce la soluzione di un problema?
– Qual è il giusto punto di partenza?
– Quali metodologie o tecniche utilizzare?

Il **primo problema** è proprio quello di "capire il problema"! Questo non è solo di aritmetica, di geometria, di algebra ma generalmente è "*una questione, situazione difficile o complessa di cui si cerca la soluzione*": quindi il primo passo è quello di capire **cosa si deve risolvere** (**analisi del problema**).
Il **secondo problema** inizia quando finisce il primo: una volta individuato **cosa si deve risolvere** (e come si vedrà già questo non sempre è semplice!) "arriva il bello", poiché occorre ricercare la soluzione individuando **come risolvere il problema** (**individuazione della strategia risolutiva**).

Un "indizio che ci illumina" sul lavoro del programmatore lo possiamo trovare nella differenza tra problema e mistero: "il mistero è una porta chiusa, di cui non si ha la chiave; il problema è una porta da aprire, di cui si deve cercare la chiave, la serratura, il sistema di apertura".

Il programmatore è quindi dapprima un investigatore e successivamente uno stratega che con esperienza, intuito, fantasia e, perché no, intelligenza affronta il processo creativo relativo alla ricerca della "chiave di soluzione" del problema.

Analisi e comprensione del problema

Analisi del problema

Quindi l'**analisi** consiste nell'affrontare in modo sistematico il problema, analizzandone i vari aspetti della formulazione: mediante l'analisi scomponiamo situazioni complesse e piene di incognite in elementi riconoscibili e accessibili.
La comprensione del problema è necessaria per poter ricavare dalla realtà tutte quelle informazioni che risultano essenziali al fine di risolvere il problema in oggetto.

Occorre preliminarmente delimitare con precisione l'area di interesse, ricordando che più ampia è la portata del problema da risolvere e maggiori diventano le complessità da affrontare sul piano concettuale, progettistico e operativo.

Quindi bisogna individuare con precisione l'insieme dei dati ai quali si intende estendere i risultati e specificarne esattamente le condizioni di eleggibilità, ovvero le caratteristiche che ne determinano l'inclusione o l'esclusione (casi particolari e casi limite di funzionamento).

Comprensione del problema

Sovente un problema è espresso in modo confuso o fuorviante e una prima naturale esigenza consiste nell'eliminare ogni tipo di ambiguità dalla sua formulazione.
Per ottenere tale risultato è indispensabile eseguire un'operazione di "maquillage" del testo intervenendo con il duplice scopo di:

- evidenziare
  - i reali obiettivi del problema;
  - le regole;
  - i dati espliciti ed impliciti;
- eliminare
  - i dettagli inutili e ambigui.

Per agevolare le operazioni di comprensione del problema il programmatore utilizza due strumenti fondamentali: l'**astrazione** e la **creazione del modello**.

Astrazione, modellizzazione e definizione della strategia

Astrazione

L'**astrazione** è il primo importante strumento di lavoro nella fase di progettazione di un programma.

LE PAROLE DELL'INFORMATICA

Astrazione È il risultato di un processo secondo il quale assegnato un sistema, complesso quanto si voglia, si possono tenerne nascosti alcuni particolari evidenziando quelli che si ritiene essenziali ai fini della corretta comprensione del sistema.

Con l'astrazione il problema viene semplificato, ne vengono individuate le caratteristiche principali e contemporaneamente viene "scollegato dalla sua natura fisica" mediante il processo di modellizzazione.

Modellizzazione

Il **modello** è una rappresentazione del problema sotto una forma diversa da quella fisica: è possibile che uno stesso problema possa essere associato a:

- un modello **grafico**;
- un modello **tabellare**;
- un modello **simbolico** (come, ad esempio, mediante formule matematiche).

a seconda del progettista o delle esigenze e dell'ambito in cui si opera. Tutte e tre le modellizzazioni possono anche essere usate contemporaneamente per descrivere lo stesso problema.

LE PAROLE DELL'INFORMATICA

Modello Definiamo modello una rappresentazione semplificata della situazione in esame che esplicita gli elementi presenti, le loro proprietà e le relazioni tra essi.
Il processo di modellazione porta sempre a una astrazione del modello reale.

Definizione della strategia

I **modelli** rappresentano lo strumento attraverso il quale viene concretizzato il concetto di astrazione fornendo la base del processo di risoluzione del problema.
Il processo di ricerca della soluzione applica su tale modello un insieme di conoscenze e la soggettiva capacità di utilizzarle per elaborare una strategia risolutiva, ma la ricerca della soluzione non è guidata da un insieme di regole generali che consentano di trovare automaticamente la soluzione.

La **ricerca** può essere svolta in direzioni diverse: esistono tecniche che, partendo dai dati iniziali, cercano di arrivare alla soluzione applicando delle regole o degli operatori (**metodi diretti**), e tecniche che partendo dalle possibili soluzioni, applicando opportuni operatori, cercano di ottenere i dati di partenza (**metodi inversi**).

I **metodi** si basano su:
- l'utilizzo dell'esperienza passata;
  - somiglianza con altri problemi noti
  - analogia con altri problemi risolti
- la scomposizione dei problemi in sottoproblemi;
- la conoscenza dell'argomento;
- il procedimento per tentativi.

Gli **strumenti** a disposizione del progettista sono molteplici, tutti di medesima importanza e tutti da utilizzarsi in contemporanea, ma ogni strumento deve sempre essere integrato e confrontato con l'esperienza e le abilità creative proprie del progettista per vincere la sfida creativa-intellettuale di riuscire a dare vita alla soluzione del problema e quindi al progetto software.

Tra gli **strumenti** ricordiamo:
- le conoscenze di matematica e algebra;
- l'intuito;
- la logica e il ragionamento deduttivo/induttivo;
- la fantasia e l'ingegno.

Ricapitolando, per arrivare al progetto di programmi:
- si inizia dallo studio di problemi molto semplici, come ad esempio il calcolo della spesa in euro per una settimana di viaggi di andata e ritorno da casa a scuola, oppure il calcolo dei giorni che mancano alla fine delle lezioni. In questi casi specifici, dato che la situazione è nota, siamo facilmente in grado di risolverla;
- si passa poi ad affrontare problemi sempre più complessi ma, anche di fronte a questi, è tuttavia sempre la fase di analisi e di comprensione che ci consente di giungere alla soluzione;
- quando si è trovata la soluzione possiamo passare alla scrittura del programma vero e proprio.

L'algoritmo

Una volta individuata la strategia risolutiva, quindi quando si è "scoperto" il criterio risolutivo del problema, si scrivono le singole istruzioni da compiere, una dopo l'altra. L'insieme delle operazioni che permettono di risolvere un problema prende il nome di **algoritmo**, dal nome del matematico arabo **Al Khwarismi**, vissuto nell'800 d.C., ritenuto l'ideatore del procedimento che consente di effettuare il calcolo della moltiplicazione tra due numeri mediante la disposizione a cifre incolonnate (che è quella che usiamo ancora oggi).

L'**algoritmo** è quindi una sequenza ordinata di passi semplici che hanno lo scopo di portare a termine un compito complesso. Ricorrendo a un esempio culinario, la ricetta per la preparazione di un piatto può essere considerata come un algoritmo che, partendo da un insieme di singoli ingredienti ed eseguendo una sequenza di passi, porta come risultato finale al piatto in questione.

L'**algoritmo** deve avere le seguenti **caratteristiche**:
- deve essere generale, cioè risolvere un insieme di problemi;
- opera su dati in ingresso producendo un risultato in uscita;
- le istruzioni sono ordinate e in numero finito;
- le istruzioni sono chiare e unicamente interpretate da chi le esegue;
- il risultato viene prodotto in un tempo finito;
- ogni volta che viene eseguito con gli stessi dati produce gli stessi risultati.

Durante la fase di progetto vengono usate due modalità per descrivere gli algoritmi (e saranno quelle che anche noi utilizzeremo):
- il **diagramma** a **blocchi** (o **flow chart**), che è un metodo grafico che bene si presta a rappresentare gli algoritmi:
- il **linguaggio** di **progetto** (o **metalinguaggio**), che è composto dalla scrittura ordinata delle istruzioni dell'algoritmo in un formato molto vicino al linguaggio di programmazione, ma è molto semplificato e utilizza come lingua l'italiano.

Dall'algoritmo al codice macchina

L'algoritmo codificato in metalinguaggio è ancora molto lontano dal programma che viene eseguito dal calcolatore: sappiamo che il microprocessore è in grado di utilizzare solo il codice binario, cioè è in grado di distinguere solo il numero 0 dal numero 1, e quindi può eseguire solo un codice rappresentato con zero e uno, che si chiama codice macchina.

Ma per arrivare al codice macchina è prima necessario codificare l'algoritmo in un linguaggio particolare, il linguaggio di programmazione ad alto livello, e solo in un secondo tempo, mediante uno specifico programma, il compilatore, il nostro "programma sorgente" viene tradotto ottenendo il codice macchina.

Il compilatore, infatti, traduce le istruzioni scritte in linguaggio di programmazione una per una, generando il codice macchina che prende anche il nome di programma eseguibile proprio perché in questo formato è finalmente *eseguibile dal processore*.

Il primo passo è quindi quello di tradurre l'algoritmo in linguaggio di programmazione: un linguaggio di programmazione è un linguaggio rigoroso, composto da un insieme di regole lessicali e sintattiche come ogni linguaggio parlato, ma molto ridotte e schematizzate in numero tale da essere sufficienti per poter descrivere le istruzioni che il calcolatore deve eseguire per soddisfare le nostre esigenze, cioè risolvere i nostri problemi.

