

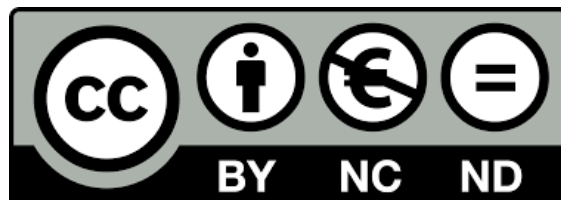
Introduzione alla programmazione

ITS - Umbria

A.S. 2022-23

MATERIA: Fondamenti di informatica

Docente: prof. Paolo Bernardi



Introduzione alla programmazione

Utilizzare il proprio smartphone, tablet, PC per chattare su Snapchat o comunicare con Whatsapp, effettuare ricerche con Google su Internet, vedere e pubblicare video su YouTube, condividere foto su Facebook, giocare con Pokemon Go e molto altro ancora...Sono tutte azioni che compiamo quasi quotidianamente e ci accompagnano in vari momenti della giornata:

a scuola, al lavoro, sui mezzi di trasporto, durante il tempo libero.

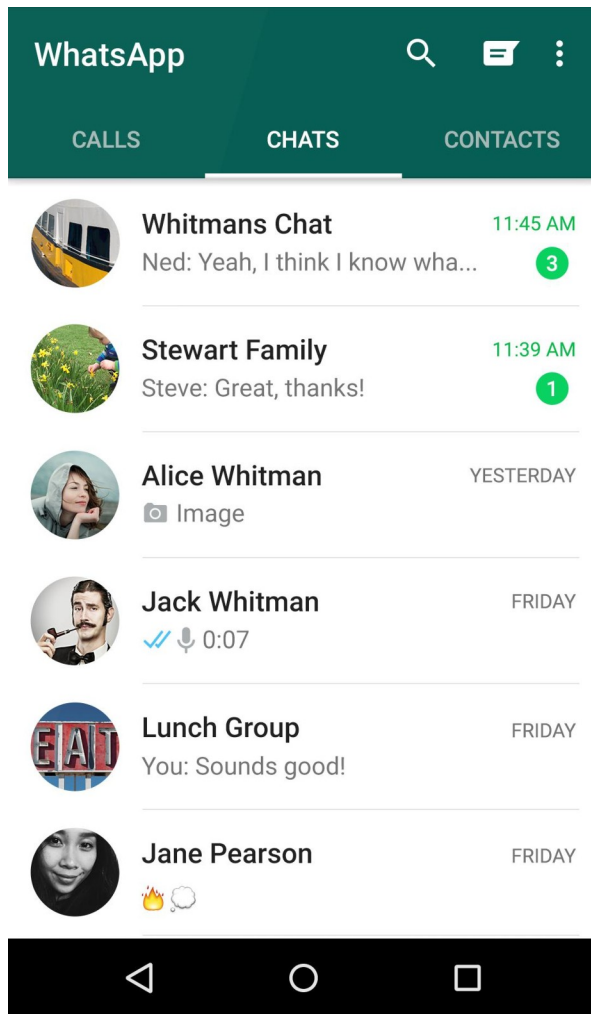
Facciamo parte di una grande comunità che si serve del digitale in tutte le sue forme e in tutti i campi in cui trova applicazione e ogni giorno la tecnologia offre qualche novità che rende obsolete quelle di pochi mesi prima.

Introduzione alla programmazione

Ma cosa succede in realtà all'interno di un dispositivo quando “si lancia” un programma, facendo “doppio clic” sulla sua icona, oppure quando si tocca l'icona di una app su di un dispositivo “mobile”?

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 ad altissima velocità, in modo da elaborare i dati di ingresso per fornire in uscita i risultati dell'elaborazione richiesti.

Introduzione alla programmazione



ESEMPIO

Ci offre un altro esempio la app che ci permette di “chattare” in Internet con il nostro smartphone. Non appena viene lanciata l'esecuzione dell'applicazione, le prime istruzioni eseguite sono quelle che ci presentano l'elenco dei nostri contatti con l'indicazione dei messaggi arrivati. Se si decide di leggere i messaggi, dopo aver selezionato il contatto che ci interessa, le istruzioni dell'applicazione ci mostrano la sequenza cronologica dei messaggi che compongono la conversazione. Nel caso in cui decidessimo di rispondere, le istruzioni del programma ci mostreranno sul display una casella di testo dove inserire il messaggio e si occuperanno di farlo pervenire all'amico attraverso la rete. E così via, messaggio dopo messaggio.

Introduzione alla programmazione

Possiamo quindi definire un programma come segue:

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.

```
COMMODORE BASIC V7.0 122365 BYTES FREE
(C)1986 COMMODORE ELECTRONICS, LTD.
(C)1977 MICROSOFT CORP.
ALL RIGHTS RESERVED

READY.
MONITOR

MONITOR
PC SR AC XR YR SP
; FB000 00 00 00 00 F9
M F41B0
>F41B0 D0 F8 20 69 92 C8 C0 96 D0 E3 60 93 0D 40 20 43:
>F41C0 4F 4D 4D 4F 44 4F 52 45 20 42 41 53 49 43 20 56:
>F41D0 37 2E 30 20 31 32 32 33 36 35 20 42 59 54 45 53:
>F41E0 20 46 52 45 45 0D 40 20 20 20 28 43 29 31 39 38:
>F41F0 36 20 43 4F 4D 4D 4F 44 4F 52 45 20 45 4C 45 43:
>F4200 54 52 4F 4E 49 43 53 2C 20 4C 54 44 2E 0D 40 20:
>F4210 20 20 20 20 20 20 20 20 20 28 43 29 31 39 37 37:
>F4220 4D 49 43 52 4F 53 4F 46 54 20 43 4F 52 50 2E 0D:
>F4230 40 20 20 20 20 20 20 20 20 20 20 20 41 4C 4C 20:
>F4240 52 49 47 48 54 53 20 52 45 53 45 52 56 45 44 0D:
>F4250 00 A2 11 BD 67 42 9D 00 03 CA 10 F7 A9 78 8D FC:
>F4260 02 A9 4C 8D FD 02 60 3F 4D C6 4D 0D 43 51 51 A2:
```

Introduzione alla programmazione

Nell'esempio precedente, il problema da gestire è lo scambio di messaggi tra utenti collegati in chat, i dati in ingresso sono il nickname del destinatario e il testo del messaggio da inviare, i risultati sono i messaggi ricevuti dagli altri utenti connessi in chat.

Chiameremo in generale "elaboratore" il computer a disposizione (portatile, desktop, netbook, smartphone, tablet o console di gioco, ecc.) con il quale comunichiamo per mezzo di una tastiera, di un mouse, di un touchscreen o di un controller remoto.



Introduzione alla programmazione

I problemi da risolvere, appositamente codificati, verranno fatti eseguire da una macchina particolare, chiamata “elaboratore”.

Chiameremo in generale "elaboratore" il computer a disposizione (portatile, desktop, netbook, smartphone, tablet o console di gioco, ecc.) con il quale comunichiamo per mezzo di una tastiera, di un mouse, di un touchscreen o di un controller remoto.



Introduzione alla programmazione



FIGURA 1

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].

