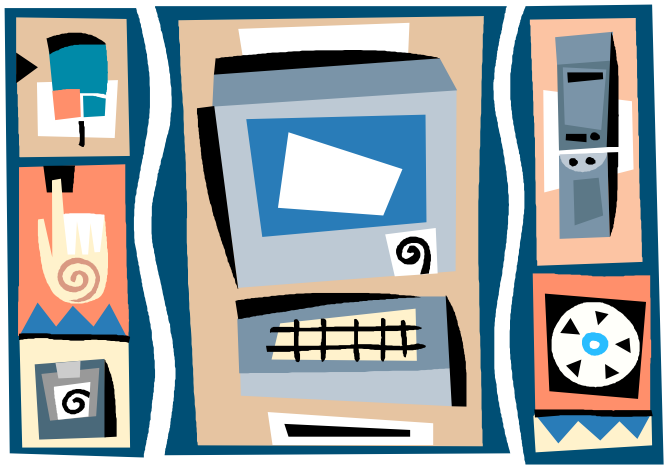
APPUNTI DI TECNOLOGIE INFORMATICHE



di Giusto Matteo Classe 1ª G

Docente prof. Beniamino Bortelli

ITIS Vito Volterra

San Donà di Piave

Anno Scolastico 2010 – 2011

L'INFORMATICA

L'informatica è: *la disciplina che si occupa della elaborazione automatizzata delle informazioni.*

L'***informatica*** si occupa:

- del trattamento dei dati (di persone, aziende, società, ecc.) comprendente la raccolta, l'archiviazione in data base elettronici, la consultazione, la modifica, la trasmissione a distanza, mediante linee di comunicazione veloci.
- della implementazione di programmi applicativi nei campi della matematica: calcolo, dell'ingegneria: robotica, automazione, simulazione, dell'economia: finanza, e-commerce, dell'arte: grafica e animazione … .

Il termine **disciplina** rimanda ad un **lavoro ordinato e coordinato**.

Elaborazione è una qualunque modifica tipo:

apri – chiudi

scrivi – leggi

aggiungi – togli

moltiplica – dividi

Informazione è, ad esempio, il contenuto di un messaggio.

Il termine **automatizzata** si riferisce, infine, ad una azione meccanica e ripetitiva.

Tutto ciò mostra che l'informatica modifica le informazioni secondo uno schema meccanico preordinato.

L'INFORMATICA

continuazione

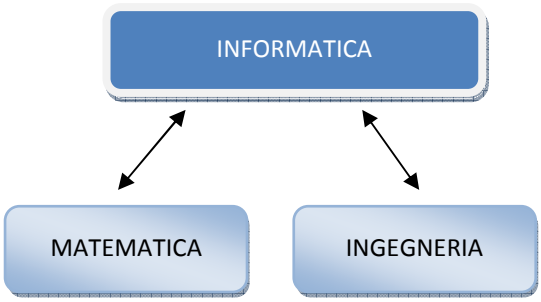
✓Il termine ***informatica*** deriva dall'unione di due parole:

informazione + ***automazione***

"informazione": più inerente alla "matematica"

"automazione": più inerente all' "ingegneria"

L'**informatica** poggia su due discipline: la matematica e l'ingegneria e sulla sinergia tra di esse:



La matematica contribuisce con la *ricerca pura* ed opera nei seguenti campi:

- Logica
- Calcolo numerico
- Linguaggi ed algoritmi
- Teoria dell'informazione

L'ingegneria si occupa soprattutto della ricerca applicata nei seguenti campi:

- Tecnologia
- Reti e comunicazione
- Controllo e automazione
- Robotica

Da un lato la matematica elabora ed affina sempre nuovi linguaggi ed algoritmi e dall'altro l'ingegneria produce sempre nuove soluzioni hardwere, favorendo la penetrazione dell'informatica in nuovi campi applicativi.

La sinergia tra queste due discipline ha determinato il rapido sviluppo dell'informatica, che raddoppia le proprie prestazioni tipicamente ogni 18 mesi.

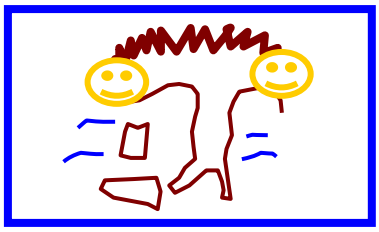
INFORMAZIONE E DATI

L'informazione è: *tutto ciò che ha significato per una persona.*

Avere le **informazioni** al momento giusto consente di agire in modo più **determinato** e meno **casuale**, mentre non averle, ovvero ***ignorare*** le informazioni, costringe ad agire in modo più **casuale** e meno **determinato**.

- L'**informazione** può essere di tipo verbale e/o non verbale

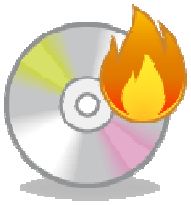
VERBALE



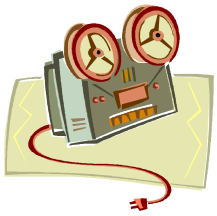
NON VERBALE – VISIVA

- L'**informazione** necessita di un supporto fisico, che può essere:

CARTACEO



OTTICO



MAGNETICO

AUDIO

ELETTRICO

- L'**informazione** necessita di una persona che le attribuisca un **significato**. Se verbale, deve essere espressa con una frase completa, con soggetto, verbo e complemento, che abbia senso e significato:

Gli elementi, i frammenti di una informazione sono i **dati**.