I linguaggi di programmazione ad alto livello sono linguaggi formali che eliminano l'ambiguità e le ridondanze tipiche del linguaggio naturale in modo tale che sia possibile la codifica dell'algoritmo in un programma scritto in un linguaggio ad alto livello per ottenere poi in modo automatico la sua traduzione in linguaggio macchina.

Esistono molteplici linguaggi di programmazione, diversi tra loro per metodologia e filosofia adottata (**paradigma**), complessità e numero di istruzioni, settore e ambito di applicazione.
Tra tutti ricordiamo solamente il linguaggio Pascal, elaborato nel 1968 dal professor Wirth del Politecnico di Zurigo, che è tutt'oggi il linguaggio più diffuso e utilizzato per affrontare lo studio della programmazione, e il linguaggio C, messo a punto da Dennis Ritchie per implementare i primi sistemi operativi negli anni '70, che è il linguaggio di riferimento sia per i programmatori "più esperti" sia per i moderni linguaggi di programmazione dell'ambiente Web (C++, Java, PHP ecc. usano la stessa sintassi propria del linguaggio C).

Il linguaggio di programmazione permette di realizzare un programma che implementa l'algoritmo in maniera precisa in un linguaggio "ad alto livello", in modo tale che non dipenda dal particolare calcolatore su cui è stato realizzato: un programma viene realizzato senza neanche sapere su quale macchina dovrà poi essere eseguito.

Uno schema riassuntivo delle differenze tra i vari linguaggi, analizzati dal punto di vista dell'esecutore, è il seguente:

ABBIAMO IMPARATO CHE...

- Il programmatore studia il problema e ne individua un algoritmo risolutivo, cioè l'insieme delle operazioni che devono essere eseguite per raggiungere il risultato.
- Descrive la soluzione di un problema utilizzando un linguaggio di programmazione: tale linguaggio ad alto livello è di fatto molto lontano dal formato dei comandi che un PC è in grado di eseguire (il codice macchina).
- L'algoritmo è quindi una sequenza ordinata di passi semplici che hanno lo scopo di portare a termine un compito complesso.
- L'algoritmo viene codificato in un linguaggio di programmazione ad alto livello dal programmatore e tradotto mediante il compilatore in codice eseguibile dal processore.
- Esistono molteplici linguaggi di programmazione ad alto livello, diversi tra loro per metodologia e filosofia adottata (paradigma), complessità e numero di istruzioni, settore e ambito di applicazione.
- Per scrivere l'algoritmo il programmatore effettua l'analisi del problema, cioè affronta in modo sistematico i vari aspetti della sua formulazione: mediante l'analisi scompone situazioni complesse e piene di incognite in elementi riconoscibili e accessibili.
- Con l'astrazione vengono individuate le caratteristiche principali di un problema "scollegandolo dalla sua natura fisica" e creandone un modello semplificato.
- Dallo studio del modello si individua la strategia risolutiva e quindi si scrivono le singole istruzioni che costituiscono l'algoritmo.

UNITÀ DIDATTICA 2

IMPARIAMO A FARE I DIAGRAMMI A BLOCCHI

IN QUESTA UNITÀ IMPAREREMO...

- come descrivere l'algoritmo risolutivo utilizzando i diagrammi a blocchi
- il concetto di programmazione strutturata

L'uomo come esecutore di algoritmi

Abbiamo detto che con il termine algoritmo intendiamo l'"insieme di azioni elementari che consentono di risolvere un problema trasformando i dati iniziali del problema stesso nel risultato".

Non necessariamente il problema deve essere di tipo matematico: ad esempio, può essere un problema culinario come preparare "il vitello tonnato"; l'algoritmo in questo caso si identifica con la ricetta che il cuoco applica per la realizzazione del piatto.

Un altro algoritmo potrebbe essere il procedimento che seguiamo per sostituire l'olio nel motore della nostra automobile, oppure quello che eseguiamo per prelevare una lattina di bibita dalla macchina distributrice automatica e via dicendo.

Un algoritmo è quindi una sequenza ordinata di passi semplici che hanno lo scopo di portare a termine un compito più complesso, qualunque sia la natura del problema.

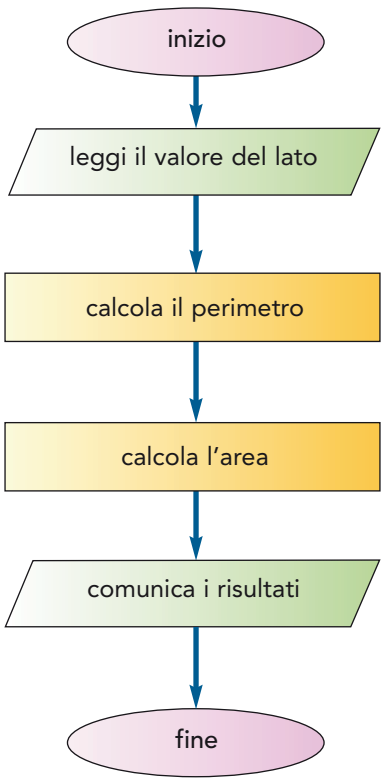
L'esecutore dell'algoritmo non è quindi sempre un calcolatore, anzi, il più delle volte è un uomo che esegue un algoritmo, anche senza saperlo.
Noi stessi, tutti i giorni, eseguiamo delle operazioni in sequenza, come quelle di alzarci, vestirci, andare a scuola ecc., che non sono altro che un algoritmo.

In questa Unità didattica impareremo a descrivere anche questi algoritmi mediante un linguaggio grafico che è alla base della rappresentazione di tutti gli algoritmi: la differenza

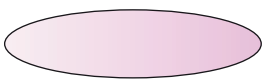
con gli algoritmi che devono essere poi eseguiti da un calcolatore sta nella codifica finale nel linguaggio di programmazione: se l'esecutore è un essere umano li scriviamo in "lingua italiana" in modo che l'uomo li possa capire, se è un calcolatore in "codice binario" in modo che la CPU del nostro personal computer li possa eseguire.

Diagrammi a blocchi o flow chart

Esistono diversi modi per scrivere un algoritmo: il più semplice, e anche il più diffuso, è quello che utilizza i diagrammi a blocchi (o *flow chart*, "diagrammi di flusso"): si tratta di una rappresentazione grafica, di facile interpretazione per chiunque.
Osserviamo un diagramma a blocchi che descrive le operazioni necessarie per effettuare il calcolo del perimetro e dell'area di un quadrato:



Possiamo distinguere tre diversi elementi grafici utilizzati per differenziare la natura delle operazioni.



Blocco terminale: è usato solo per indicare l'inizio e la fine dell'algoritmo, che devono essere unici nel diagramma.

Blocco di comunicazione: al suo interno vengono scritte le operazioni di dialogo tra l'uomo e la macchina (leggi, scrivi ecc.).

Blocco di elaborazione: contiene le istruzioni vere e proprie che compiono operazioni.

Le operazioni vengono eseguite seguendo le frecce, una dopo l'altra, quindi in sequenza: la sequenza di operazioni è la caratteristica principale di un algoritmo ed è la struttura fondamentale di un programma scritto in un linguaggio di programmazione.

Una volta completato il diagramma di flusso è possibile trasformarlo nel **linguaggio di progetto** (o **metalinguaggio**), scrivendolo in sequenza nel modo seguente:

```
inizio
  leggi il valore del lato
  calcola il perimetro
  calcola l'area
  comunica i risultati
fine
```

Il metalinguaggio è l'ultimo passo per arrivare alla scrittura di un programma in linguaggio di programmazione: a questo punto, infatti, sarà sufficiente tradurre ogni singola istruzione nel formato richiesto dal linguaggio che si vuole utilizzare. Vediamo un esempio.

Esempio 1 Quanto costa andare a scuola?

Calcoliamo il costo settimanale di carburante utilizzato per andare a scuola.
Innanzitutto individuiamo le operazioni da eseguire, cioè l'algoritmo che risolve il problema:
– calcolare il numero di litri di benzina consumati;
– calcolare il costo totale del carburante giornaliero;
– determinare il costo settimanale.