Introduzione alla programmazione

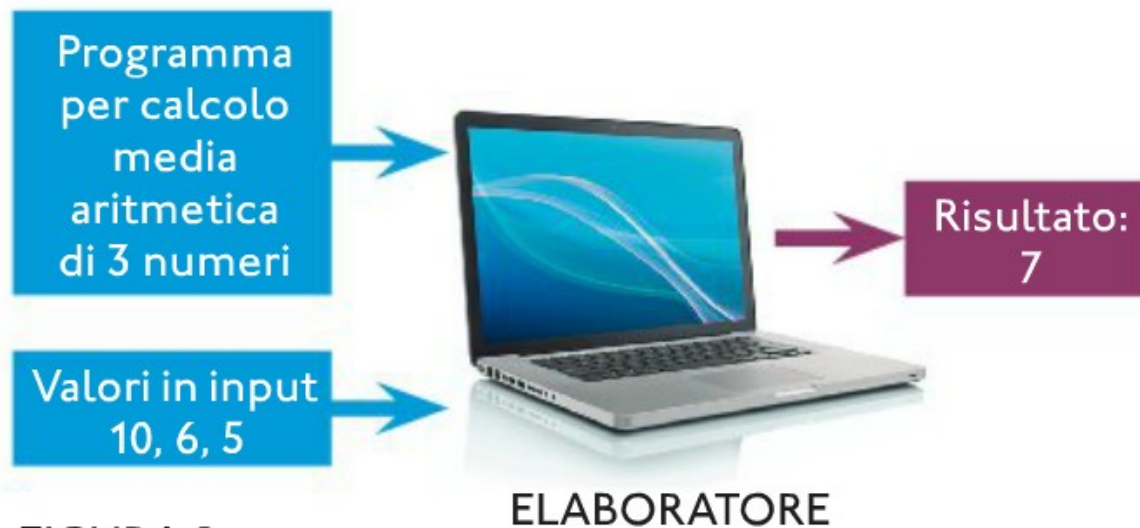


FIGURA 2

Supponiamo, per esempio, di voler sviluppare un semplice 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.

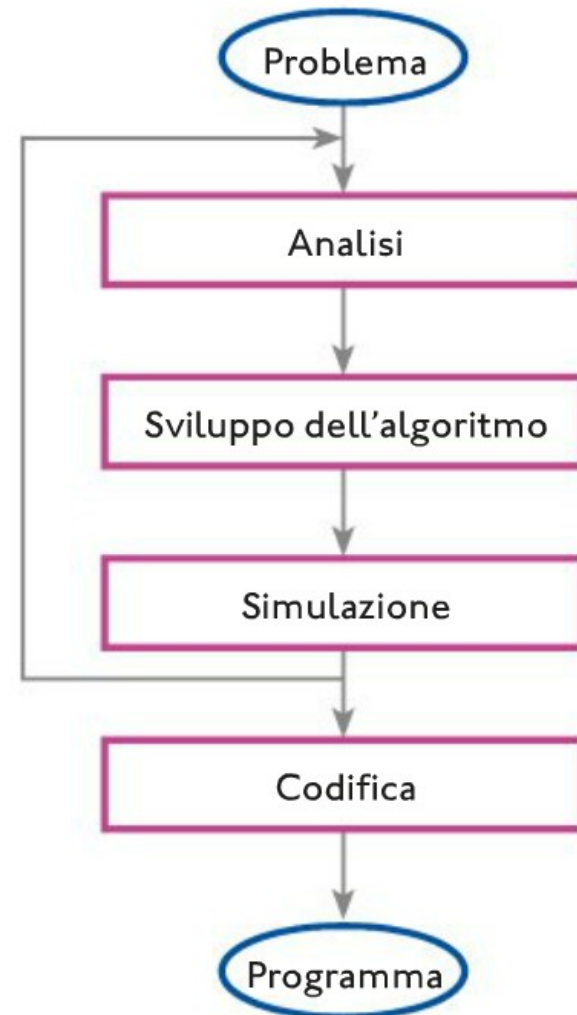
[FIG. 2].

Introduzione alla programmazione

Per passare dal problema da risolvere al programma che lo risolve “automaticamente”, occorre quindi formalizzare il problema.

Espresso inizialmente in linguaggio naturale, il problema di partenza 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 lo risolve.

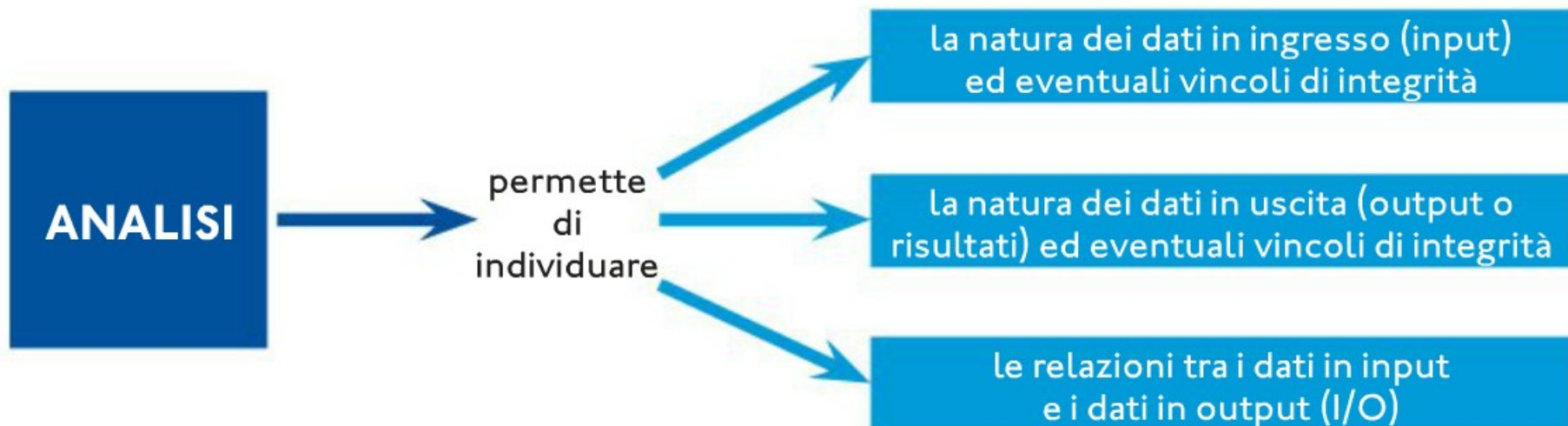
Nello schema a destra sono riassunte alcune tra le principali fasi che costituiscono il processo di formalizzazione.



Introduzione alla programmazione

La fase di **ANALISI** del problema

Il primo passo da affrontare consiste nell'analisi approfondita del problema da risolvere. In questa fase occorre porre molta attenzione a ciò che viene richiesto, per essere sicuri di aver capito esattamente qual è l'**obiettivo** che ci poniamo e quali **strumenti** abbiamo a disposizione per raggiungerlo. Più dettagliatamente, la fase di analisi può essere suddivisa in tre sottofasi e schematizzata come di seguito:



Introduzione alla programmazione

ANALISI: dati in ingresso (input)

Il primo passo dell'analisi consiste 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.

I **vincoli di integrità** sono tutte quelle condizioni che gli input devono rispettare per essere accettati dal programma.

Un **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 dipende dal problema che si sta affrontando.

Introduzione alla programmazione

ANALISI: dati in uscita (output)

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.

Un esempio: se un programma deve **ordinare alfabeticamente in senso crescente** (dalla A alla Z) un elenco di cognomi e in uscita si ottiene, invece, come risultato l'elenco ordinato in modo decrescente (dalla Z alla A), significa che l'ordinamento effettuato dal programma non ha rispettato il vincolo posto all'output.

Introduzione alla programmazione

ANALISI: relazione tra i dati

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.

Introduzione alla programmazione

ESEMPIO n.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:

$$\text{Media} = \frac{1^{\circ} \text{ valore} + 2^{\circ} \text{ valore} + 3^{\circ} \text{ valore}}{3}$$

Introduzione alla programmazione

ESEMPIO n.2

P R O B L E M A

Fornita in ingresso al programma la lunghezza del lato di un quadrato, calcolarne il perimetro e l'area.

ANALISI

Dati in input La lunghezza del lato del quadrato; il vincolo da porre è accettare in ingresso solo valori maggiori di 0.

Dati in output Il perimetro e l'area del quadrato.

Relazione tra I/O Il perimetro del quadrato si calcola moltiplicando la lunghezza del lato per 4 mentre l'area si calcola moltiplicando la lunghezza del lato per se stessa:

$$\text{Perimetro} = \text{Lato} * 4$$

$$\text{Area} = \text{Lato} * \text{Lato}$$

dove il simbolo * indica, nel campo informatico, la moltiplicazione.