Dato è: *ciò che un computer: riceve in input, elabora, restituisce in output..*

I computer non danno significato ai dati, ma si limitano alla manipolazione degli stessi seguendo pedantemente le istruzioni che ricevono con i programmi.

L'importanza dell'informazione

Oggi l'**informazione** ha a disposizione per la comunicazione, un insieme di mezzi di grandi potenzialità, denominati anche **media**:

1:	I giornali
2:	La Radio
3:	Il telefono fisso e mobile
4:	La TV
5:	La rete internet

Tali potenzialità sono oggi così elevate che spesso si parla di un vero e proprio **POTERE** dell'informazione o anche del **POTERE mediatico**.

Esso si affianca ai tradizionali "***poteri***" che sono:

•**Potere esecutivo**	Di chi ha l'autorità per l'uso della forza.
•**Potere legislativo**	Di chi stabilisce le regole di comportamento.
•**Potere giudiziario**	Di chi deve giudicare sul rispetto o meno delle regole.
•**Potere economico**	Di chi possiede ricchezze in beni e denaro.
•**Potere dell'informazione**	Di chi possiede i media.

I primi tre poteri, sono i poteri sui quali è organizzato politicamente uno stato e sono conosciuti dall'antichità. Sulle implicazioni di questi poteri sono stati scritti numerosi trattati tra i quali sono molto conosciuti: **La republica** di **Platone (400 a.C.)** **e** **Il contratto sociale** di **Rosseau (1700 d.C.)**.

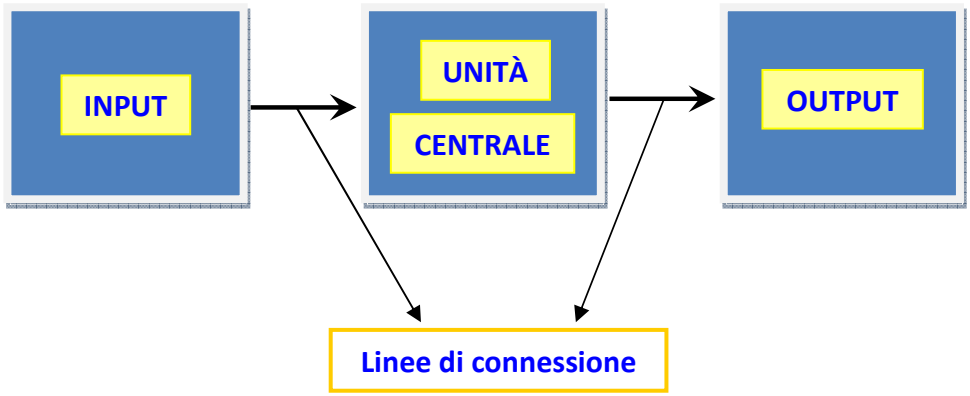
Il quarto potere è connesso con lo sviluppo economico di una nazione ed è stato studiato approfonditamente solo con lo sviluppo delle nazioni moderne, quindi in tempi abbastanza recenti. Il primo trattato economico importante è: **La richezza delle nazioni** di **Adam Smith (1700 d.C.)**.

Invece, benché l'informazione abbia sempre rivestito grande importanza, l'argomento è stato trattato teoricamente in modo organico sono in tempi recentissimi, con lo sviluppo dell'informatica. Il primo trattato completo sull'informazione è infatti: **La teoria dell'informazione** di **Shannon (1900 d.C.)**.

La composizione interna del computer

Il computer è un'apparecchiatura costituita a un insieme di parti di diversa natura: meccanica, elettrica, ottica, che si distinguono in

- ✓ **PERIFERICHE DI INPUT**: consentono l'immissione dei dati da elaborare
- ✓ **UNITA' CENTRALE**: ha il compito di elaborare i dati
- ✓ **PERIFERICHE DI OUTPUT**: consentono l'emissione dei dati elaborati
- ✓ **PERIFERICHE DI IN/OUT**: consentono sia l'immissione, che l'emissione dei dati



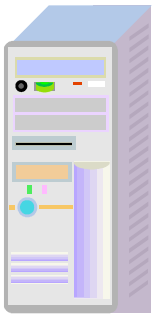
Sono **periferiche di input**: la **tastiera**, il **mouse**, la **tavoletta grafica**, la **videocamera** (**webcam**), la **fotocamera digitale**, ecc.

Sono **periferiche di output**: il **monitor**, il **diffusore audio**, la **stampante**, il **videoproiettore**, ecc.

Sono **periferiche di in/out**: i dispositivi **touch screen**.

Nel contenitore (case) di un computer troviamo:

- l'**alimentatore**
- La **scheda madre** (**mother board**)
- Le unità di **memoria di massa**: **magnetica** (**hard disk**) e **ottica** (**CD – DVD**)
- La **scheda di controllo** delle unità di memoria di massa (**IDE**)
- Le **schede di controllo** delle unità periferiche: **scheda video**, **scheda audio**, **scheda di rete**.