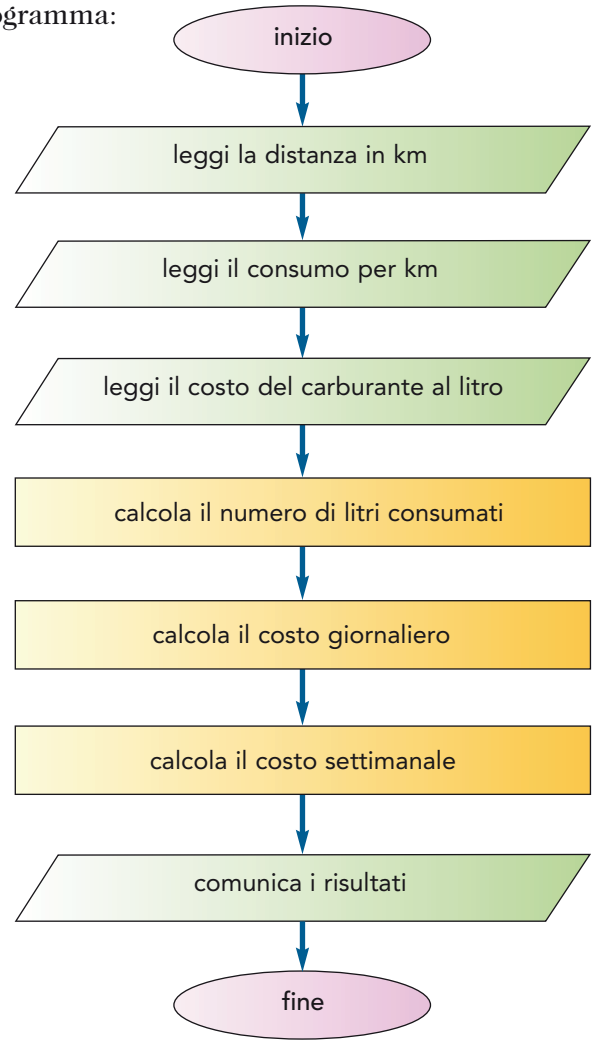
A questo punto è necessario avere a disposizione alcuni dati da fornire al programma, in modo che esso possa elaborarli. Questi dati, chiamati **dati di ingresso** (o **input**), sono:
– il costo di un litro di carburante;
– il numero di km tra la casa e la scuola;
– il numero di km che il mezzo utilizzato effettua con un litro di carburante.

I risultati della elaborazione prendono il nome di **dati in uscita** (o **output**) del programma e vengono comunicati all'utente.

Anche in questo primo semplice esempio possiamo osservare come viene rispettata la struttura generale di ogni algoritmo, costituito da tre fasi fondamentali: **input** dai dati iniziali, **elaborazione** mediante il programma ed output dei risultati.

In un algoritmo questo schema può anche essere ripetuta diverse volte.

Scriviamo ora il diagramma di flusso del programma:



La traduzione in metalinguaggio è la seguente:

```
inizio
  leggi la distanza in km
  leggi il consumo per km
  leggi il costo del carburante al litro
  esegui il calcolo del numero di litri consumati
  esegui il calcolo del costo giornaliero
  esegui il calcolo del costo settimanale
  comunica i risultati
fine
```

L'algoritmo che abbiamo scritto ha la proprietà di essere generico, può essere cioè utilizzato per la risoluzione di problemi simili senza che venga specificato il tipo di automezzo utilizzato, il tipo di percorso effettuato o di carburante impiegato. Potrebbe ad esempio essere applicato al calcolo dei costi di trasporto aereo di diamanti dalle miniere del Sudafrica a Parigi!

La sequenza delle istruzioni è ordinata, cioè esiste un ordine preciso in base al quale vengono eseguite le istruzioni, che non possono essere scambiate di posto tra loro.
Il numero di operazioni deve essere finito: ciò significa che le istruzioni possono essere anche moltissime, ma non in numero infinito, altrimenti l'algoritmo non terminerebbe mai.
Ogni operazione, inoltre, deve essere chiara, in modo che l'esecutore possa comprendere ed eseguire ogni istruzione senza errori di interpretazione.
Deve sempre esserci una fine, sia che venga prodotto un risultato valido sia che non esista soluzione: tali risultati devono essere gli stessi tutte le volte che l'algoritmo viene eseguito sugli stessi dati, anche se dovesse essere utilizzato un elaboratore diverso.

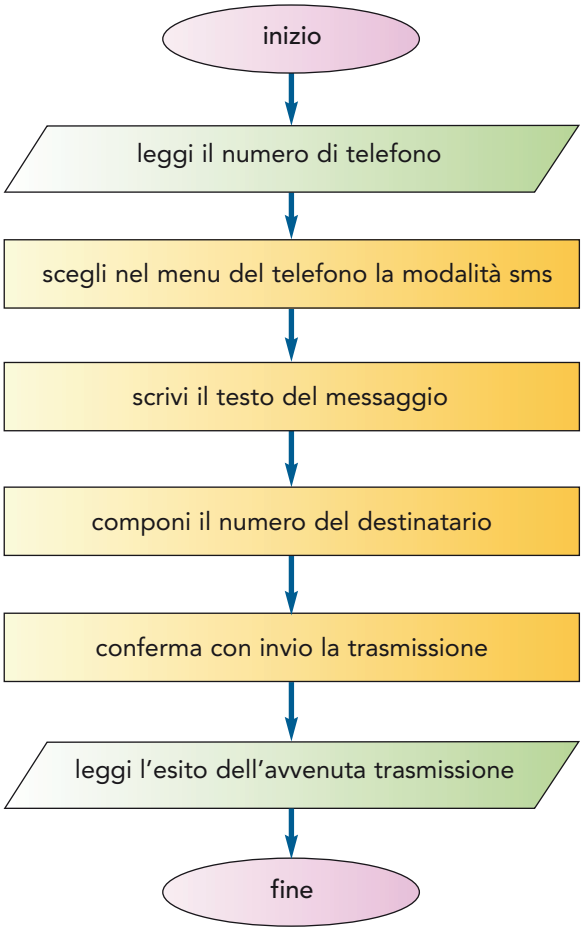
Vediamo un secondo esempio.

Esempio 2 Spediamo un sms

Si descriva la procedura eseguita per spedire un sms.
Innanzi tutto individuiamo le operazioni da eseguire, cioè l'algoritmo che risolve il problema:
- individuare il numero dell'amico da messaggiare;
- comporre il testo del messaggio;
- spedire l'sms;
- visualizzare la conferma dell'invio.

È necessario avere a disposizione un unico dato in ingresso (o input): il numero di telefono al quale spedire il messaggio. In questo caso non abbiamo un vero e proprio risultato dell'elaborazione ma semplicemente la conferma di avvenuta trasmissione ci permette di considerare terminato l'algoritmo.

Scriviamo il diagramma di flusso dell'algoritmo:



La traduzione in metalinguaggio è la seguente:

```
inizio
  leggi il numero di telefono
  esegui "accendi" il telefono in modalità sms
  esegui "scrivi" il messaggio
  esegui componi il numero del destinatario
  esegui la trasmissione del messaggio mediante il tasto invio
  leggi l'esito dell'avvenuta trasmissione
fine
```

Vediamo un ultimo esempio.

Esempio 3 Secchi con l'acqua

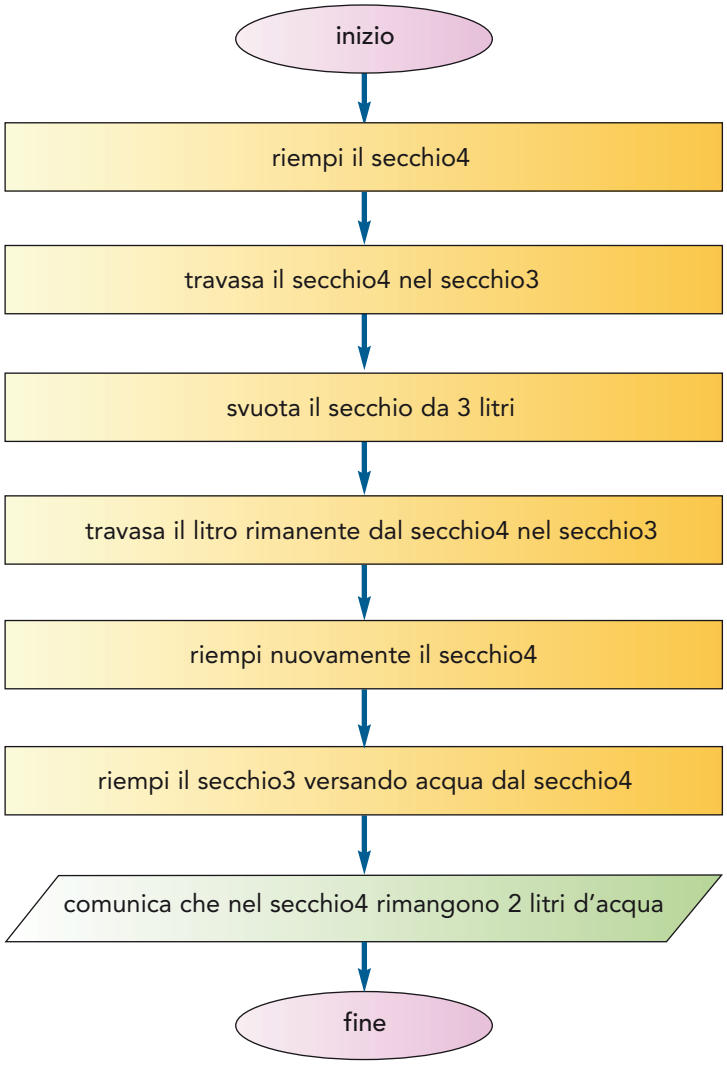
Descrivi l'algoritmo che permetta di riempire un secchio della capacità di 4 litri con soli 2 litri di acqua utilizzando solo due secchi con capacità volumetrica rispettivamente di 4 e 3 litri.
Partiamo sempre individuando le operazioni da eseguire per risolvere il problema e in questo caso il procedimento è un po' più complesso in quanto è necessario individuare una strategia risolutiva.