Introduzione alla programmazione

ESEMPIO n.3

PROBLEMA

Calcolare e applicare lo sconto del 20% a un prezzo fornito in ingresso solo se questo supera 100 €.

ANALISI

Dati in input Il prezzo; il vincolo da porre è accettare in ingresso solo valori maggiori di 0.

Dati in output Il prezzo finale.

Relazione tra I/O – Se il prezzo è maggiore di 100 €, allora si calcola lo sconto del 20%:

$$\text{Sconto} = \frac{\text{Prezzo} * 20}{100}$$

quindi si calcola il prezzo finale dato da:

$$\text{Prezzo finale} = \text{Prezzo} - \text{Sconto}$$

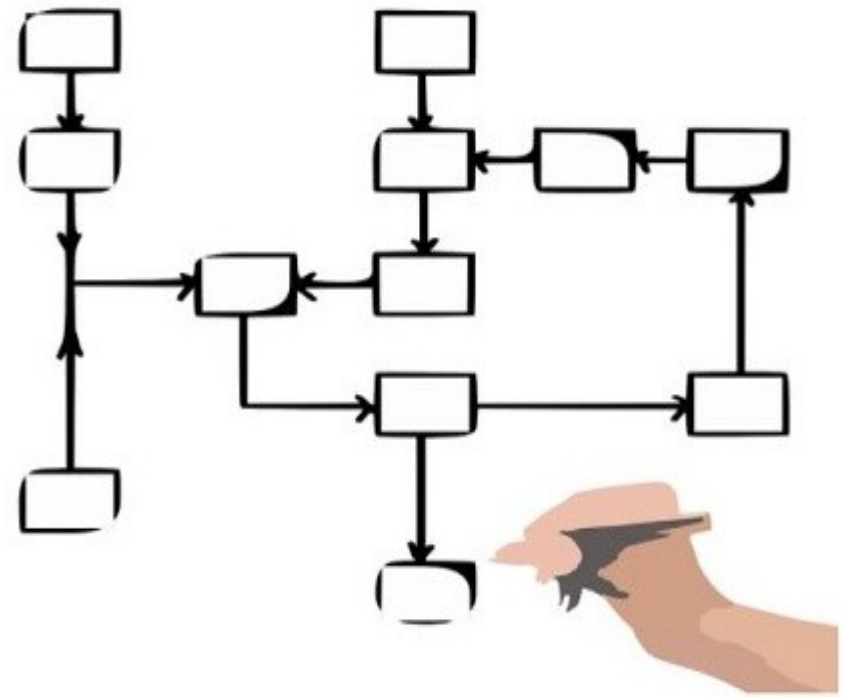
– Se il prezzo è minore o uguale a 100 €, allora il prezzo finale è uguale al prezzo fornito in ingresso (non viene cioè applicato alcuno sconto):

$$\text{Prezzo finale} = \text{Prezzo}$$

Introduzione alla programmazione

La fase di **SVILUPPO** dell'algoritmo

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**.



Introduzione alla programmazione

SVILUPPO: Sviluppo dell'algoritmo

Questa fase, partendo dai risultati ricavati nell'analisi svolta precedentemente e, in particolar modo, dalle relazioni tra I/O, è indirizzata a **individuare un metodo ottimale di risoluzione del problema di partenza**, cioè una **sequenza di azioni** che, seguendo le **indicazioni date dalla relazione tra I/O**, **permetta di trasformare i dati in ingresso in risultati**, cioè in dati di uscita.

Introduzione alla programmazione

SVILUPPO: Sviluppo dell'algoritmo

L'elaboratore non è in grado di trovare autonomamente il metodo risolutore: il suo compito, infatti, consiste nell'eseguire in sequenza le istruzioni del programma.

Sta al programmatore il compito di “indicare” all'elaboratore il metodo risolutore del problema, le azioni da eseguire e le regole da seguire per ottenere il risultato voluto, cioè l'algoritmo risolutore del problema di partenza.

Quindi il programmatore è colui che “trasmette” all'elaboratore il proprio ragionamento sotto forma di algoritmo che, tradotto in un programma, potrà essere eseguito.

Introduzione alla programmazione

ESEMPIO n.4

ESEMPIO

$$\begin{array}{r} 12^18 + \\ 357 = \\ \hline \dots 85 \end{array}$$

L'addizione di due numeri viene eseguita applicando un algoritmo in cui i due addendi sono i dati in ingresso e la somma è il risultato finale. Iniziando da destra, si sommano le 2 cifre corrispondenti alle unità. Se la somma ottenuta è maggiore di 9 allora si sottrae 10 tenendo il resto ottenuto e ricordando poi il riporto nella prossima somma. Si prosegue spostandosi verso sinistra sommando le cifre corrispondenti alle decine – con l'eventuale riporto – e si applica lo stesso criterio già visto ciclicamente sino a fine addizione.

Una semplice addizione richiede l'esecuzione di un algoritmo che ripete ciclicamente la somma di coppie di cifre da destra verso sinistra.

Introduzione alla programmazione

SVILUPPO: *Sviluppo dell'algoritmo*

Il concetto di algoritmo è applicato in molte discipline, ma nell'informatica, e in particolare nella programmazione, assume una notevole importanza: in questa delicata **fase del processo di formalizzazione** sono infatti richieste **capacità di astrazione** tali da individuare non solo l'algoritmo risolutore del problema, ma soprattutto l'**algoritmo risolutore ottimale**, che segue criteri di generalità e di sintesi.

Un algoritmo è un insieme finito di azioni che risolvono un determinato problema, trasformando i dati di input in dati di output (o risultati) attraverso le relazioni esistenti tra input e output.

Introduzione alla programmazione

ESEMPIO n.1

Vediamo come esempio l'algoritmo risolutore del problema n.1

1. Inizio.
2. Ricevi in ingresso da tastiera i tre valori.
3. Calcola la media sommando i tre valori in ingresso e dividendo la somma ottenuta per 3.
4. Stampa a video la media aritmetica calcolata.
5. Fine.

La sequenza di passi da compiere per ricavare un algoritmo che risolva il problema è chiara e le azioni, eseguite una dopo l'altra, ci portano a calcolare la media aritmetica in modo molto semplice e diretto.

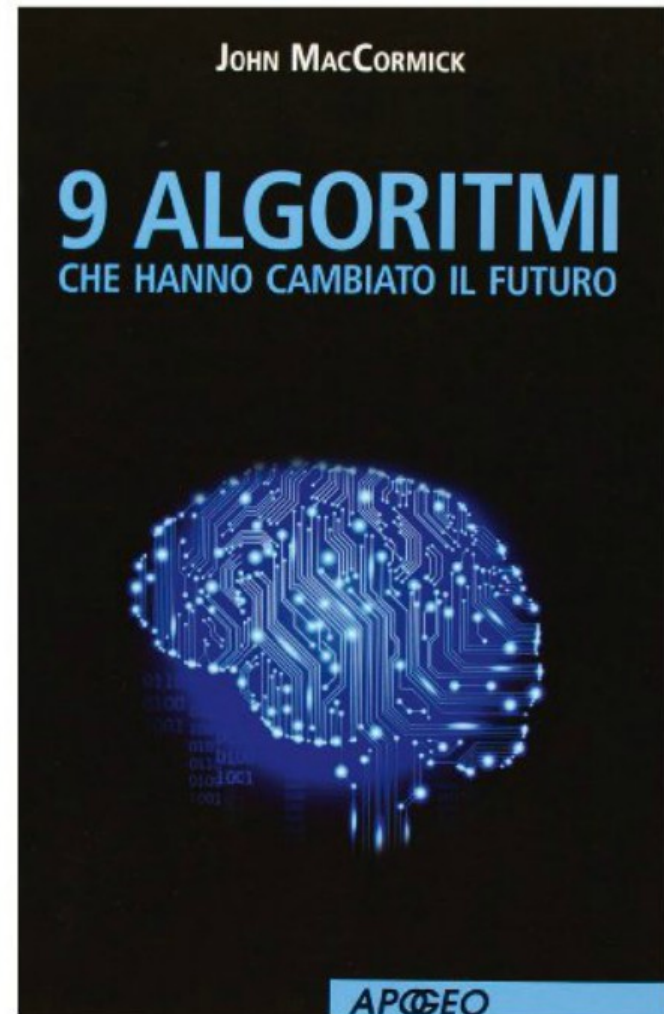
Se infatti fornissimo in ingresso da tastiera i tre valori 15, 5, 40, si otterrebbe come risultato: 20 (infatti, $15 + 5 + 40 = 60$ e $60 : 3 = 20$) che rappresenta la media aritmetica cercata.