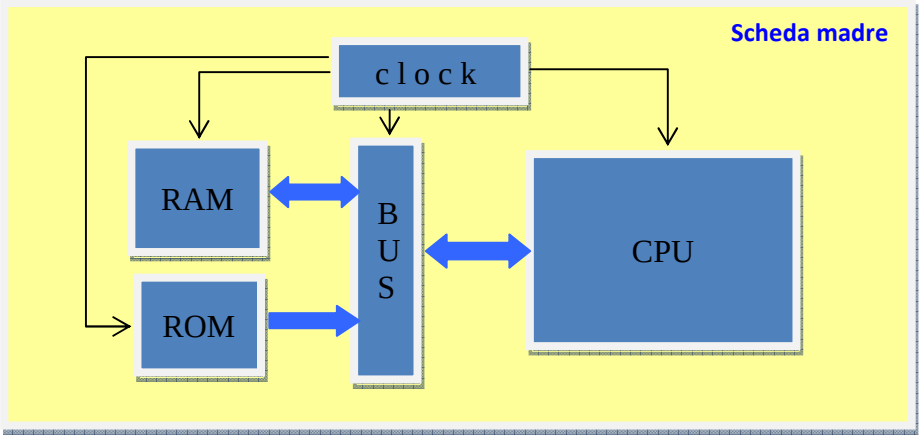


Case

la mother board

La **mother board** è il cuore del sistema di elaborazione, dove risiede **l'unità centrale di elaborazione** (**CPU**), o **microprocessore**.

Oltre alla **CPU** la mother board contiene: la **memoria principale** (**RAM**), la **memoria a sola lettura** (**ROM**), le **linee di interconnessione** (**BUS**) e **l'orologio elettronico** (**clock**) che scandisce i tempi delle elaborazioni.



La **memoria RAM** è una memoria a **scrittura** e **lettura** di tipo **elettronico**, organizzata con dei moduli chiamati anche **SIMM**. Essa è molto veloce, ma volatile, e quando il computer viene spento si cancella.

La **memoria ROM** è una memoria **elettronica non volatile** a sola **lettura**, e contiene dei dati molto importanti per l'avvio del computer.

Quando un programma va in esecuzione la CPU prende i dati dalla memoria di massa e li trasferisce nella RAM, molto più veloce, e li elabora seguendo le istruzioni degli utenti. Quando l'applicazione finisce ed il programma viene chiuso, i dati elaborati devono essere trasferiti nuovamente nella memoria di massa.

Il **Clock** è l'orologio interno del computer. Esso fornisce un segnale ciclico, cioè che ripete lo stesso ciclo di base ad intervalli regolari di tempo. Determina la **velocità di elaborazione** di un computer.

la velocità di elaborazione

La velocità di elaborazione di un computer indica quante operazioni è in grado di eseguire la CPU nell'unità di tempo.

Attualmente i computer hanno una velocità di elaborazione molto alta perché sono dotati di orologi sempre più potenti.

Si chiama Frequenza il **n° di cicli** che compie il segnale di clock **in un secondo**.

L'Hertz è l'unità di base

1 Hz corrisponde a **1 CICLO AL SECONDO**

1 MHz significa **1 milione di Hz** (10^6 **Hz**) cioè **un milione di cicli al secondo**.

1 GHz significa **1 miliardo di Hz** (10^9 **Hz**)

La frequenza di clock di un computer è oggi di **alcuni GHz** , per cui essi sono in grado di compiere almeno **un miliardo di operazioni al secondo**.

I CODICI

Un codice è un insieme di convenzioni che consentono di rappresentare le informazioni.

Esso è formato da:

1. Un insieme di segni convenzionali (simboli) scelti arbitrariamente dagli autori del codice.
2. Un insieme di regole con cui i simboli devono essere composti.

Esempi

Simboli	Cos'è	Come si legge	Informazione
	Lettera dell' alfabeto latino	effe	suono consonantico "F"
	Lettera del Codice morse	"punto punto punto"	lettera "S"
	Icona di Windows	…	"salva"

- Il significato dei simboli è di solito immediatamente deducibile della loro forma.
- Un codice è in chiaro (pubblico) quando le informazioni per il suo utilizzo sono accessibili a tutti.
- Un codice è cifrato quando le informazioni per il suo utilizzo restano riservate a una cerchia di persone.

I codici vengono prodotti da:

- Comitati internazionali degli stati.
- Laboratori e enti di ricerca.
- Industrie leader del mercato.
- Chiunque.

A seconda del tipo di informazione cambia il codice con cui viene rappresentata.

Tipo di informazione	codice
Numeri	Sistema binario
Lettere	Codice Ascii
Immagini fisse (foto)	jpeg
Immagini in movimento (filmati)	mpeg
Suoni	mp3

IL SISTEMA BINARIO

Il sistema binario è il sistema di numerazione a base 2, affine al sistema decimale, utilizzato negli elaboratori.

Si chiama BASE di un sistema di numerazione il numero di simboli utilizzati per esprimere le diverse cifre di un numero.