Con **strategia** intendiamo il "ragionamento" che ci permette di ottenere la soluzione, cioè l'idea che abbiamo avuto per risolvere "il nocciolo della questione".
In questo caso ci accorgiamo che la differenza tra i due contenitori è di 1 litro di capacità: questa quantità ci può servire per ottenere la quantità di 2 litri desiderata:
- riempio per primo il secchio da 4 litri;
- lo travaso in quello da 3 litri in modo che rimanga 1 litro nel primo secchio;
- ora svuoto il secchio da 3 litri;
- travaso il litro presente nel secchio da 4 litri in quello da 3 litri;
- riempio nuovamente il secchio da 4 litri;
- travaso due litri in quello da 3 litri, dove 1 litro è già presente;
- nel secchio da 4 litri sono rimasti 2 litri.

In questo caso non abbiamo né dati in input né in output ma semplicemente una sequenza di istruzioni di elaborazione: per semplicità chiamiamo **secchio4** il secchio con capacità di 4 litri e **secchio3** il secchio con capacità di 3 litri.

Non è proprio vero che non ci sono i dati di input: in questo caso i dati di input, che sono i dati di partenza sui quali il nostro programma dovrà compiere le operazioni, sono presenti all'interno del testo del problema stesso, fanno cioè parte delle **specifiche** del problema.

Scriviamo il diagramma di flusso del programma:



La traduzione in metalinguaggio è la seguente:

```
inizio
  esegui    il riempimento del primo secchio da 4 litri
  esegui    il travaso in quello da 3 litri: rimane 1 litro
            nel primo secchio
  esegui    lo svuotamento del secchio da 3 litri
  esegui    il travaso del litro presente nel secchio da 4 litri
            in quello da 3 litri
  esegui    il riempimento del secchio da 4 litri
  esegui    il travaso di 2 litri in quello da 3 litri,
            dove 1 litro è già presente
  comunica  che nel secchio da 4 litri sono presenti i 2 litri
            richiesti
fine
```

La programmazione strutturata

Con il termine **programmazione strutturata** si intende un **paradigma** di programmazione emerso tra gli anni '60 e '70 che ha introdotto i concetti fondamentali che sono stati alla base di tutti gli altri paradigmi successivi, non ultimo quello orientato agli oggetti.

Un **paradigma di programmazione** è uno stile fondamentale di programmazione, ovvero un insieme di strumenti concettuali forniti da un linguaggio di programmazione per la stesura dei programmi.

NOTA STORICA

Le idee chiave della programmazione strutturata si possono ricondurre alla critica dell'istruzione del *salto incondizionato* (o *goto*, "vai a") che rappresentava nei primi linguaggi lo strumento fondamentale per la definizione degli algoritmi e che rendeva il codice scritto praticamente incomprensibile, tanto da essere chiamato spaghetti code per la sua natura "ingarbugliata"!

La **programmazione strutturata** è una tecnica di programmazione che limita l'utilizzo delle regole ammesse per descrivere gli algoritmi a sole tre regole fondamentali:
- la **sequenza** (o concatenazione);
- la **selezione** (o alternativa);
- l'**iterazione** (o ciclo, ripetizione).

La prima figura strutturale fondamentale, la **sequenza**, è stata descritta e utilizzata in questa Unità didattica mentre nelle prossime due Unità affronteranno rispettivamente la selezione e l'iterazione.

Il programmatore che applica le regola di strutturazione, colui cioè che utilizza solo queste tre tipologie di istruzioni, ottiene la descrizione degli algoritmi in modo chiaro, facilmente leggibile e comprensibile: tale codice può essere agevolmente capito e modificato in un secondo tempo anche da un eventuale programmatore che non sia il suo "creatore".

I **linguaggi di programmazione strutturati** iniziarono a emergere nei primi anni '70 facendo tesoro delle idee dei due matematici italiani **Corrado Bohm** e **Giuseppe Jacopini** che per primi sottolinearono, in un teorema, come le tre figure strutturali fondamentali offrivano un insieme di strutture di controllo completo, che garantivano la possibilità di descrivere tutti gli algoritmi.
Secondo tale teorema, "qualsiasi programma può essere riscritto avendo a disposizione altri tre tipi di strutture di controllo: *sequenza*, *ripetizione* e *alternativa*".

Ricapitolando, se il linguaggio di programmazione ammette almeno una forma di sequenza, una di alternativa e una di ripetizione, è in grado di descrivere ogni algoritmo: nei moderni linguaggi, per semplificare l'operato dei programmatori, vengono fornite alcune varianti per ciascuna famiglia di strutture di controllo, ma devono essere sempre presenti tutte e tre.

Fra i linguaggi tipici del paradigma strutturato si possono citare il Pascal, il C, l'Algol, il Cobol ecc.

ABBIAMO IMPARATO CHE...

- L'algoritmo è l'insieme delle operazioni che permettono di risolvere un problema generico, non necessariamente di natura matematica.
- Nella vita quotidiana eseguiamo algoritmi "praticamente" in ogni occasione senza che ce ne rendiamo conto: quindi l'esecutore di un algoritmo non è necessariamente il personal computer.
- Il diagramma a blocchi (o flow chart) è un metodo grafico che bene si presta a rappresentare gli algoritmi.
- Il linguaggio di progetto (o metalinguaggio) è composto dalla scrittura ordinata delle istruzioni dell'algoritmo in un formato molto vicino al linguaggio di programmazione.
- La programmazione strutturata è una tecnica di programmazione che limita l'utilizzo delle regole ammesse per descrivere gli algoritmi a sole tre regole fondamentali:
  – la sequenza (o concatenazione);
  – la selezione (o alternativa);
  – l'iterazione (o ciclo, ripetizione).
- Il teorema di Bohm e Jacopini dice che "qualsiasi programma può essere riscritto avendo a disposizione altri tre tipi di strutture di controllo: *sequenza*, *ripetizione* e *alternativa*".

UNITÀ DIDATTICA 3

CONOSCIAMO LA SELEZIONE E LE CONDIZIONI LOGICHE

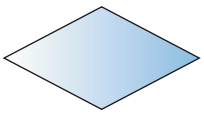
IN QUESTA UNITÀ IMPAREREMO...

- cos'è l'istruzione di selezione
- come descriverla utilizzando i diagrammi a blocchi e il linguaggio di progetto

L'istruzione di selezione doppia

Nella descrizione di un algoritmo si può verificare che le operazioni da eseguire siano diverse a seconda dei dati inseriti.
Ad esempio, se dobbiamo comperare un oggetto, dobbiamo verificare se i soldi a disposizione sono sufficienti, e quindi si prospettano due soluzioni alternative:
– se i soldi ci bastano, allora compriamo l'oggetto;
– altrimenti, se i soldi non sono sufficienti, dobbiamo procurarcene degli altri.

Nel diagramma a blocchi, questa situazione è illustrata ricorrendo a un nuovo elemento grafico, detto blocco di test o di confronto.



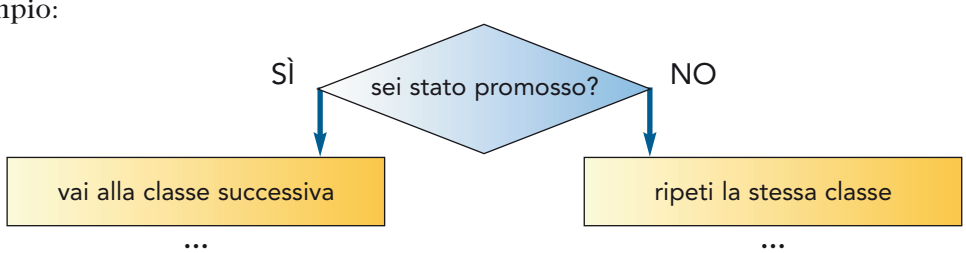
Blocco di confronto: al suo interno vengono effettuate operazioni di confronto e il risultato può essere soltanto SÌ o NO (oppure VERO o FALSO).

Ecco degli esempi di istruzioni di confronto che possiamo scrivere nei blocchi condizionali.

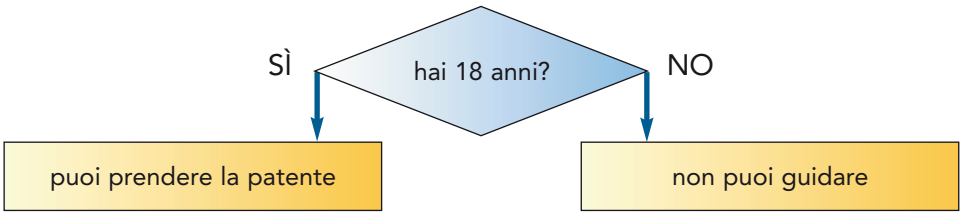
Ho soldi sufficienti per comperare la moto?
Il numero 50 è maggiore di 0?
Mario è un alunno di questa classe?
Il serbatoio è pieno?
Una mucca depone le uova?
27 è un numero pari?

L'unica possibile risposta a queste domande è **SÌ** oppure **NO**, esistono cioè soltanto due alternative: si tratta dunque di "istruzioni di confronto" – che prendono il nome di **condizioni logiche** – e, a seconda del valore della risposta, è sempre necessario intraprendere percorsi alternativi.