Introduzione alla programmazione

Il futuro in nove algoritmi

Ogni giorno, attraverso l'utilizzo degli strumenti tecnologici a nostra disposizione, compiamo azioni che a noi sembrano del tutto normali non accorgendoci della complessità che le caratterizza. Cerchiamo informazioni in rete per mezzo di motori di ricerca, archiviamo dati su potenti hard disk o su server remoti, facciamo acquisti on line, comprimiamo file di grandi dimensioni, chattiamo con gli amici attraverso i social network. Tutto ci sembra a portata di computer... ma solamente grazie a idee fondamentali, trasformate in algoritmi risolutori, è possibile eseguire dei compiti impensabili sino a qualche anno fa.

Degli algoritmi che hanno cambiato i nostri comportamenti quotidiani ci parla John MacCormick, docente universitario di Informatica al Dickinson College in Pennsylvania, nel suo saggio dal titolo *9 algoritmi che hanno cambiato il futuro*. Google non potrebbe essere così veloce nelle ricerche se non esistesse l'algoritmo del *PageRank*, un lettore musicale di file MP3 non potrebbe contenere centinaia di canzoni se non ci fossero gli algoritmi di compressione, l'e-commerce non esisterebbe se non esistesse la crittografia e gli algoritmi che la implementano... Insomma, senza algoritmi il nostro mondo, e il nostro futuro, non sarebbero quello che sono e quello che saranno.



Introduzione alla programmazione

Le fasi di simulazione e codifica dell'algoritmo

Una volta sviluppato l'algoritmo, procedendo nel processo di formalizzazione si arriva alla simulazione dell'algoritmo; questa è la 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.

Introduzione alla programmazione

Le fasi di simulazione e codifica dell'algoritmo

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 compaiono **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 bisogno di istruzioni aggiuntive** per eventuali casi non considerati o altro ancora.

Introduzione alla programmazione

Le fasi di simulazione e codifica dell'algoritmo

Inoltre, al fine di conservare “memoria” degli stati intermedi di evoluzione dell'algoritmo, le informazioni processate vengono allocate all'interno di “contenitori logici”, chiamati **variabili**.

Le **variabili** sono gli oggetti utilizzati dalle istruzioni del programma. Esse **risiedono nella memoria dell'elaboratore** e corrispondono a contenitori dei valori che sono elaborati durante l'esecuzione del programma. Alle variabili è associato un nome, detto **identificatore**, che le individua univocamente all'interno del programma. Sono chiamate variabili perché a loro viene **associato un valore che può cambiare durante l'esecuzione del programma**.

Introduzione alla programmazione

Vediamo la simulazione passo per passo, relativa al problema n.1, utilizzando una tabella con tante colonne quante sono le variabili manipolate dalle istruzioni dell'algoritmo, più una colonna che serve per indicare le azioni di cui si simula l'esecuzione

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.				

Introduzione alla programmazione

Le fasi di simulazione e codifica dell'algoritmo

Quando si è sicuri del funzionamento dell'algoritmo, si può passare alla fase della sua codifica in un linguaggio di programmazione che l'elaboratore è in grado di interpretare e comprendere. La **codifica** consiste, quindi, nella **traduzione dell'algoritmo in un insieme di istruzioni equivalenti** codificate in un determinato **linguaggio di programmazione**, ottenendo finalmente il programma, che a questo punto potrà essere “caricato” nella memoria del computer per essere successivamente eseguito.

Alcuni tra i principali linguaggi di programmazione sono **Pascal, Visual Basic, il linguaggio C, il linguaggio C#, Java, Python...**

Introduzione alla programmazione

Le fasi di simulazione e codifica dell'algoritmo

Alcuni tra i principali linguaggi di programmazione

Pascal

Visual Basic

il linguaggio C

il linguaggio C#

Java

Python



Introduzione alla programmazione

Le fasi di simulazione e codifica dell'algoritmo

Tornando all'ambito della programmazione, ogni linguaggio di programmazione offre un proprio **ambiente di editing** in cui scrivere il corrispondente codice per poi salvarlo in memoria. Molto spesso suggeriscono la **sintassi del linguaggio**, aiutando il programmatore a evitare errori di battitura, oppure indicando eventuali dimenticanze nella scrittura del codice (es: **PyCharm**).

Introduzione alla programmazione

Le fasi di simulazione e codifica dell'algoritmo

Una volta editato il codice, è possibile **lanciarlo in esecuzione**. Prima che però si passi all'esecuzione vera e propria delle istruzioni, il programma viene **tradotto in linguaggio macchina** da un altro programma chiamato **traduttore**, in modo che a questo punto il microprocessore dell'elaboratore su cui si sta lavorando possa **interpretare ed eseguire** le istruzioni del programma eseguibile, chiedendo in input valori opportuni e fornendo in output risultati richiesti.

Introduzione alla programmazione

Le fasi di simulazione e codifica dell'algoritmo

```
class Hello
{
    public static void main(String [] argv)
    {
        System.out.println("Hello world!");
    }
}
```

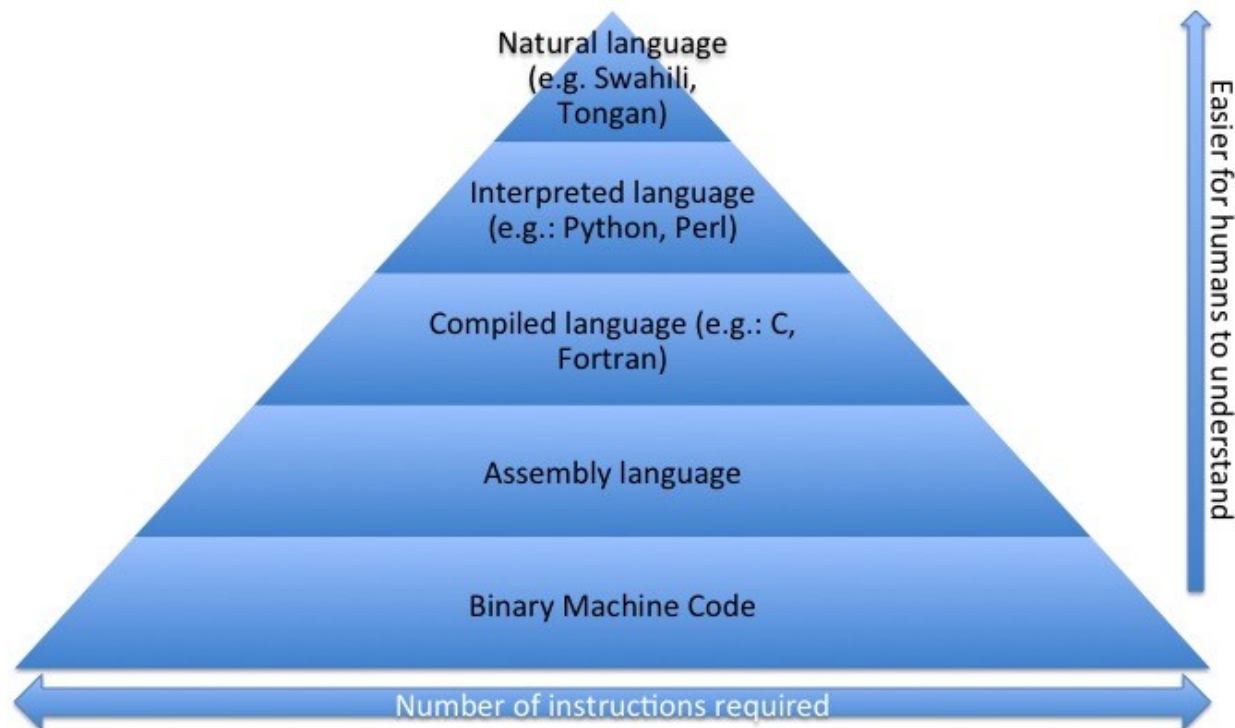


```
11001010 11111110 10111010 10111110 00000000 00000011 00000000 00101101
00000000 00100000 00001000 00000000 00010100 00000111 00000000 00010011
00000111 00000000 00011010 00000111 00000000 00011011 00000111 00000000
00011100 00001010 00000000 00000100 00000000 00001001 00001001 00000000
00000101 00000000 00001010 00001010 00000000 00000011 00000000 00001011
00001100 00000000 00001111 00000000 00001100 00001100 00000000 00011110
00000000 00010110 00001100 00000000 00011111 00000000 00001101 00000001
00000000 00000011 00101000 00101001 01010110 00000001 00000000 00010101
00101000 01001100 01101010 01100001 01110110 01100001 00101111 01101100
01100001 01101110 01100111 00101111 01010011 01110100 01110010 01101001
01101110 01100111 00111011 00101001 01010110 00000001 00000000 00010110
00101000 01011011 01001100 01101010 01100001 01110110 01100001 00101111
01101100 01100001 01101110 01100111 00101111 01010011 01110100 01110010
01101001 01101110 01100111 00111011 00101001 01010110 00000001 00000000
00000110 00111100 01101001 01101110 01101001 01110100 00111110 00000001
00000000 00000100 01000011 01101111 01100100 01100101 00000001 00000000
00001101 01000011 01101111 01101110 01110011 01110100 01100001 01101110
01110100 01010110 01100001 01101100 01110101 01100101 00000001 00000000
00001010 01000101 01111000 01100011 01100101 01110000 01110100 01101001
01101111 01101110 01110011 00000001 00000000 00000101 01001000 01100101
[...]
```

