1 Introduzione alla programmazione

Quante volte ti è capitato di utilizzare il computer di casa per fare "girare" i tuoi programmi, per esempio un videogioco appena regalato oppure un software di videoscrittura per impaginare la relazione di storia o, ancora, un browser per navigare su Internet e visitare le pagine Web che più ti piacciono? Quasi ogni giorno, probabilmente.
Ti sei mai chiesto che cosa succede dentro il computer quando "lanci" un programma facendo doppio clic sulla sua icona? O quando carichi un gioco sulla console mentre ti prepari a impugnare il controller remoto?
Devi sapere che ogni programma è costituito da tante, tantissime istruzioni scritte una dopo l'altra, una lista delle cose da fare che il microprocessore del dispositivo utilizzato legge ed esegue in sequenza ad altissima velocità, in modo da elaborare i dati di ingresso per fornire i risultati dell'elaborazione richiesti.

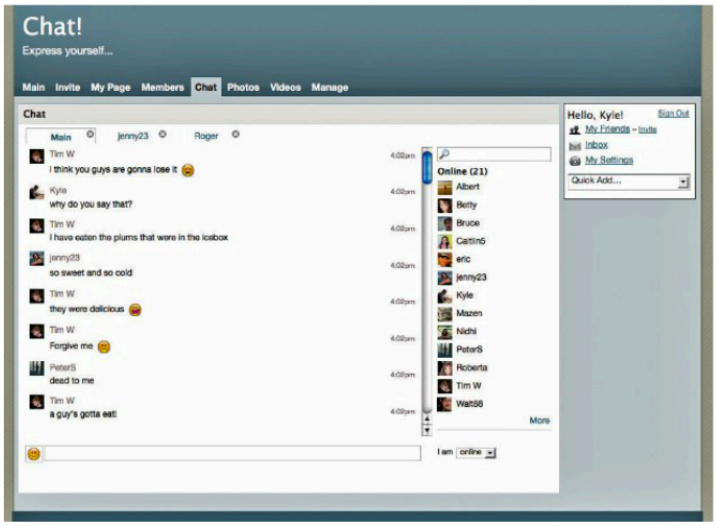
Esempio 1

Prendiamo come esempio il programma che ci permette di giocare a scacchi contro il computer.
Una volta lanciata l'esecuzione del programma, le prime istruzioni che il microprocessore deve eseguire sono quelle che hanno lo scopo di disegnare a video la scacchiera con i pezzi posizionati a inizio partita. Seguiranno poi in sequenza nel programma le istruzioni necessarie a richiedere al giocatore il suo nome e se vuole giocare con i pezzi bianchi o neri. Una volta forniti in ingresso questi dati iniziali – attraverso la tastiera e un clic del mouse – saranno richieste in ingresso al giocatore le coordinate della sua prima mossa. Anche in questo caso, attraverso il mouse, il giocatore cliccherà sul pezzo che desidera muovere e sulla casella su cui vuole eseguire lo spostamento. Il programma risponderà eseguendo le istruzioni che disegnano a video la scacchiera con il pezzo selezionato nella nuova posizione e, in successione, le istruzioni per decidere la sua prossima mossa.
E così via, mossa dopo mossa sino a fine partita o sino a quando il giocatore avrà voglia.

Videata di un programma per giocare a scacchi al computer.

Esempio 2

Un altro esempio ci può essere fornito dal programma che ci permette di "chattare" in Internet.
Una volta lanciata l'esecuzione dell'applicazione, le prime istruzioni che vengono eseguite richiedono all'utente il suo nickname e la sua password, che sono inseriti attraverso la tastiera. Dopo che le istruzioni del programma di "chat" avranno verificato la correttezza dei dati inseriti, l'utente avrà accesso al sito e potrà sapere quali sono i suoi amici on line e, attraverso il mouse e la tastiera, interagire con loro. Saranno le istruzioni del programma che, una volta selezionato il contatto al quale si vuole mandare un messaggio e riempita la casella di testo, si occuperanno di farlo pervenire all'amico attraverso la rete. E così via, messaggio dopo messaggio.



Videata di un programma di chat on line.

Possiamo quindi definire un programma come segue:

Programma

Un *programma* è un insieme finito di istruzioni che, eseguite in sequenza, permettono di elaborare i dati in ingresso per ottenere in uscita i risultati richiesti dall'elaborazione, in modo da risolvere un determinato problema.