Ad esempio:



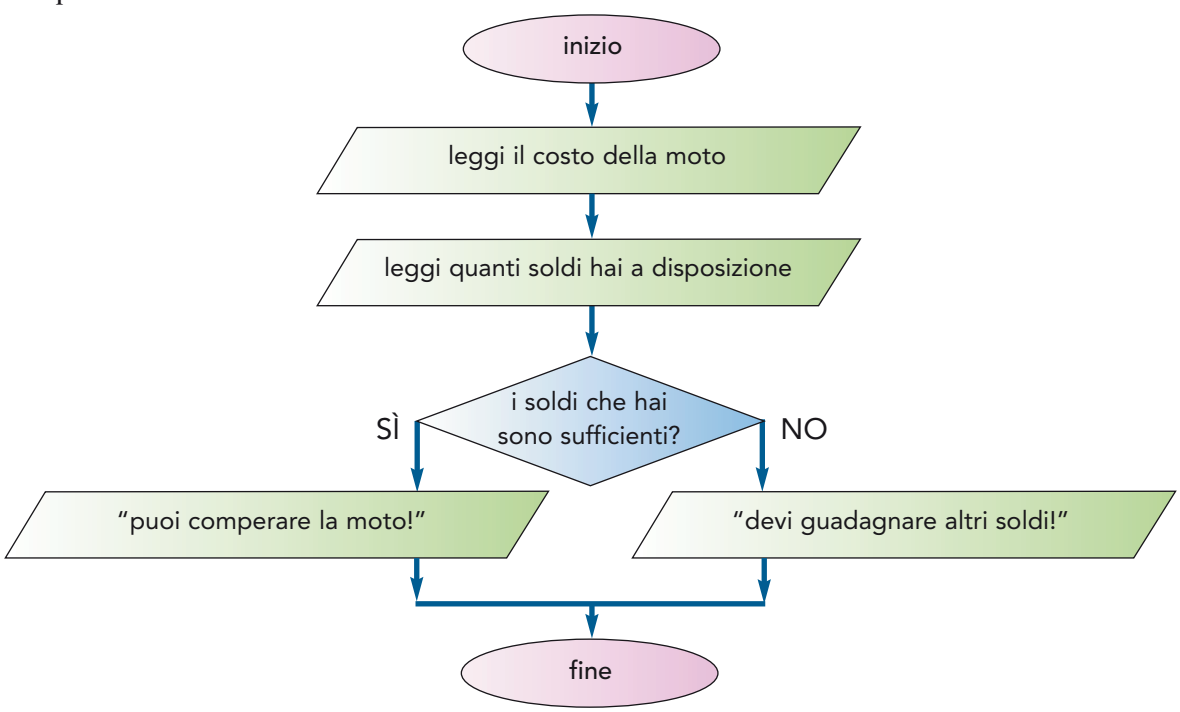
oppure:



Possiamo osservare che, a seconda della risposta data alla domanda (cioè al **test**), le strade "si dividono" e vengono eseguite operazioni diverse nei due "rami".

La presenza di istruzioni in entrambi i rami del diagramma fa sì che in questo caso si parli di **selezione doppia**.

Completiamo adesso il diagramma a blocchi per l'algoritmo che descrive la procedura per l'acquisto di una moto:



Vediamo come tradurre questa istruzione in **linguaggio di progetto**: dato che dobbiamo scrivere un'istruzione per riga, è necessario stabilire una regola che ci permetta di riconoscere in modo chiaro l'istruzione di selezione.

Utilizziamo a tale scopo una scrittura incolonnata in modo particolare, ricorrendo a tre parole che ci consentono di individuare l'istruzione di test e i due percorsi alternativi che possono essere intrapresi.

```
se <il test dà risultato vero>
  allora <esegui questa istruzione>
  altrimenti <esegui quest'altra istruzione>
```

Traduciamo il precedente diagramma di flusso in metalinguaggio:

```
inizio
  leggi quanto costa la moto
  leggi quanti soldi hai a disposizione
  se i soldi che hai sono sufficienti
    allora
      scrivi 'puoi comperare la moto!'
    altrimenti
      scrivi 'devi guadagnare altri soldi!'
 fine
```

L'incolonnamento utilizzato consente di riconoscere subito le istruzioni interne a un'istruzione di selezione, in quanto esse sono spostate più a destra di quest'ultima.

Le due operazioni di scrittura vengono scritte alternativamente, a seconda del risultato del test.

Con la notazione che abbiamo adottato risulta semplice seguire il flusso del programma nel linguaggio di progetto.

```
se <condizione>
  allora
    esegui il primo ramo
  altrimenti
    esegui il secondo ramo
```

Le regole utilizzate per scrivere il programma in modo incolonnato prendono il nome di **regole di indentazione**.

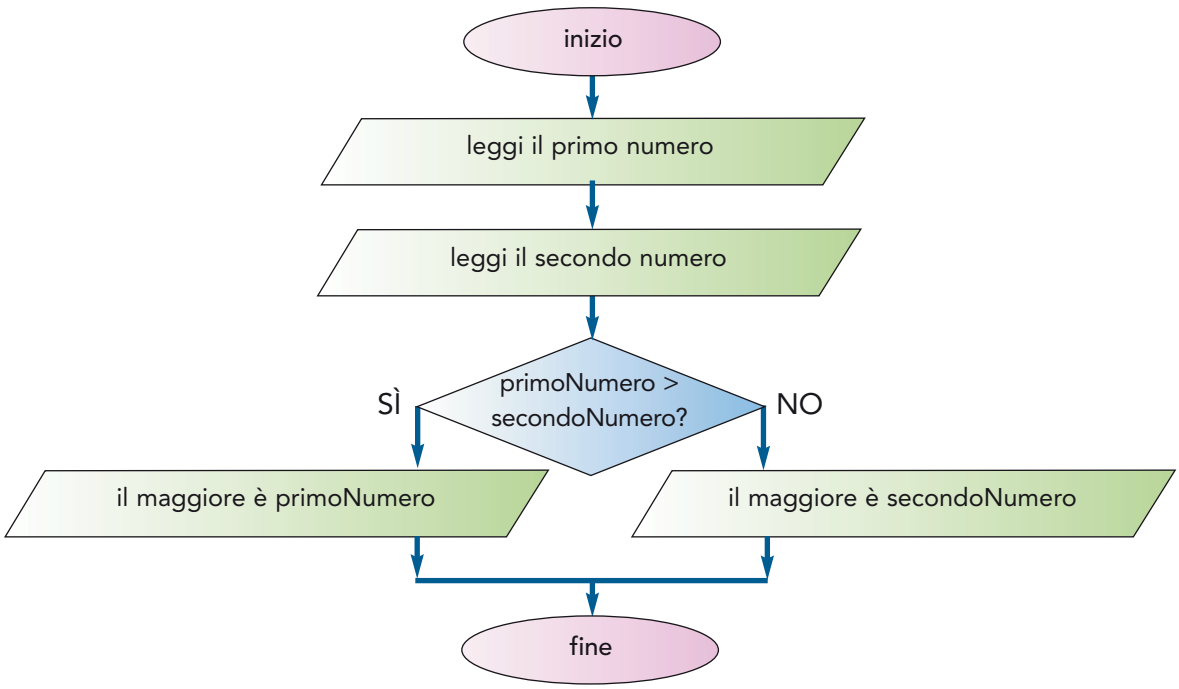
Le regole di indentazione non sono uniche né obbligatorie, e ogni programmatore può scegliere di scrivere un programma nel modo che più lo facilita.
Unico obbligo da rispettare è l'"uniformità": una volta stabilite, infatti, le regole devono essere applicate sempre nello stesso modo.

Esempio 4 Determinare il maggiore tra due numeri

Leggiamo due numeri da tastiera e determiniamo quale dei due è il maggiore.
In questo esempio, all'interno della istruzione di test, è necessario effettuare il confronto tra i due numeri, quindi l'istruzione condizionale è la seguente:

```
primoNumero > secondoNumero
```

Inseriamola pertanto nel blocco di test:



Traducendo l'algoritmo in linguaggio di progetto otteniamo:

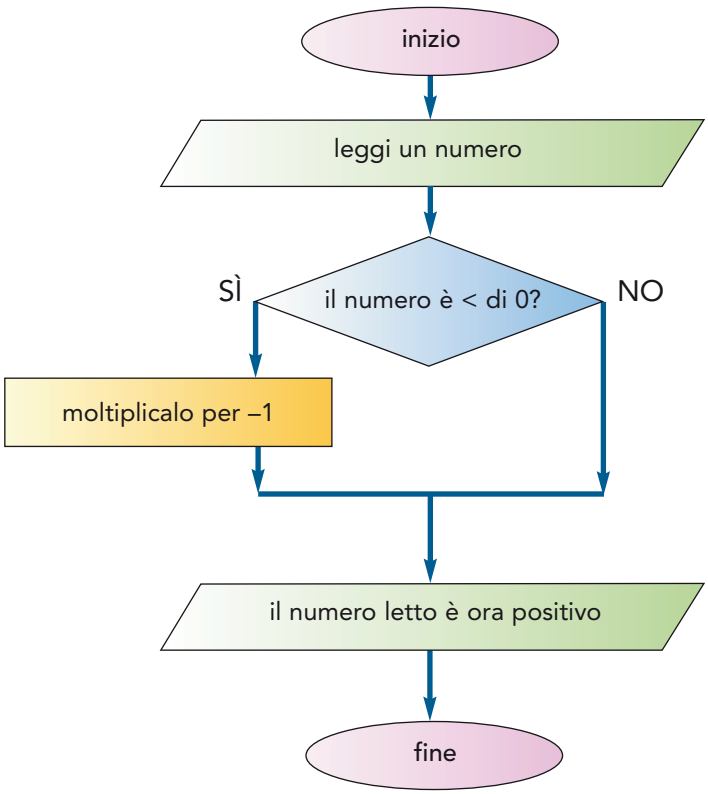
```
inizio
leggi il primo numero
leggi il secondo numero
se il primo numero è > del secondo numero
  allora
    scrivi 'il maggiore è il primo numero'
  altrimenti
    scrivi 'il maggiore è il secondo numero'
fine
```

La selezione semplice

È possibile che le operazioni da eseguire siano presenti in un solo ramo, quando ad esempio si deve effettuare un'operazione in una sola situazione. In questo caso, si parla di selezione semplice. Vediamo un esempio.