Introduzione alla programmazione

Gerarchia di un linguaggio

Language hierarchy



Introduzione alla programmazione

Apr 2019	Apr 2018	Change	Programming Language	Ratings	Change
1	1		Java	15.035%	-0.74%
2	2		C	14.076%	+0.49%
3	3		C++	8.838%	+1.62%
4	4		Python	8.166%	+2.36%
5	6	↑	Visual Basic .NET	5.795%	+0.85%
6	5	↓	C#	3.515%	-1.75%
7	8	↑	JavaScript	2.507%	-0.99%
8	9	↑	SQL	2.272%	-0.38%
9	7	↓	PHP	2.239%	-1.98%
10	14	↑↑	Assembly language	1.710%	+0.05%
11	18	↑↑	Objective-C	1.505%	+0.25%
12	17	↑↑	MATLAB	1.285%	-0.17%
13	10	↓	Ruby	1.277%	-0.74%
14	16	↑	Perl	1.269%	-0.26%
15	11	↓↓	Delphi/Object Pascal	1.264%	-0.70%
16	12	↓↓	R	1.181%	-0.63%
17	13	↓↓	Visual Basic	1.060%	-0.74%
18	19	↑	Go	1.009%	-0.17%
19	15	↓↓	Swift	0.978%	-0.56%
20	68	↑↑	Groovy	0.932%	+0.82%

*TIOBE index:
i linguaggi di
programmazione
più famosi*



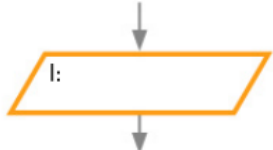
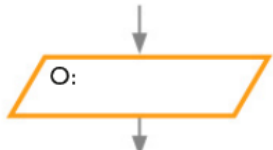
Introduzione alla programmazione

Gli schemi di flusso

Al fine di descrivere l'algoritmo in maniera **oggettiva e senza ambiguità**, si ricorre al formalismo degli **schemi di flusso**, detti anche **diagrammi a blocchi** o **flow chart**: attraverso gli schemi di flusso è possibile rappresentare graficamente le istruzioni che compongono un algoritmo mediante l'utilizzo di simboli grafici la cui forma varia a seconda del tipo di azione che si esprime. I simboli sono uniti da frecce che rappresentano il flusso delle azioni che costituiscono l'algoritmo. È così possibile ottenere una descrizione dell'algoritmo molto più efficace e chiara, semplice da capire e da modificare, standard e, soprattutto, priva di ambiguità interpretative.




Introduzione alla programmazione

Gli schemi di flusso: *tabella dei simboli (1)*

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.

Introduzione alla programmazione

*Gli schemi di flusso: **tabella dei simboli** (2)*

Tipo di istruzione	Simbolo	Significato
Salto	<p>È rappresentato da una freccia che si innesta in un'altra freccia nel punto dell'algoritmo cui si deve saltare.</p> 	Salto condizionato o incondizionato: va a eseguire l'istruzione indirizzata dal flusso delle frecce.
Inizio algoritmo		Blocco di Inizio algoritmo.
Fine algoritmo		Blocco di Fine algoritmo.