Sistema decimale utilizza 10 simboli : 0,1…..9 chiamati "decimal digit"

Il sistema binario utilizza 2 simboli : 0,1 chiamati "binary digit" o "bit"

s. decimale	s.decimale codificato	s. binario	s. binario codificato
0	00	0	0000
1	01	1	0001
2	02	10	0010
3	03	11	0011
4	04	100	0100
5	05	101	0101
6	06	110	0110
7	07	111	0111
8	08	1000	1000
9	09	1001	1001
10	10	1010	1010
11	11	1011	1011
12	12	1100	1100
13	13	1101	1101
14	14	1110	1110
15	15	1111	1111

Tipi base sono:

BIT 1 CIFRA BINARIA *acronimo di b(inary dig)it*

NIBBLE 4 CIFRE BINARIE

BYTE 8 CIFRE BINARIE

WORD 16 CIFRE BINARIE

Cifre binarie ancora più grandi sono espresse in multipli di byte:

➔1 kbyte (1 KB) = 2^{10} byte = 1024 byte ≈ 1000 byte

➔1 Mbyte (1 MB) = 2^{10} kbyte = 1024 byte ≈ 1000 kbyte

➔1 Gbyte (1 GB) = 2^{10} Mbyte = 1024 Mbyte ≈ 1000 Mbyte

Proprietà del sistema binario

•***Aggiunta di uno 0 a destra***

Esempi: *dato un numero, lo convertiamo in binario, aggiungiamo uno zero e osserviamo che numero otteniamo*

tre	**11**	**110**	**sei**
cinque	**101**	**1010**	**dieci**
sei	**110**	**1100**	**dodici**
sette	**111**	**1110**	**quattordici**
nove	**1001**	**10010**	**diciotto**

REGOLA:aggiundendo uno 0 a destra il numero binario raddoppia.
(2 corrisponde alla base, quindi è moltiplicato per la base)

•***Valore di un n° binario con una sola cifra a "1".***

Esempio **prendiamo un numero binario di quattro cifre (NIBBLE), con una sola cifra a "1". Abbiamo quattro possibilità:**

1 0 0 0	0 1 0 0	0 0 1 0	0 0 0 1
1	2	3	4

1: **1000 = 8(= 1*2*2*2)***ci sono tre zeri e quindi si moltiplica tre volte per 2*

2: **0100 = 4(= 1*2*2)***ci sono due zeri e quindi si moltiplica due volte per 2*

3: **0010 = 2 (= 1*2)***c'è uno zero e quindi si moltiplica una volta per 2*

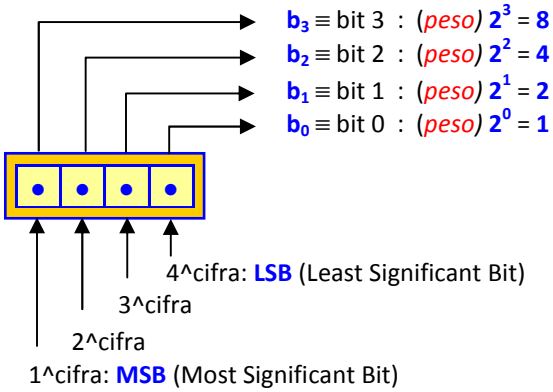
4: **0001 = 1***non ci sono zeri*

REGOLA:le cifre binarie hanno valore diverso a seconda della posizione in cui sono collocate. Più è a sinistra più il valore cresce.

IL SISTEMA BINARIO E UN SISTEMA POSIZIONALE.

•*Numerazione delle cifre binarie*

Le cifre binarie vengono numerate con un indice crescente verso sinistra



•n® *binario con più cifre a "1"*

Esempio **prendiamo un numero binario di quattro cifre (NIBBLE), con due cifre a "1". Scomponiamo il numero binario in due, ciascuno con una sola cifra a "1":**

1	0	1	0	=	1	0	0	0	+	0	0	1	0

$= 1*2^3 + 1*2^1$

$= 8 + 2$

$= 10$

Deduciamo la seguente

REGOLA: il valore di un numero binario si ottiene sommando i pesi (fattori moltiplicativi) delle sue cifre binarie a "1".

•*Espressione generale di un numero binario*

Esempio **prendiamo un numero binario di quattro cifre (NIBBLE), con le cifre indicate, le quali possono essere a "0" oppure a "1". Scomponiamo il numero come prima:**

b_3	b_2	b_1	b_0	=	b_3	0	0	0	+	0	b_2	0	0	+	0	0	b_1	0	+	0	0	0	b_0	0

$n = b_3 * 2^3 + b_2 * 2^2 + b_1 * 2^1 + b_0 * 2^0$

il bit è "0", anche il termine relativo è zero, mentre se il bit è "1", allora si ritorna nel caso precedente. Quindi questa espressione si applica a tutte le situazioni.

•*Conversione da binario a decimale*

Dato un numero qualsiasi espresso in binario, dalla sua espressione non riusciamo ad avere idea del suo effettivo valore. Per risalire al suo valore lo dobbiamo convertire in decimale. A tale scopo dobbiamo utilizzare l'espressione precedentemente ricavata.