Esempio 5 Trasformazione di numeri negativi in positivi

Scriviamo un programma che trasforma i numeri negativi in numeri positivi: in questo caso, se il numero è già positivo non dobbiamo effettuare nessuna operazione, mentre se è negativo lo moltiplichiamo per –1, in modo da cambiarne il segno.



Traduciamo ora il linguaggio di flusso in linguaggio di progetto ottenendo:

```
inizio
leggi un numero
se il numero è < 0
  allora
    esegui moltiplicalo per -1 in modo da renderlo positivo
scrivi 'ora il numero letto è sicuramente positivo'
fine
```

Come possiamo notare, viene a mancare un "pezzo" di istruzione e il programma risulta più semplice: prende infatti il nome di selezione semplice.
L'istruzione di selezione semplice può quindi essere considerata un caso particolare di selezione doppia dove viene a mancare il ramo **altrimenti**; non è invece possibile avere un'istruzione semplice senza il ramo **allora**, con cioè il solo ramo **altrimenti**: se il ramo è uno solo, e quindi siamo in presenza di selezione semplice, necessariamente deve essere presente il ramo **allora**.

ABBIAMO IMPARATO CHE...

- L'istruzione di selezione permette di percorrere alternativamente due rami del diagramma di flusso, in modo da eseguire un'istruzione oppure un'altra a seconda del valore del test.
- L'istruzione che viene testata si chiama anche condizione logica in quanto ha come possibili soluzioni solamente i due valori logici SÌ e NO, oppure VERO o FALSO.
- Nel caso in cui siano presenti istruzioni in entrambi i rami, l'istruzione prende il nome di selezione doppia e, nel linguaggio di progetto, viene scritta nella seguente forma:

```
se <condizione logica>
  allora
     esegui il primo ramo
  altrimenti
     esegui il secondo ramo
```

- Se invece le istruzioni sono presenti in un solo ramo si parla di selezione semplice.

```
se <condizione>
  allora
     esegui il primo ramo
```

- L'unico ramo presente nell'istruzione di selezione semplice è il ramo `allora`.

UNITÀ DIDATTICA 4

CONOSCIAMO L'ITERAZIONE DEFINITA E INDEFINITA

IN QUESTA UNITÀ IMPAREREMO...

- che cos'è l'istruzione di iterazione
- come descriverla utilizzando i diagrammi a blocchi e nel linguaggio di progetto

L'istruzione di iterazione o ciclo

Nella descrizione di un algoritmo può capitare che alcune operazioni debbano essere eseguite più di una volta, cioè ripetute in modo identico.
Ad esempio, se dobbiamo effettuare il conto alla rovescia da 10 a 1, dobbiamo **ripetere** per 10 volte l'operazione "sottrai 1", mentre se dobbiamo riempire d'acqua una vasca da bagno utilizzando un bicchiere è necessario versare ripetutamente il contenuto del bicchiere **finché** la vasca non risulti piena.

La ripetizione di un insieme di istruzioni prende il nome di **iterazione** o **ciclo** (in inglese, *loop*); con questa istruzione indichiamo al calcolatore di ripetere un gruppo di istruzioni:
– un numero di volte ben determinato, come nel primo esempio;
– finché non si verifica una situazione, come nel secondo esempio.

Il gruppo di istruzioni ripetute prende il nome di **corpo del ciclo**: l'istruzione di iterazione viene classificata proprio per la caratteristica di conoscere o meno a priori il numero di ripetizioni del corpo del ciclo che deve effettuare.

Quando il numero di ripetizioni è noto a priori, cioè sai quante volte le istruzioni devono essere ripetute, il ciclo prende il nome di iterazione definita.
Quando invece il ciclo viene ripetuto un numero di volte sconosciuto a priori e termina quando si verifica una particolare condizione, l'iterazione si dice indefinita.

LE PAROLE DELL'INFORMATICA

Iterazione Struttura di controllo che permette di eseguire più volte un'istruzione. Nella maggior parte dei casi occorre indicare, oltre all'istruzione che dovrà essere iterata, anche la condizione per uscire dall'iterazione stessa.

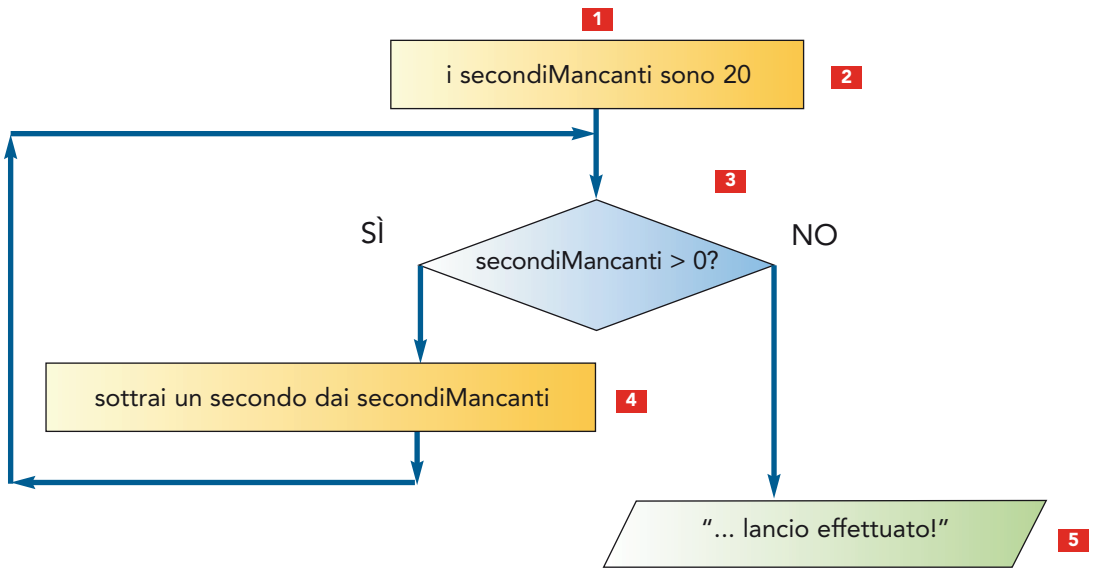
Vediamo alcuni esempi dei due diversi tipi di istruzioni.

Iterazioni definite	Iterazioni indefinite
Fai 10 giorni di ferie.	Mescola la pasta finché è cotta.
Leggi 20 pagine.	Innaffia le piante finché basta.
Per 10 volte scrivi "devo studiare di più".	Mangia finché sei sazio.
Attacca 30 figurine sull'album.	Mentre piove fatti una dormita.
Invia 100 sms alla zia.	Mentre non sei stanco, studia.
Imbianca con 3 mani la parete.	Finché c'è vita c'è speranza.
Tra 210 giorni è finita la scuola.	Mentre il serbatoio non è pieno metti benzina.
Per 210 giorni devi andare a scuola.	Dormi finché hai sonno.

Osservando attentamente i due gruppi di istruzioni, possiamo affermare che nelle iterazioni definite è sempre presente un numero, cioè viene sempre indicato chiaramente quante volte deve essere eseguita una operazione, mentre in quelle indefinite non si sa per quante volte questa debba essere ripetuta.

Esempio 6 Conto alla rovescia

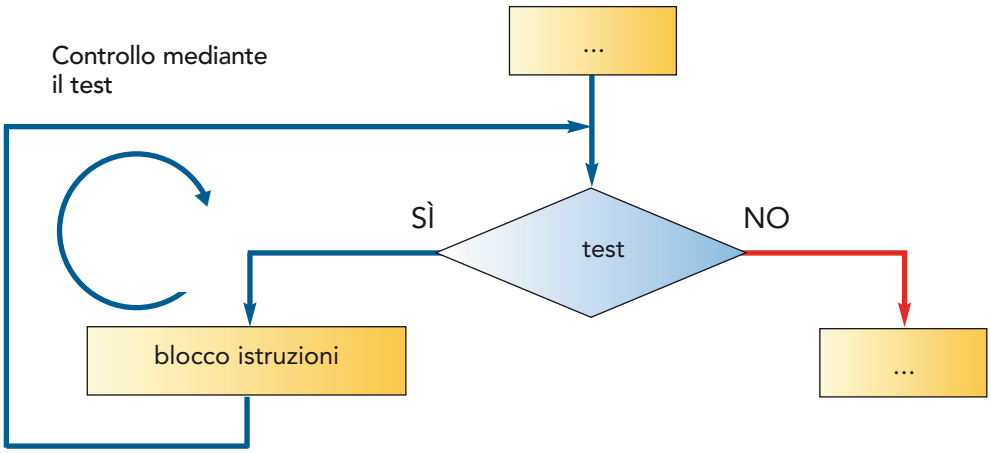
Un missile è in partenza dalla base spaziale Venere1: mancano 20 secondi al lancio e deve essere effettuato il conto alla rovescia.
Il diagramma di flusso viene realizzato utilizzando i blocchi già conosciuti. Per ottenere la ripetizione ciclica combiniamo un'istruzione di test con una (o più) istruzioni di esecuzione.