Nell'**esempio 1**, il problema è la partita a scacchi fra utente e computer, i dati in ingresso sono le mosse che il giocatore immette volta per volta, i risultati sono le contromosse, sino a fine partita. Nell'**esempio 2**, il problema da gestire è lo scambio di messaggi fra utenti collegati in chat, i dati in ingresso sono i nickname del mittente e del destinatario e il testo dei messaggi da inviare, i risultati sono i messaggi ricevuti dagli altri utenti connessi inviati in risposta.
In questo capitolo vogliamo iniziare a capire come si "sviluppa" un programma. Naturalmente non partiremo dai programmi che abbiamo utilizzato per gli esempi precedenti, ma da programmi molto più semplici. Inoltre, per semplificare ulteriormente lo studio, chiameremo in generale "elaboratore" il computer a disposizione – portatile, desktop, netbook, smartphone, tablet o console di gioco che sia – con il quale comunichiamo per mezzo di una tastiera, di un mouse, di un touch screen o di un controller remoto.

L'elaboratore è quindi l'esecutore delle istruzioni dei nostri programmi, istruzioni che elaborano i dati in ingresso per fornire, al termine dell'elaborazione, i risultati in uscita (fig. 1).

Figura 1

Supponiamo, per esempio, di voler sviluppare un programma per il calcolo della media aritmetica di 3 numeri, cioè un programma che richiede all'utente in ingresso da tastiera 3 numeri e, al termine degli input, ne calcola la media aritmetica e la visualizza a video. Se fornissimo in ingresso i numeri 10, 6 e 5, la loro media aritmetica sarebbe data dalla somma $10 + 6 + 5 = 21$ che divisa per 3 ci fornirebbe il risultato 7 (fig. 2).

Figura 2

Quello che ora vogliamo fare è passare *dalla descrizione a parole del problema* "calcolare la media aritmetica di 3 numeri" *all'insieme di istruzioni che costituiscono il programma*: in questo modo, fornito all'elaboratore il programma e i 3 numeri in input da tastiera, l'elaboratore può calcolare e fornire in uscita la media richiesta.

2 Dal problema al programma

Per passare dal problema da risolvere al programma che lo risolve "automaticamente" occorre quindi **formalizzare il problema** che, espresso inizialmente in linguaggio naturale, deve passare attraverso varie *fasi di trasformazione* che permettano di focalizzare meglio gli obiettivi e il modo per raggiungerli, fino a ottenere il programma che risolve il problema di partenza.

Figura 3
Fasi del processo per passare dal problema al programma.

In **figura 3** sono riassunte alcune tra le principali fasi che costituiscono il processo di formalizzazione e che applicheremo negli esempi successivi.
Il primo passo da affrontare consiste nell'**analisi** approfondita del problema che si vuole risolvere. In questa fase occorre porre molta attenzione a ciò che ci viene richiesto, per essere sicuri di aver capito esattamente *qual è l'obiettivo che ci poniamo e che cosa abbiamo a disposizione per raggiungerlo*. Più dettagliatamente, la fase di analisi può a sua volta essere suddivisa in tre *sottofasi*:

Il primo passo dell'analisi consiste, dopo una lettura approfondita del problema da risolvere, nell'individuazione dei **dati in ingresso** (**input**) al problema: si tratta di tutte le informazioni che devono essere fornite in ingresso al programma che si sta sviluppando, con gli eventuali vincoli di integrità. Per **vincoli di integrità** si intendono tutte quelle condizioni che gli input devono rispettare per essere accettati dal programma. Se, per esempio, in un videogioco possiamo scegliere di spostarci in un labirinto verso destra o verso sinistra, con la tastiera non potremo che premere i tasti freccia o  perché saranno gli unici tasti considerati validi (**fig. 4**).

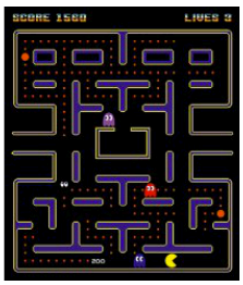


Figura 4
Lo screenshot del famosissimo labirinto di PacMan, dove ci si muove con i tasti freccia.

Un ulteriore esempio potrebbe essere il seguente: supponiamo di dover fornire in ingresso la nostra data di nascita nel formato giorno/mese/anno per iscriverci a un sito. I vincoli che questi input devono rispettare per essere accettati dal programma impongono che il valore che indica il mese sia compreso tra 1 e 12, il valore che indica il giorno sia compreso tra 1 e il numero di giorni del mese corrispondente, mentre per il valore che indica l'anno il vincolo dipenderà dal problema che stiamo affrontando.
Il secondo passo dell'analisi consiste nell'individuazione dei **dati in uscita** (**output**), cioè dei risultati attesi dall'elaborazione dei dati in ingresso da parte del programma. Anche per gli output devono essere individuati gli eventuali vincoli ai quali essi devono sottostare per essere considerati validi.