Esempio ***convertire in decimale il numero binario***: 1001101_2

Segniamo accanto a ciascuna cifra il suo indice ed il suo peso e applichiamo la formula di conversione generale:

	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	b_6	b_5	b_4	b_3	b_2	b_1	b_0
n =	1	0	0	1	1	0	1

$= b_6 2^6 + b_5 2^5 + b_4 2^4 + b_3 2^3 + b_2 2^2 + b_1 2^1 + b_0 2^0$

$= 1 \cdot 2^6 + 0 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$

$= 2^6 + 2^3 + 2^2 + 2^0$

$= 64 + 8 + 4 + 1$

$= 77_{10}$

•*Eliminazione di una cifra a destra*

Esempi: *dato un numero, lo esprimiamo in binario e togliamo la cifra di destra*

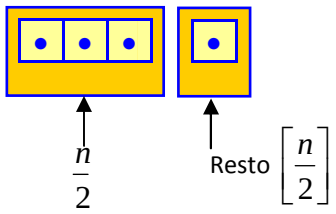
dieci	**1010**	**101**	**cinque**
dodici	**1100**	**110**	**sei**
diciotto	**10010**	**1001**	**nove**
undici	**1011**	**101**	**cinque**
tredici	**1101**	**110**	**sei**
diciannove	**10011**	**1001**	**nove**

REGOLA:togliendo una cifra a destra il numero binario viene diviso per due. La divisione è intera, quindi vengono perse le cifre dopo la virgola.

Osserviamo anche un'altra cosa: se il numero è pari termina con "0", ed anche il resto della divisione è "0" . In questo caso la divisione è esatta. Se, invece, il numero è dispari termina con "1", ed anche il resto della divisione è "1" . In questo caso la divisione non è esatta. In entrambi i casi, comunque, l'ultima cifra coincide con il resto della divisione del numero per 2.

•*Conversione da decimale a binario*

Per convertire un numero da decimale a binario basta dividere ripetutamente il numero per 2 e ogni volta vedere se è pari o dispari. La divisione per 2 ci fa scalare di una cifra e il fatto che il numero ottenuto sia pari o dispari ci dice se la cifra è zero od uno.



Il procedimento da seguire è il seguente:

INIZIO

- **1° passo: si prende il numero binario dato e lo si divide per 2. Si registrano il quoziente ed il resto della divisione.**
- **2° passo: si prende il quoziente precedente e lo si divide nuovamente per 2, ottenendo un nuovo quoziente ed un nuovo resto.**
- **3° passo: Si ripete il procedimento, finché il quoziente non è zero.**
- **Ultimo passo: La sequenza dei resti, letti in ordine inverso, fornisce il risultato della conversione.**

FINE

ESEMPIO convertire in binario il numero 45_{10}

INIZIO

1° Passo:	**45**	**:**	**2**	**=**	**22**	***Con resto***	**1**
2° passo	**22**	**:**	**2**	**=**	**11**	***Con resto***	**0**
3° passo	**11**	**:**	**2**	**=**	**5**	***Con resto***	**1**
4° passo	**5**	**:**	**2**	**=**	**2**	***Con resto***	**1**
5° passo	**2**	**:**	**2**	**=**	**1**	***Con resto***	**0**
6° passo	**1**	**:**	**2**	**=**	**0**	***Con resto***	**1**

7° passo ***Il risultato della conversione è:*** $45_{10} = 101101_2$

FINE

COMPLEMENTO A 1 e COMPLEMENTO A 2

In binario il complemento ad 1 ed il complemento a 2 sono come in decimale rispettivamente il complemento a 9 (quel numero che sommato al numero dato dà tutti 9) ed il complemento a 10 (quel numero che sommato al numero dato dà 10, o 100, o 1000, ecc.).

✓Il complemento a 1 di un numero binario è un numero binario in cui ciascuna cifra è il complemento della cifra corrispondente.

Dove:
il complemento di (0) = 1

il complemento di (1) = 0

ESEMPIO: *determinare il complemento a 1 di:* 1 0 0 1 1 0

Svolgimento:

1 0 0 1 1 0
↓ ↓ ↓ ↓ ↓ ↓
0 1 1 0 0 1

Il complemento a 1 di *:* 1 0 0 1 1 0 è 0 1 1 0 0 1

✓Il complemento a 2 di un numero binario si esegue eseguendo prima il complemento a 1 e poi sommando 1.

ESEMPIO: *determinare il complemento a 2 di:* 1 1 0 1 1 1

1 1 0 1 1 1
↓ ↓ ↓ ↓ ↓ ↓
0 0 1 0 0 0

+1=
↓ ↓ ↓ ↓ ↓ ↓
0 0 1 0 0 1

Il complemento a 2 di *:* 1 1 0 1 1 1 è 0 0 1 0 0 1

✓PROPRIETA' DEL COMPLEMENTO A 1:

sommando un numero con il suo complemento a 1 si ottengono tutti 1:

ESEMPIO:

N = 0110 1001

\+

$\overline{N}$ = 1001 0110

———————— =

$N + \overline{N}$ = 1111 1111

✓PROPRIETA' DEL COMPLEMENTO a 2:

Sommando un numero con il suo complemento a 2 si ottiene 0 a meno dell'overflow

N = 1 0 1 0 0 0 1 1

$\overline{N}$ = 0 1 0 1 1 1 0 0

\+

1

———————— =

0 1 0 1 1 1 0 1

$N + \overline{N}$ = $^{1}1^{1}0^{1}1\ ^{1}0\ ^{1}0^{1}0^{1}1^{1}1$

\+

= 0 1 0 1 1 1 0 1

———————— =

1 0 0 0 0 0 0 0 0

Over flow

Rappresentazione dei dati

All'interno di un elaboratore tutti i dati sono rappresentati mediante sequenze di "0" e di "1".