1. Il programma inizia con l'**istruzione** 2, che stabilisce il valore da cui iniziare il conto alla rovescia.
2. L'**istruzione** 3 ti permette di verificare se il conto alla rovescia è terminato o meno, cioè se devi continuare a sottrarre un numero (risposta SÌ quando il conteggio è maggiore di 0) oppure se il valore del conteggio è arrivato a 0 e quindi il razzo può partire.
3. L'**istruzione** 4 viene eseguita solo quando il test ha esito positivo e decrementa di una unità il valore del conteggio: quindi ti consente di **ritornare** con la sequenza di istruzioni "indietro nel programma", cioè prima del punto 3, in modo da poter effettuare un nuovo confronto per stabilire se ora il conteggio ha raggiunto il valore 0.
4. A questo punto si ritorna a eseguire l'**istruzione** 3 e, se il test ha esito ancora positivo, devi ripetere il ramo di sinistra, eseguendo nuovamente l'**istruzione** 4 e ritornando prima del punto 3.
5. Continua a ripetere queste istruzioni fino a quando il risultato del test 3 diventa falso, cioè quando i secondiMancanti sono finalmente arrivati a 0.
6. Ora esegui il ramo di destra e puoi… effettuare il lancio del missile.

Codifichiamo l'iterazione

Lo schema generale di un'istruzione iterativa è riportato nella figura che segue, dove possiamo osservare come il percorso di un ramo formi un "anello" con l'istruzione di test.



Il **test** controlla se deve essere **ripetuto** il blocco di istruzioni, oppure se la ripetizione del ciclo termina e quindi si deve "uscire" dalla parte di destra dove prosegue il programma.

Esistono diversi tipi di ciclo e ciascuno di essi può essere realizzato in diverse modalità che verranno analizzate successivamente, insieme alla codifica in linguaggio Pascal.

Una scrittura in linguaggio di progetto che permette di descrivere l'istruzione di ciclo è:

```
mentre <la condizione è vera> fai
    <istruzioni da ripetere>
```

L'esempio 6 viene quindi così codificato in linguaggio di progetto:

```
1 inizio
2  inizia il conto alla rovescia partendo da 20
3  mentre il conto alla rovescia è maggiore di 0 fai
4     sottrai 1 al conteggio
5  ora il razzo parte
6 fine
```

La tabella di traccia o trace table

Uno strumento molto utile che ci permette di seguire passo passo il funzionamento del programma, soprattutto quando sono presenti istruzioni cicliche, è la tabella di traccia (o trace table). In questa tabella sono presenti tante colonne quante sono le istruzioni da controllare. Ritornando all'esempio 6, avremo pertanto:

Numero di istruzione	Istruzione di test	Valore dei secondiMancanti	Risultato

La tabella viene realizzata aggiungendo una riga per ogni istruzione eseguita e riportando in ogni colonna l'esito che questa istruzione ha avuto nel programma.

Numero di istruzione	Istruzione di test	Valore dei secondiMancanti	Risultato
2		20	
3	VERO		
4		19	
3	VERO		
4		18	
3	VERO		
4		17	
3	VERO		
...			
4		2	
3	VERO		
4		1	
3	VERO		
4		0	
3	FALSO		
5		0	Lancio del missile

La tabella di traccia appena realizzata può sembrare inutile, ma spesso è lo strumento più efficace per individuare eventuali errori presenti nel codice.

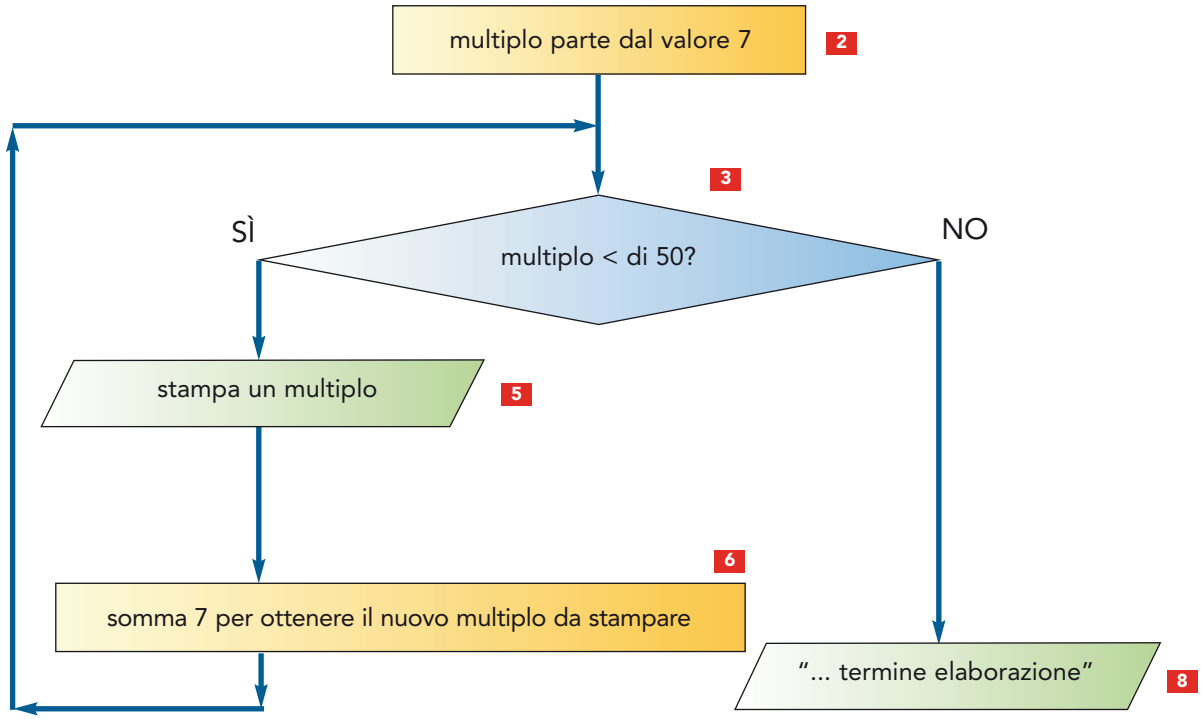
LE PAROLE DELL'INFORMATICA

Trace table È una tabella che ha tante colonne quante sono le variabili dell'algoritmo, nelle quali vengono inseriti i valori che si ottengono dall'esecuzione manuale dell'algoritmo stesso da parte di un esecutore umano: dopo avere eseguito un'istruzione si riporta su una nuova riga della trace table il valore di tutte le variabili.
Nella trace table si inserisce una colonna, generalmente la prima, in cui si scrive il numero d'ordine dell'istruzione che è stata eseguita.

Esempio 7 Multipli di 7 nei primi 50 numeri

Dobbiamo visualizzare sullo schermo i multipli di 7 che hanno valore inferiore al numero 50.
Descriviamo innanzitutto a parole il procedimento che intendiamo utilizzare: partiamo dal numero 7 e continuiamo a sommare a questo il numero 7, in modo da ottenere il primo multiplo: 7 + 7 = 14; lo visualizziamo e procediamo sommando a questo nuovamente il numero 7, ottenendo così: 14 + 7 = 21.
Ripetiamo queste operazioni finché non otteniamo un numero superiore a 50: a questo punto ci fermiamo uscendo dal ciclo e scriviamo che l'elaborazione è terminata.

Il diagramma di flusso è il seguente:



Traducendolo in metalinguaggio avremo:

```
1  inizio
2   il primo multiplo è 7
3   mentre il multiplo è < di 50 fai
4      inizio
5        stampa il multiplo
6        prepara il nuovo multiplo sommando 7
7      fine
8   scrivi('elaborazione terminata')
9  fine
```

Poiché all'interno del ciclo dobbiamo eseguire due istruzioni, è necessario racchiuderle tra un'istruzione di inizio e una di fine, in modo da non confonderle con le istruzioni di inizio e fine esterne al ciclo.

Numero di istruzione	Istruzione di test multiplo < di 50	Valore del multiplo	Risultato a video
2		7	
3	VERO		
5			7
6		14	
3	VERO		
5			14
6		21	
3	VERO		
5			21
6		28	
3	VERO		
5			28
6		35	
3	VERO		
5			35
6		42	
3	VERO		
5			42
6		49	
3	VERO		
5			49
6		56	
3	FALSO		
8		0	Fine elaborazione

ABBIAMO IMPARATO CHE...