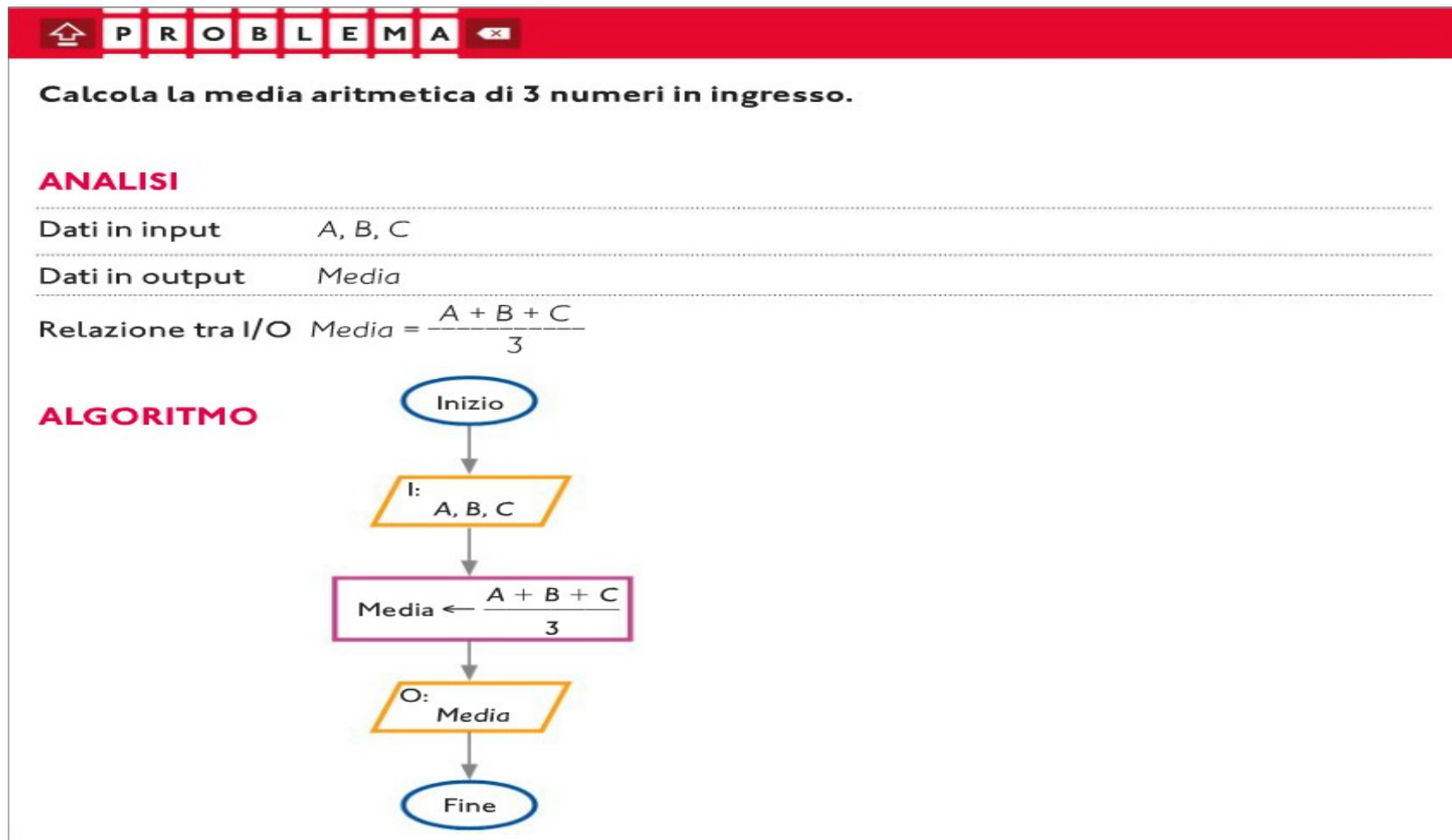
Introduzione alla programmazione

Lo schema di flusso: definizione

Uno **schema di flusso** (o diagramma a blocchi o flow chart) è una **rappresentazione grafica di un algoritmo** realizzata mediante **l'utilizzo di simboli**, la cui forma dipende dal tipo di azione che si vuole descrivere, **uniti da frecce** che rappresentano il flusso dell'esecuzione delle istruzioni che compongono l'algoritmo.

Introduzione alla programmazione

Schema di Flusso: esempio n.1



Introduzione alla programmazione

PROBLEMA

Fornita in ingresso al programma la lunghezza del lato di un quadrato, calcolarne il perimetro e l'area

ANALISI

Dati in input $LatoQuadrato$ (con $LatoQuadrato > 0$)

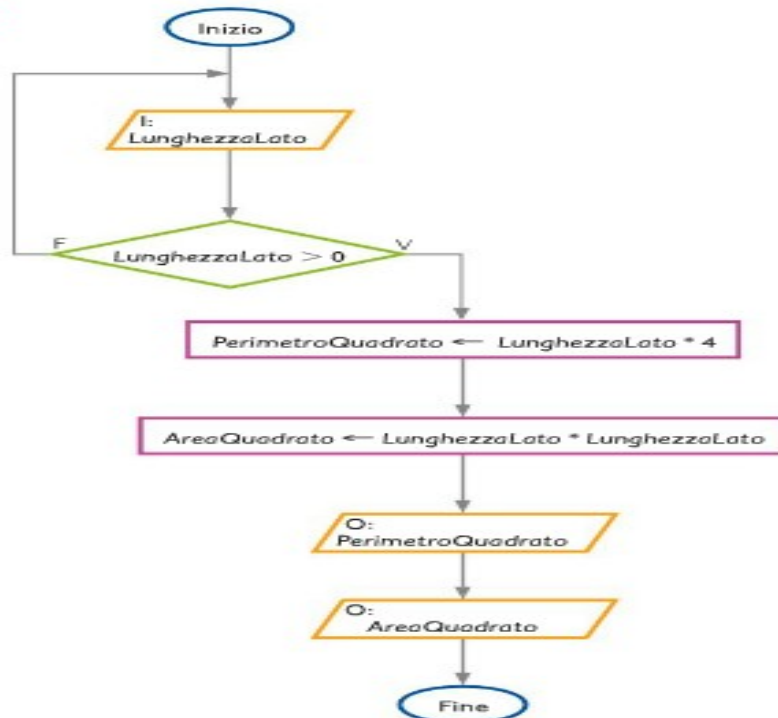
Dati in output $PerimetroQuadrato$

$AreaQuadrato$

Relazione tra I/O $PerimetroQuadrato = LatoQuadrato * 4$

$AreaQuadrato = LatoQuadrato * LatoQuadrato$

ALGORITMO



*Schema di Flusso
esempio n.2*

Introduzione alla programmazione

PROBLEMA

Calcolare e applicare lo sconto del 20% a un prezzo fornito in ingresso solo se questo supera 100 €.

ANALISI

Dati in input *Prezzo* (con $\text{Prezzo} > 0$)

Dati in output *PrezzoFinale*

Relazione tra I/O – Se $\text{Prezzo} > 100$ allora calcolo:

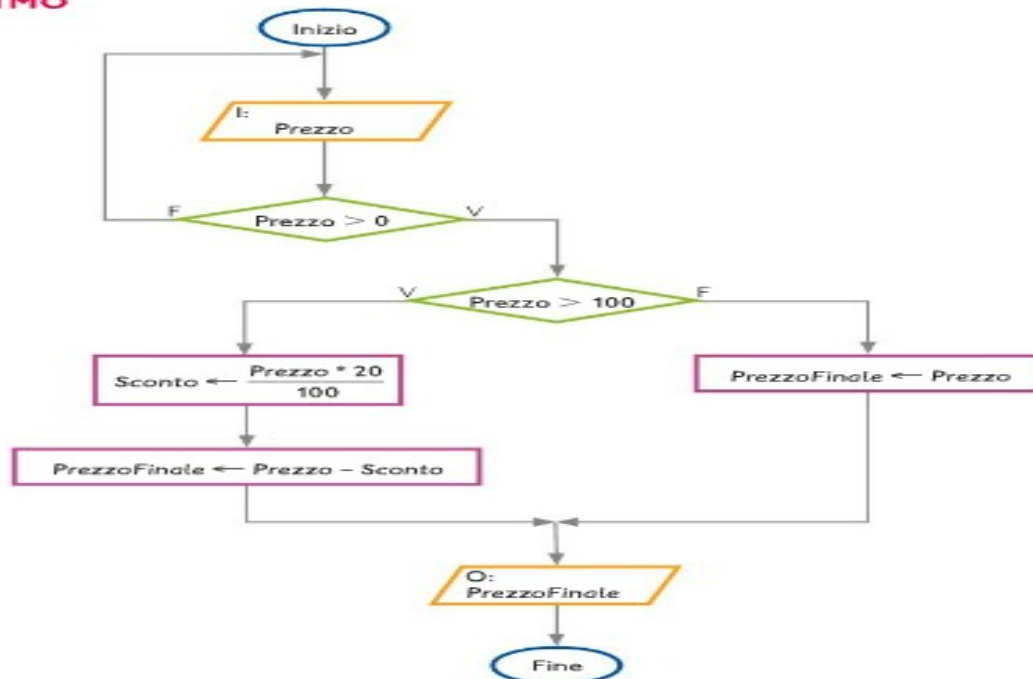
$$\text{Sconto} = \frac{\text{Prezzo} * 20}{100}$$

quindi calcolo:

$$\text{PrezzoFinale} = \text{Prezzo} - \text{Sconto}.$$

– Se $\text{Prezzo} \leq 100$ allora $\text{PrezzoFinale} = \text{Prezzo}$.