Il formato di rappresentazione, però, cambia a seconda che il dato sia un carattere alfanumerico, oppure un numero, o un'immagine.

I caratteri alfanumerici

I caratteri alfanumerici sono i caratteri che vengono digitali con la tastiera: lettere maiuscole e minuscole, caratteri che esprimono le cifre da 0 a 9, caratteri di interpunzione, operazioni matematiche, ed altri.

I caratteri alfanumerici sono rappresentati con il codice ASCII (American Standard Code for International Interchange). Con questo codice ciascun carattere è associato ad una sequenza di 8 bit corrispondente ad 1 byte. Il codice ASCII consente di codificare 256 caratteri. Essi sono numerati da 1 a 255 con il sistema binario.

I numeri

Nelle applicazioni per il calcolo numerico sono utilizzati:

- Numeri interi relativi (con segno), spesso denominati ***integer***: il valore del numero è sempre intero ed è preceduto dal segno (+) o (-). Se il segno non è presente si sott'intende il segno (+).

  Esempi: +47, -208, +1954

- Numeri decimali (con la virgola), spesso denominati ***real***: il valore del numero ed è preceduto dal segno (+) o (-) e presenta un certo numero di cifre dopo la virgola. Può presentarsi in diversi modi tra i quali i più diffusi sono: la rappresentazione in virgola fissa, se la virgola separa effettivamente la parte intera dalla parte decimale, ed in virgola mobile, se è presente un fattore moltiplicativo a causa del quale la virgola non separa più la parte fissa da quella decimale. Anche il fattore moltiplicativo può avere segno positivo o negativo.

  Esempi: (virgola fissa)+47, 802; -208,32; +1954,057

  Esempi: (virgola mobile – notazione esponenziale)$+5{,}2942 \times 10^{3}$; $-2{,}38052 \times 10^{-2}$; $+1{,}041 \times 10^{2}$

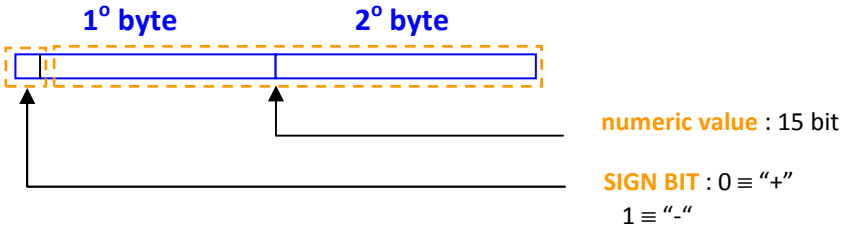
Rappresentazione dei numeri interi

Per rappresentare i numeri interi con segno è necessario rappresentare sia il valore del numero, che il suo segno.

Si utilizzano due byte (16 bit) di cui:

- Il primo bit è usato come bit di segno: **0** ≡ "**+**", **1** ≡ "**–**"
- I rimanenti 15 bit sono usati per rappresentare il valore

Formato:



Se il numero è positivo, il suo valore è rappresentato con il sistema binario puro.

Se invece il numero è negativo il suo valore è rappresentato in binario con la tecnica del complemento a 2.

Si utilizza la tecnica del complemento a 2 per soddisfare la proprietà fondamentale dei numeri relativi: *la somma di un numero relativo con il suo opposto è sempre zero*.

Sono rappresentati tutti i numeri da (-32768) a (+32767)

Rappresentazione dei numeri decimali

Per rappresentare i numeri decimali, cioè con la virgola, si procede con le seguenti fasi:

1° FASE: il numero decimale viene convertito in virgola mobile secondo la notazione esponenziale normalizzata.

Esempio: 509,347

Il numero è in virgola fissa (non c'è il fattore moltiplicativo), quindi si trasforma in virgola mobile:

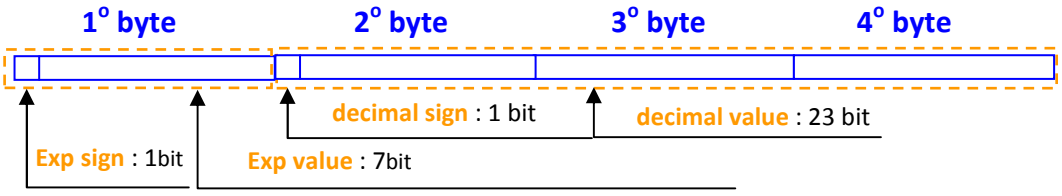
$509.347 \rightarrow +5{,}09347 \times 10^{2}$

Ora il numero è in virgola mobile in notazione esponenziale standard. Da essa si passa alla notazione esponenziale normalizzata:

$+5{,}09347 \times 10^{2} \rightarrow +0{,}509347 \times 10^{+3}$

Nella rappresentazione esponenziale normalizzata la parte intera è sempre nulla, mentre la parte dopo la virgola comincia sempre con un numero diverso da zero.

2° FASE: il numero decimale viene rappresentato (*di solito*) con 4 byte.



Il 1° byt è utilizzato per rappresentare l'esponente (con segno) del fattore moltiplicativo.