- L'istruzione di iterazione permette di ripetere una o più istruzioni presenti in un diagramma di flusso in base al risultato di un'istruzione di test.
- Quando il numero di ripetizioni è noto a priori, cioè sappiamo quante volte le istruzioni devono essere ripetute, il ciclo prende il nome di iterazione definita.
- Quando invece il ciclo viene ripetuto un numero di volte sconosciuto a priori e termina quando si verifica una particolare condizione, l'iterazione si dice indefinita.
- Una possibile formulazione in linguaggio di progetto è la seguente:

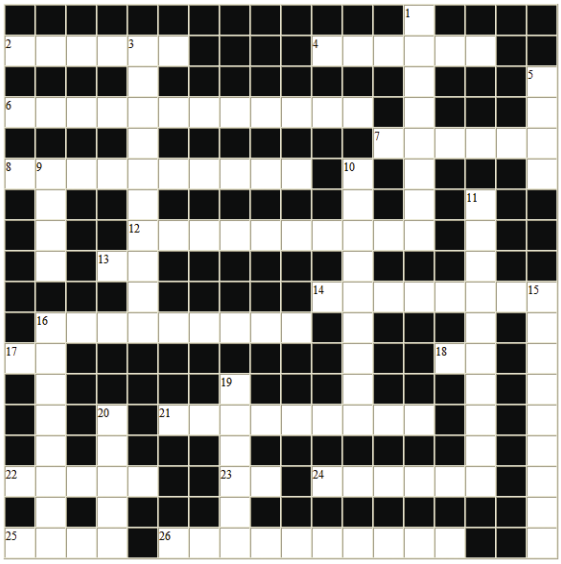
```
mentre <condizione logica> fai
   <esegui una istruzione>
```

- Nel caso in cui si debbano eseguire più istruzioni è necessario racchiuderle tra un'istruzione di inizio e una di fine, in modo da individuare le istruzioni del ciclo che devono essere ripetute.

```
mentre <condizione logica> fai
   inizio
     <esegui una istruzione>
        . . .
     <esegui una istruzione>
   fine
```

VERIFICHIAMO LE CONOSCENZE

Cruciverba



ORIZZONTALI
- 2 Insieme di due o più istruzioni
- 4 Un ramo della selezione
- 6 Regole per scrivere un codice ordinato e leggibile
- 7 Parola riservata per indicare il ciclo
- 8 Un ramo della selezione
- 12 Ripetizione di una o più istruzioni
- 13 Alternativa a SÌ
- 14 Situazione da risolvere
- 16 Istruzione che offre due alternative
- 17 Parola riservata per l'istruzione di selezione
- 18 Una possibile risposta al test
- 21 Modalità grafica di rappresentazione degli algoritmi
- 22 Iterazione, loop
- 23 Esegue i programmi
- 24 Lo è una condizione con solo due alternative
- 25 Se non è falso è...
- 26 Metodologia che permette di seguire l'evoluzione del programma

VERTICALI
- 1 Programma scritto in Pascal
- 3 Istruzione logica
- 5 Permette di individuare la soluzione di un problema
- 9 Ciclo, iterazione
- 10 Insieme di più istruzioni che traducono un algoritmo
- 11 Tipo di iterazione di cui non si sa la durata
- 15 Insieme di passi elementari, finiti, che termina in un tempo finito
- 16 Un tipo di selezione
- 19 Un tipo di selezione
- 20 Alternativa a VERO

Esercizi di completamento

Completa le seguenti frasi con la parola mancante:

problema – compilatori – selezione semplice – algoritmo – linguaggio – programma algoritmo – trace-table – loop – ciclo – iterazione definita – sorgente – indefinita – macchina – calcolatore – problema – flow chart – indentazione – condizione-logica – selezione-doppia

1. Un ..... di programmazione è un insieme di regole sintattiche.

2. I ..... trasformano il codice ..... scritto dal programmatore in linguaggio ad alto livello in codice ..... scritto in linguaggio a basso livello ed eseguibile dal .....

3. Un ..... è una situazione difficile che si deve affrontare e risolvere.

4. Un ..... è un insieme di azioni elementari che consentono di risolvere un ..... trasformando i dati iniziali nel risultato.

5. Il ..... (o diagramma di flusso) è una rappresentazione grafica dell' .....
6. Le regole utilizzate per scrivere il programma in modo incolonnato prendono il nome di regole di .....
7. Una ..... ha come possibili soluzioni solamente i due valori VERO o FALSO.
8. Nel caso in cui siano presenti istruzioni in entrambi i rami l'istruzione prende il nome di ....., mentre se le istruzioni sono presenti solo in un ramo si dice .....
9. Il termine ..... è la traduzione italiana della parola inglese .....
10. Nel caso si debbano eseguire più istruzioni di seguito è necessario racchiuderle tra ..... e fine.
11. Se il numero di ripetizioni:
    – è noto a priori siamo in presenza di una .....;
    – viene eseguito un numero di volte sconosciuto a priori l'iterazione è .....
12. Per seguire passo passo il funzionamento del programma si utilizza la .....

VERIFICHIAMO LE COMPETENZE

Problemi

1. Scrivi l'algoritmo per cucinare una colazione a base di caffè e latte. Dopo aver disegnato il diagramma di flusso traducilo in metalinguaggio.
2. Scrivi l'algoritmo che ti permette di prendere una bibita da un distributore automatico che non dà il resto. Dopo aver disegnato il diagramma di flusso traducilo in metalinguaggio.
3. Scrivi l'algoritmo che ti permette di prendere una bibita da un distributore automatico in grado di dare il resto. Dopo aver disegnato il diagramma di flusso traducilo in metalinguaggio.
4. Scrivi l'algoritmo che descrive la procedura per "la preparazione della torta alla marmellata". Dopo aver disegnato il diagramma di flusso traducilo in metalinguaggio.
5. Scrivi l'algoritmo che descrive la procedura per la preparazione di una focaccia farcita al prosciutto partendo dai suoi componenti. Dopo aver disegnato il diagramma di flusso traducilo in metalinguaggio.
6. Scrivi l'algoritmo che descrive la procedura per ottenere in prestito un libro dalla biblioteca. Dopo aver disegnato il diagramma di flusso traducilo in metalinguaggio.
7. Scrivi l'algoritmo che riceve in ingresso due numeri entrambi di tre cifre e ne effettua la somma utilizzando il metodo "in colonna", cioè una cifra alla volta partendo dalle unità e considerando il riporto. Dopo aver disegnato il diagramma di flusso traducilo in metalinguaggio.
8. Scrivi l'algoritmo che calcola l'importo da pagare per l'ingresso a un teatro sapendo che dal prezzo di ingresso di 12 euro vengono applicati i seguenti sconti:
   – bambini gratis fino a 10 anni;
   – sconto 20% ai ragazzi fino a 16 anni;
   – sconto del 50% agli anziani sopra i 75 anni.

   Dopo aver disegnato il diagramma di flusso traducilo in metalinguaggio.

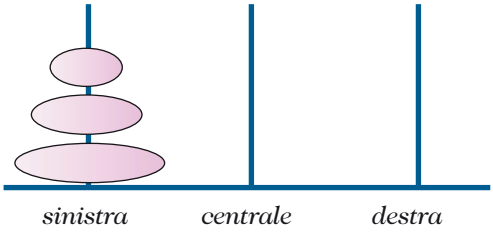
Impariamo giocando

1. Il problema della capra, del cavolo e del lupo
Una capra,un cavolo e un lupo devono essere traghettati da un contadino tra le due rive di un fiume utilizzando una barca che può portare solo due "cose" alla volta. Se vengono lasciati da soli il lupo con la capra, oppure la capra con il cavolo, i primi divorano i secondi! Descrivi l'algoritmo che permette di traghettare sani e salvi tutti quanti mediante un flow chart.

2. Il problema dei missionari
Tre missionari e tre cannibali devono attraversare un fiume utilizzando una barca che può traghettare solo due persone: se su una delle due sponde il numero dei cannibali diventa superiore al numero dei missionari, questi vengono mangiati!
Descrivi l'algoritmo che permette di traghettare sani e salvi tutti quanti mediante un flow chart.

3. La torre di Hanoi
Descrivi la procedura per la soluzione del problema della torre di Hanoi con 3 cerchi: i cerchi sono inseriti nel primo piolo a sinistra e devono essere "trasferiti" nell'ultimo piolo a destra utilizzando il piolo centrale con le uniche due regole:
– è possibile spostare un solo cerchio alla volta;
– un cerchio non può essere posizionato su un altro cerchio che abbia un diametro inferiore.

Descrivi la soluzione utilizzando un flow chart.



4. La tanica di vino
Maria e Filippo acquistano una tanica con 12 litri di vino. Se lo vogliono dividere in parti uguali, ma hanno a disposizione soltanto due recipienti: uno di 5 litri e l'altro di 9.
Descrivi con un algoritmo come possono giungere alla soluzione.

5. Con le damigiane è più difficile
Ci sono tre damigiane dalle seguenti capacità: 16 litri, 11 litri, 6 litri.
Quella da 16 litri è piena e le altre sono vuote: si vuole ottenere 8 litri d'acqua in una delle tre damigiane.
Descrivi con un algoritmo come si può giungere alla soluzione.

6. I tre mariti gelosi
Tre mariti e le rispettive tre mogli devono attraversare un fiume su una barca che può trasportare al massimo due persone alla volta.
Poiché i mariti sono molto gelosi, nessuna donna deve trovarsi mai assieme ad altri uomini se non in presenza del proprio marito.
Descrivi l'algoritmo che permette alle tre coppie di attraversare il fiume utilizzando un flow chart.

7. I cinque mariti gelosi
Questa volta abbiamo 5 coppie e la barca può trasportare al massimo 3 persone.
Descrivi l'algoritmo che permette alle tre coppie di attraversare il fiume mediante un flow chart.

8. Questione di peso
Un padre, una madre, i loro due figli e il cane devono attraversare un fiume su una barca che può trasportare al massimo un carico di 160 kg.
I genitori assieme pesano 160 kg, i due figli assieme pesano 80 kg e il cane pesa 12 kg.
Descrivi l'algoritmo che permette alle tre coppie di attraversare il fiume utlizzando un flow chart.