ALGORITMO



*Schema di Flusso
esempio n.3*

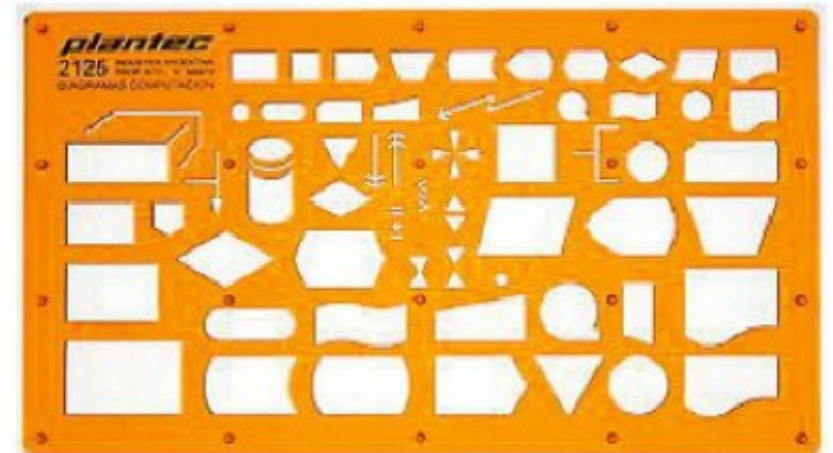
Introduzione alla programmazione

Schema di Flusso: *strumenti*

Gli strumenti per il disegno dei flow chart

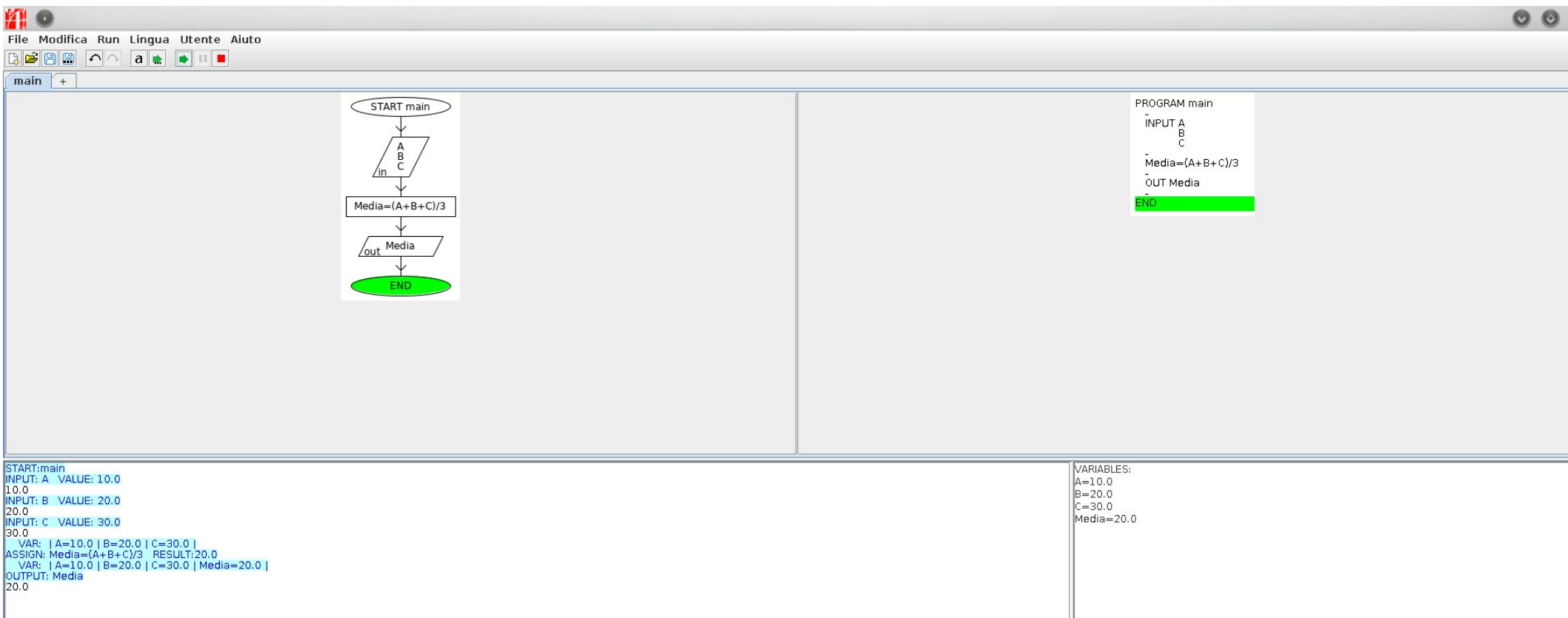
Esistono vari tools gratuiti o a pagamento per aiutarsi nel disegno degli schemi di flusso. Tra questi segnaliamo *Draw.io*, un sito web gratuito (www.draw.io) molto intuitivo e pratico e che non richiede alcuna registrazione per il suo utilizzo, e il software *AlgoBuild* scaricabile e installabile gratuitamente (www.algobuild.com) che permette anche di ottenere, mentre si disegna lo schema di flusso, la sua pseudocodifica e un semplice sistema di simulazione dell'esecuzione.

Se invece preferite adoperare carta e penna, vi segnaliamo un gadget che potete trovare in cartoleria: si tratta di un normografo realizzato apposta per il disegno degli schemi di flusso.



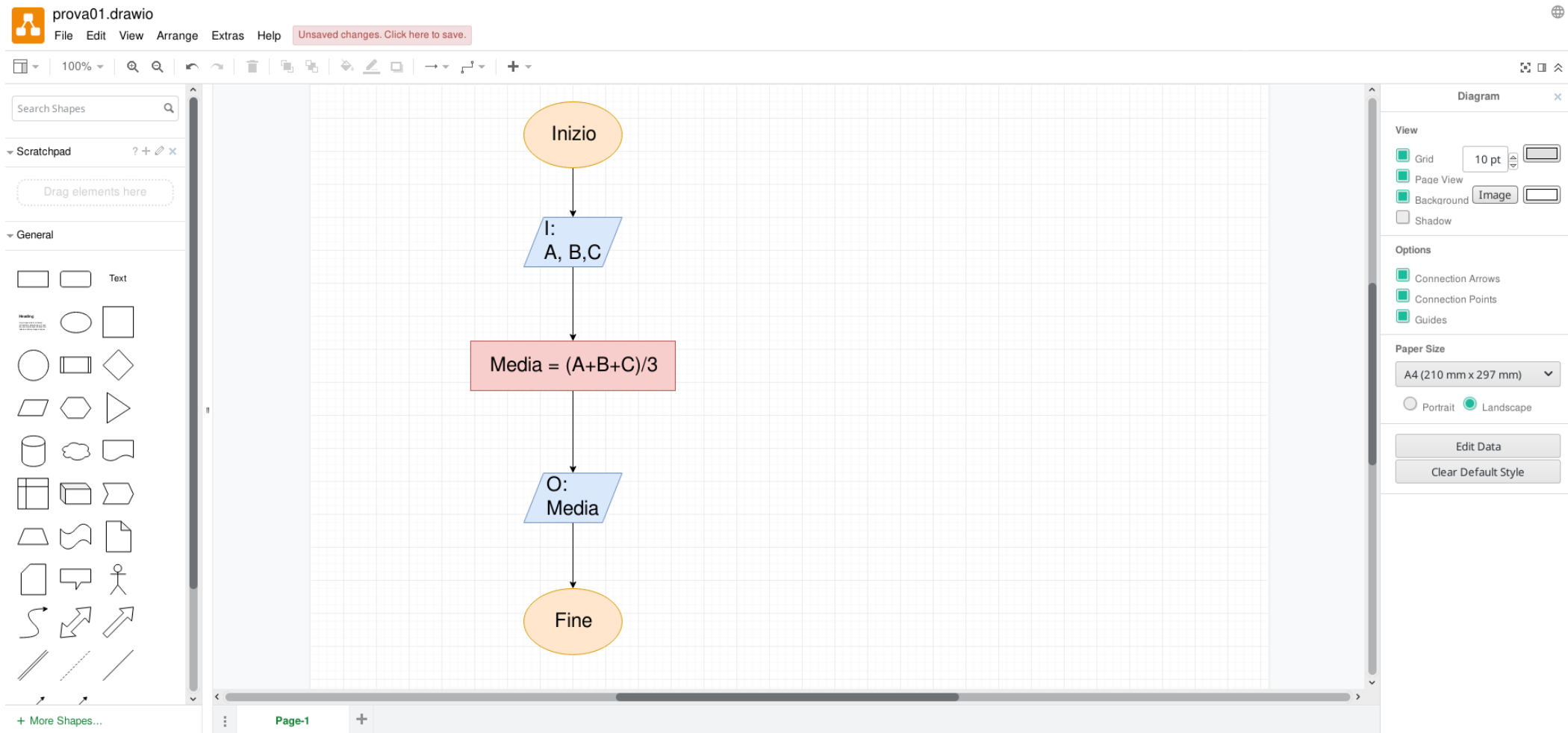
Introduzione alla programmazione

Schema di Flusso: esempio n.1 (con AlgoBuild)