I restanti tre byte sono utilizzati per rappresentare il valore decimale del numero.

Rappresentazione delle immagini

Sono usate due tecniche per rappresentare le immagini:

- Tecnica a "mappa di punti" o raster -> schermi, fotocamere
- Grafica vettoriale -> disegno tecnico/artistico

GRAFICA RASTER o a mappa di punti

L'immagine viene scomposta in un insieme di elementi di immagini detti **pixel**, a ciascuno dei quali è associato un **singolo colore**.

- ✓Il **pixel** abbreviazione di **picture element** rappresenta la più piccola porzione d'immagine.
- ✓Il **colore** del pixel è definito attraverso le intensità relative dei **tre colori primari R G B**: **R**ed, **G**reen, **B**lue.

✓La qualità di un'immagine raster dipende da due parametri:

-La *risoluzione* che rappresenta il numero di punti o *pixels* di un'immagine. Si misura di solito in DPI = *Dots per inch*. Nel video S.VGA la *risoluzione* è di 1280x1024 = 1 310 270 punti totali.

-La *profondità di colore* rappresenta il numero di *bits* utilizzati per codificare ciascuna tinta. Nel video **S.VGA** sono utilizzati 24 bit per rappresentare le tinte: 8 bit R, 8 bit V, 8 bit B. Inoltre sono utilizzati altri 8 bit per rappresentare le immagini in *trasparenza*.

Il numero complessivo delle tinte che si ottengono con il video S.VGA è di 2^{24} = 16 777 216 tinte.

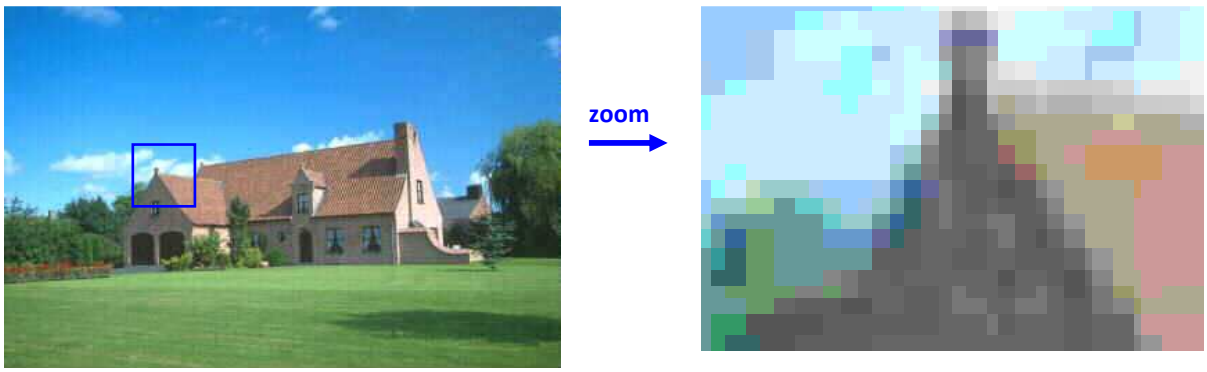
L'occupazione di memoria complessiva di una singola immagine è data da:

occupazione di memoria = n° *punti* × n° *bit colore*

Se l'immagine occupa tutto lo schermo si ottiene:

occupazione di memoria = 1280×1024×24 = 1 310 720 x 24 ≅ 31M Bit ≅ 3,9MByte

Come si vede le immagini raster sono caratterizzate da una elevata occupazione di memoria.



Una seconda caratteristica che le distingue è la dipendenza dalla risoluzione: ingrandendo una porzione di immagine aumenta la dimensione di ciascun punto, perché il numero di pixels della porzione ingrandita resta fisso.

GRAFICA VETTORIALE

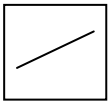
Con la grafica vettoriale le immagini devono essere disegnate.

Per rendere possibile il disegno TUTTI i programmi applicativi di *computer grafica* mettono a disposizione dell'utente un insieme di comandi per il disegno di determinate figure geometriche: **PUNTI**, **LINEE RETTE**, **LINEE CURVE**, **RETTANGOLI**, **POLIGONI**, **CERCHI**, **ELISSI**, ecc.

Quando si va a salvare, mentre la grafica raster codifica la figura geometrica disegnata con l'insieme dei pixels di cui è formata, con la grafica vettoriale, invece, la figura geometrica viene codificata mediante l'equazione matematica che la descrive.

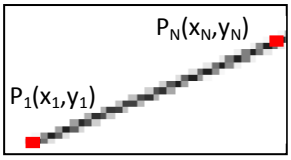
ESEMPIO: Disegno di una linea e sua codifica.

Nella maggior parte degli applicativi la linea è disegnabile cliccando su un comando di disegno, rappresentato da una icona contenente una lineetta, e trascinando sul foglio da disegno:



Icona *line*

=> trascinamento :



Linea sul foglio da disegno

Se si salva l'immagine con un formato raster, allora vengono salvati i punti della linea tracciata (ed anche i punti dello sfondo), ciascuno con il proprio colore. Il fatto che il colore sia o no lo stesso è irrilevante.

p_1	p_2	...	p_N

Se invece si salva l'immagine con un formato vettoriale, allora viene salvato: il nome del comando utilizzato, le coordinate del primo punto, le coordinate dell'ultimo punto, il colore della linea (uno solo) e lo spessore della linea.