Se, per esempio, abbiamo raggiunto il punteggio necessario per passare al livello successivo di un videogioco e invece di cambiare schema ritorniamo a quello precedente, significa che il programma non ha rispettato il vincolo posto al risultato che abbiamo ottenuto (**fig. 5**).

Figura 5
Candy Crush offre ai suoi giocatori centinaia di livelli di gioco.

O ancora, se abbiamo un programma che deve ordinare alfabeticamente (dalla A alla Z) un elenco di cognomi e in uscita otteniamo invece come risultato l'elenco ordinato in modo decrescente (dalla Z alla A), allora significa che l'ordinamento effettuato dal programma non ha rispettato il vincolo posto all'output.
L'ultima sottofase dell'analisi consiste nell'individuazione delle **relazioni tra i dati in input e quelli in output** (**I/O**), cioè dei *collegamenti logici esistenti tra le informazioni in ingresso e quelle in uscita dal programma*.
La relazione tra I/O può essere una formula matematica che lega i dati in ingresso al risultato atteso, come nel caso del calcolo della media aritmetica, oppure un modello logico che permette di approfondire il legame esistente tra I/O per facilitare poi le fasi successive del processo di formalizzazione.
Per capire meglio ciò che si intende per relazione tra i dati in input e quelli in output, applichiamo la metodologia vista ad alcuni semplici problemi, di cui sviluppiamo l'analisi.

Problema 1

Calcolare la media aritmetica di 3 numeri in ingresso.

ANALISI

Dati in input	I 3 valori numerici di cui si vuole calcolare la media aritmetica.
Dati in output	La media aritmetica.
Relazione tra I/O	La media aritmetica è data dalla somma dei 3 valori in ingresso divisa per 3: $$Media = \frac{1°valore+2°valore+3°valore}{3}$$

Problema 2

Forniti in ingresso al programma la base e l'altezza di un triangolo, calcolarne l'area.

ANALISI

Dati in input	La lunghezza della base e la lunghezza dell'altezza del triangolo; il vincolo da porre è accettare in ingresso solo valori maggiori di 0.
Dati in output	L'area del triangolo.
Relazione tra I/O	L'area del triangolo si ottiene moltiplicando la base per l'altezza e dividento per 2: $$Area = \frac{Base*Altezza}{2}$$ dove il simbolo * indica, nel campo informatico, la moltiplicazione.

Problema 3

Fornito in ingresso al programma il numero di lati (si suppone compreso tra 3 e 5) di un poligono regolare e la lunghezza del lato del poligono, indicare di quale poligono si tratta e calcolarne il perimetro.

ANALISI

Dati in input	Il numero di lati del poligono regolare e la lunghezza del lato del poligono. I vincoli da porre sono che il numero di lati sia un valore intero positivo compreso tra 3 e 5 e la lunghezza del lato sia un valore maggiore di 0.
Dati in output	La stampa a video del tipo di poligono regolare e del suo perimetro.
Relazione tra I/O	– Se il numero di lati è uguale a 3 allora è un "Triangolo equilatero"; – se il numero di lati è uguale a 4 allora è un "Quadrato"; – se il numero di lati è uguale a 5 allora è un "Pentagono". Il perimetro si calcola moltiplicando il numero di lati per la lunghezza del lato: *Perimetro = NumeroLati * LunghezzaLato*

Problema 4

Determinare il valore più grande tra 3 valori numerici in ingresso.

ANALISI

Dati in input	I 3 valori numerici: si può supporre di non avere vincoli, quindi di accettare in ingresso sia numeri interi sia numeri con la virgola, positivi e negativi.
Dati in output	Il valore più grande fra i 3.
Relazione tra I/O	– Se il 1° valore è maggiore del (o uguale al) 2° valore e se il 1° valore è maggiore del (o uguale al) 3° valore allora il valore maggiore è il 1°; – se il 2° valore è maggiore del (o uguale al) 1° valore e se il 2° valore è maggiore del (o uguale al) 3° valore allora il valore maggiore è il 2°; – se il 3° valore è maggiore del (o uguale al) 1° valore e se il 3° valore è maggiore del (o uguale al) 2° valore allora il valore maggiore è il 3°.