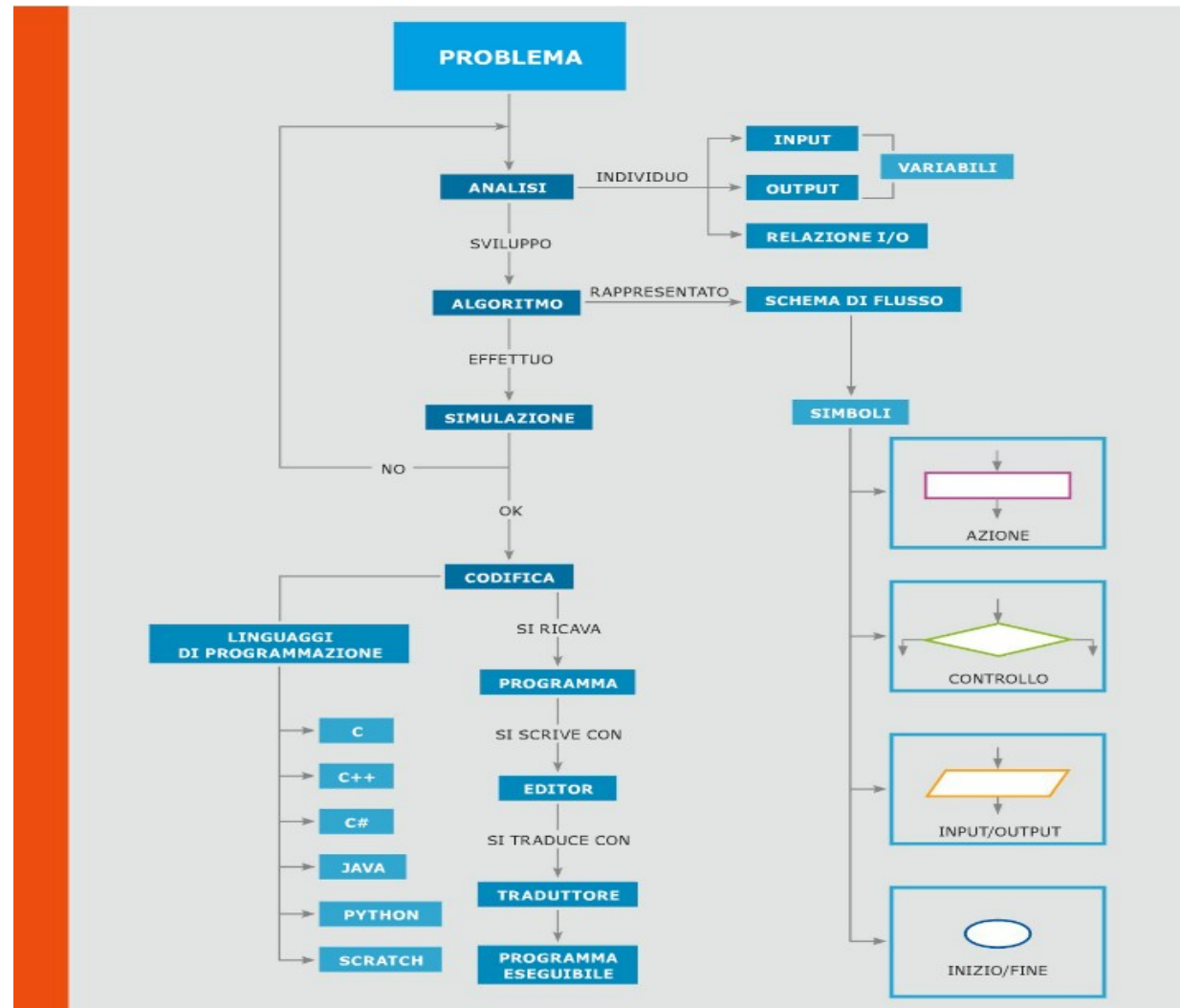
Introduzione alla programmazione

Schema di Flusso: esempio n.1 (con Draw.io)



Introduzione alla programmazione

*Per
riassumere...*



Introduzione alla programmazione

PROBLEMI

14 Calcola la somma di tre valori forniti in input.

15 Calcola l'area di un triangolo di cui sono forniti in input la base e l'altezza.

16 Ricevute in ingresso le lunghezze dei tre lati di un triangolo, determina se si tratta di un triangolo equilatero, isoscele o scaleno.

17 Dato in ingresso il raggio, calcola la circonferenza e l'area del cerchio.

18 Calcola il volume di un parallelepipedo di cui sono forniti in input altezza, larghezza e profondità.

19 Ricevuti in ingresso due valori, fanne la sottrazione se il primo è maggiore del secondo o fanne la somma se il primo è minore del secondo. Se sono uguali, segnalalo a video.

20 Determina se un voto ricevuto in ingresso è sufficiente o insufficiente.



Diagramma di flusso e codifica in Scratch - C - Pascal - Python - Java

21 Calcola la media di cinque voti ricevuti in ingresso.

22 Ricevuti in ingresso cinque voti, conta quanti sono sufficienti, quanti insufficienti e visualizza se sono di più i voti positivi o quelli negativi.

23 Determina il voto più basso tra cinque voti ricevuti in input.



Diagramma di flusso e codifica in Scratch - C - Pascal - Python - Java

24 Determina il voto più alto tra sei voti ricevuti in input.

25 Visualizza in ordine crescente e decrescente tre valori ricevuti in input.

26 Visualizza in ordine crescente quattro valori ricevuti in input.

27 Ricevuti in ingresso i nomi di due squadre di calcio e il risultato della partita, visualizza la squadra vincente o, in caso di pareggio, entrambi i nomi delle squadre.



Diagramma di flusso e codifica in Scratch - C - Pascal - Python - Java

28 Ricevuti in ingresso il nome di una squadra di calcio, il numero di partite vinte e il numero di partite pareggiate, calcola i punti in classifica.

29 Realizza una semplice calcolatrice che, ricevuti in input due operandi e un operatore aritmetico, fornisca in uscita il risultato dell'operazione richiesta.

30 Dati in ingresso il costo di quattro libri, calcola il costo medio per libro.

31 Dati in ingresso il costo di cinque libri, conta quanti di questi hanno un prezzo superiore a 15 €.

32 Un libro deve essere restituito in biblioteca dopo 15 giorni di prestito altrimenti si è multati di 0,80 € al giorno di ritardo. Ricevuto in ingresso il numero di giorni di un prestito, visualizza se il socio deve essere multato per il ritardo e a quanto ammonta la multa da pagare.



Diagramma di flusso e codifica in Scratch - C - Pascal - Python - Java

33 Calcola e applica lo sconto al prezzo fornito in ingresso di un hoverboard a seconda delle seguenti fasce di prezzo:

- se prezzo < 100 € → sconto 5%
- se prezzo ≥ 100 € e < 200 € → sconto 10%
- se prezzo ≥ 200 € → sconto 15%

34 Ricevuti in ingresso il numero di persone adulte e il numero di bambini che vanno al cinema, calcola il costo totale del biglietto sapendo che gli adulti pagano 7 € mentre i bambini pagano 4 €.

35 Ricevuti in ingresso il costo del biglietto per visitare un museo e il numero di alunni di una classe in gita scolastica, calcola la spesa totale.

Ora a voi...

Introduzione alla programmazione

Bibliografia

- Barbero, Vaschetto, “Dal bit al web”, Pearson-Linx
- AA.VV., “Informatica e algoritmo”, Wikipedia.it

Software

- AlgoBuild, www.algobuild.com
- Draw.io, www.draw.io

Immagini (free license)

- Fonte: www.duckduckgo.com