Nei problemi analizzati, dopo aver individuato i dati in ingresso e quelli in uscita al problema si è cercata la relazione che li lega; nei primi due problemi il legame tra I/O è fornito dalla formula matematica, mentre negli ultimi due problemi la relazione tra I/O fa riferimento al confronto logico tra i dati in ingresso per arrivare a individuare i dati in uscita.
Al termine della fase di analisi, il problema da risolvere dovrebbe risultare più chiaro e dettagliato, pronto quindi per la più delicata e complessa fase successiva, in cui si giunge a rappresentare il problema di partenza attraverso l'**algoritmo risolutore** corrispondente.

5 Le fasi di simulazione e codifica dell'algoritmo

Una volta sviluppato l'algoritmo, procedendo nel processo di formalizzazione si arriva alla **simulazione** dell'algoritmo, fase in cui si controlla se la sequenza di azioni ricavate è funzionante. Esistono vari metodi per simulare un algoritmo ma tutti, in pratica, consistono nell'*eseguire virtualmente le azioni dell'algoritmo* secondo la sequenza data, supponendo di avere dei dati in ingresso su cui basare la simulazione. Al termine della simulazione, controllando se il risultato ottenuto è proprio quello che ci si aspettava, è possibile capire se l'algoritmo è funzionante oppure se ci sono errori e dove sono eventualmente localizzati. In caso di errori, si dovrà ritornare alla fase precedente e rivedere l'algoritmo per eliminare eventuali malfunzionamenti. Durante la simulazione è inoltre possibile evidenziare se l'algoritmo presenta degli inconvenienti o se può essere ulteriormente migliorato, se ha ancora bisogno di qualche altra istruzione aggiuntiva per eventuali casi non considerati o altro ancora.

Prendiamo come esempio l'algoritmo associato al problema:

Calcolare la media aritmetica di 3 numeri in ingresso

e vediamone la simulazione passo per passo, utilizzando una tabella con tante colonne quante sono le variabili che sono manipolate dalle istruzioni dell'algoritmo, più una colonna che serve per indicare le azioni di cui si simula l'esecuzione (**tab. 2**)

Tabella 2

Azione	A	B	C	Media
1. Inizio				
2. Ricevi in ingresso da tastiera i valori delle variabili *A*, *B*, *C*	Input: 10	Input: 7	Input: 16	
3. *Media* $\leftarrow \frac{A+B+C}{3}$				11
4. Stampa a video il valore della variabile *Media*				Output: 11
5. Fine				

Consideriamo ora l'algoritmo associato al problema:

Forniti in ingresso al programma la base e l'altezza di un triangolo, calcolarne l'area

e vediamone la simulazione passo per passo, utilizzando anche in questo caso una tabella con tante colonne quante sono le variabili manipolate dalle istruzioni dell'algoritmo, più una che indica le azioni (**tab. 3**).

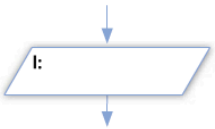
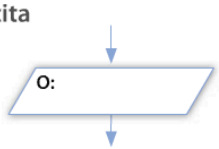
7 Gli schemi di flusso

Nei paragrafi precedenti abbiamo sviluppato alcuni semplici algoritmi utilizzando frasi in linguaggio italiano per descrivere le azioni da compiere a ogni passo. Per esempio, abbiamo usato l'espressione *Salta all'azione…* per indicare il passaggio a un'istruzione non in sequenza rispetto a quella che si sta eseguendo, oppure abbiamo usato le parole *Se…allora…altrimenti* per indicare il fatto che doveva essere testata una condizione per decidere quale azione intraprendere in base al risultato del controllo. Abbiamo cioè espresso gli algoritmi facendo uso della nostra lingua e numerando le istruzioni per sapere a quale punto dell'algoritmo eventualmente "saltare". Nella **tabella 4** sono elencate le istruzioni che più frequentemente compaiono durante lo sviluppo di un algoritmo e che abbiamo già utilizzato nei precedenti esempi.

Tabella 4

Tipo di istruzione	Significato		Esempio
Azione	Istruzione che compie determinate azioni sui dati per produrre risultati intermedi o finali		*Area Triangolo* ← $\frac{BaseTriangolo*AltezzaTriangolo}{2}$ *Perimetro* ← *NumeroLati* * *Lunghezza Lato*
Controllo (Condizionale)	Istruzione che permette di prendere una decisione in base al risultato di una condizione, in modo da sapere se seguire un cammino oppure un altro		Se *NumeroLati* = 3 allora … altrimenti … Se $B \geqslant A$ allora …
Comunicazione (Trasmissione)	Istruzione di *ingresso*: permette la comunicazione da utente a elaboratore		Ricevi in ingresso il valore della variabile *NumeroLati*
	Istruzione di *uscita*: permette la comunicazione da elaboratore a utente		Stampa a video il valore della variabile *Perimetro*
Salto	Permette di alterare l'esecuzione sequenziale delle istruzioni dell'algoritmo per saltare ad altre parti del programma	Salto *Condizionato*: associato al verificarsi di una condizione	Se *NumeroLati* = 4, allora…
		Salto *Incondizionato*: non è associato al verificarsi di una condizione	Salta all'azione 15
Inizio algoritmo	Si usa a inizio algoritmo		Inizio
Fine algoritmo	Si usa a fine algoritmo		Fine

Tabella 5

Tipo di istruzione	Simbolo	Significato
Azione		*Blocco di Azione:* esegue l'azione descritta all'interno del rettangolo
Controllo (Condizionale)		*Blocco di Controllo:* verifica la condizione e se il risultato è vero passa a eseguire le istruzioni sul ramo corrispondente a vero altrimenti passa a eseguire le istruzioni sul ramo corrispondente a falso
Comunicazione (Trasmissione)	di ingresso 	*Blocco di Input dati:* chiede in ingresso all'utente un valore che verrà memorizzato in una variabile in memoria
	di uscita 	*Blocco di Output dati:* fornisce in uscita all'utente un valore che verrà visualizzato a video
Salto	È rappresentato da una freccia che si innesta in un'altra freccia nel punto dell'algoritmo cui si deve saltare 	*Salto condizionato o incondizionato:* vado a eseguire l'istruzione indirizzata dal flusso delle frecce
Inizio algoritmo		*Blocco di Inizio algoritmo*
Fine algoritmo		*Blocco di Fine algoritmo*

8 Primi esempi di schemi di flusso

Riprendiamo ora gli stessi esempi visti in precedenza e rappresentiamoli mediante schemi di flusso. Ricordiamo che il processo di formalizzazione per il passaggio da problema a programma prevede una prima fase di analisi del problema che precede quella di sviluppo dell'algoritmo.

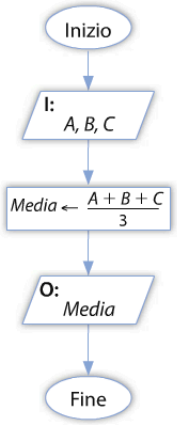
Problema 1

Calcola la media aritmetica di 3 numeri in ingresso.

ANALISI

Dati in input	*A*, *B*, *C*
Dati in output	*Media*
Relazione tra I/O	$Media = \frac{A + B + C}{3}$

ALGORITMO



Problema 2

Forniti in ingresso al programma la base e l'altezza di un triangolo, calcolarne l'area.

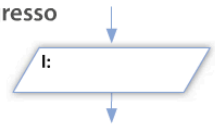
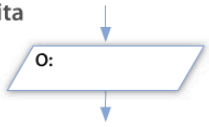
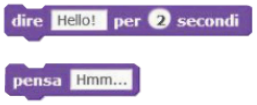
ANALISI

Dati in input	*BaseTriangolo* (con *BaseTriangolo* > 0) *AltezzaTriangolo* (con *AltezzaTriangolo* >0)
Dati in output	*AreaTriangolo*
Relazione tra I/O	$AreaTriangolo = \frac{BaseTriangolo * AltezzaTriangolo}{2}$

9 Dai simboli degli schemi di flusso ai blocchi di *Scratch*

Dovendo codificare un algoritmo mediante il linguaggio di programmazione *Scratch*, prendiamo nuovamente in considerazione la tabella 5 con l'elenco dei simboli utilizzati per lo sviluppo degli schemi di flusso, associando a ciascuno di essi alcuni fra i blocchi corrispondenti di *Scratch* (tab. 6).

Tabella 6

Tipo di istruzione	Simbolo	Blocco *Scratch*
Azione		
Controllo (Condizionale)		
Comunicazione (Trasmissione)	di ingresso 	
	di uscita 	
Salto		Tratteremo questi blocchi solo successivamente, quando parleremo di *cicli*
Inizio algoritmo		
Fine algoritmo		

\


quando si clicca su
chiedi Inserisci un primo valore e attendi
porta A a risposta
chiedi Inserisci un secondo valore e attendi
porta B a risposta
chiedi Inserisci un terzo valore e attendi
porta C a risposta
porta Media a A + B + C / 3
dire unione di La media è e Media
arresta questo script