Command line	p_1 - p_N	color – weight

Come si distingue un'immagine vettoriale da un'immagine raster:

Ritagliando e ingrandendo una porzione dell'immagine.

- Se l'immagine è a mappa di punti il n° di punti resta fisso: la dimensione dei pixels aumenta.
- Se l'immagine è vettoriale l'equazione matematica associata consente di inserire nuovi punti nell'immagine, che mantiene quindi la sua qualità.

La compressione delle immagini

La compressione è una tecnica che consente di ridurre le quantità di memoria associata ad un file.

Ci sono due tecniche di compressione:

1. **LOSS LESS** (**NO LOSSY**): senza perdita di informazione
2. **LOSSY**: con perdita di informazione

La compressione **Loss Less** utilizza un algoritmo specifico (**LZW**) con il quale è in grado di individuare le (lunghe) sequenze binarie che si ripetono e rinominarle.

Partendo del file compresso è sempre possibile riottenere il file sorgente.

LZW = **L**empel-**Z**iv-**W**elch Sono le iniziali degli inventori dell'algoritmo.

ESEMPI: ZIP – GIF

ZIP: può essere usato con tutti i file: testuali, grafici, con grafica vettoriale e raster.

Altri formati, tipo gif, sono specifici per la grafica e sono utilizzati per comprimere immagini **BMP**.

La tecnica **Lossy** è una tecnica che si utilizza con segnali audio/video e le immagini.

Il segnale viene campionato e analizzato in modo da individuare le sue componenti. Poi vengono codificate le componenti.

Non è possibile ricostruire esattamente l'immagine sorgente dal file compresso.

Più si comprime più diminuisce la qualità.

Introduzione ai problemi

Introduzione ai problemi

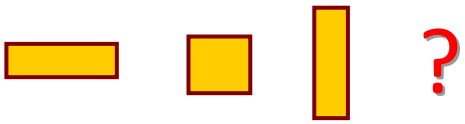
Problema deriva dalle parole greche: pro bòllein che significano: *gettare davanti.*

Indica, genericamente: ***un ostacolo davanti a noi, che ci costringe a riflettere intensamente sul da farsi***.

Inizialmente era circoscritto all'ambito matematico.

(*In ambito matematico*) Problema è: un quesito che richiede di determinare quali enti matematici (*figure geometriche, o numeri, o altro*) soddisfano a condizioni prefissate.

ESEMPIO: *si determini, tra i rettangoli aventi lo stesso perimetro, quello di area massima.*



Successivamente si è esteso in ambito tecnico – scientifico.

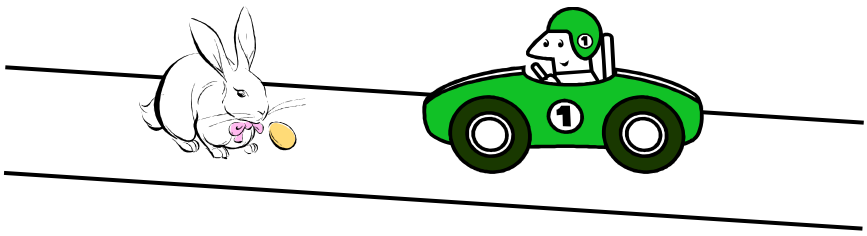
(*In ambito tecnico – scientifico*) Problema è: un quesito che richiede/attende una risposta, non facilmente determinabile.

ESEMPIO: *quanto è grande la luna?*

Infine si è esteso ad ogni contesto, compresa la vita di tutti i giorni, indicando genericamente un qualunque ostacolo.

(*Nel contesto quotidiano*) Rappresenta un problema: un qualunque ostacolo che si frappone nel percorso quotidiano.

ESEMPIO: *il coniglietto nel percorso dell'automobilista a fianco.*



In ciascun ambito tecnico – scientifico, come del resto negli altri ambiti, sono stati sviluppati dei procedimenti per la risoluzione dei rispettivi problemi.

(*Il problema dell'informatico*) L'informatico prende visione, per ciascun ambito, dei problemi e dei procedimenti propri di quell'ambito e cerca di automatizzarli.

Procedimento risolutivo

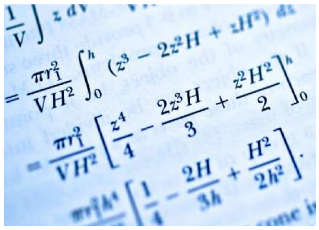
Risolvere un problema significa trovare un **procedimento risolutivo** che fornisce (se esiste) una risposta (esauriente) al problema.

La materia di studio principale dell'informatica è l'individuazione delle caratteristiche dei procedimenti risolutivi dei problemi.

Gli informatici fanno una distinzione tra: **procedimento risolutivo** e **soluzione**.

Per gli informatici :

Il procedimento risolutivo è la "ricetta" con cui viene stabilito quali sono le operazioni/azioni da compiere.



La soluzione richiede l'esecuzione del procedimento risolutivo.

Soluzione = procedimento risolutivo + esecuzione

Corrispondentemente viene fatta una distinzione tra: **RISOLUTORE** ed **ESECUTORE**.

- Il risolutore deve comporre il procedimento risolutivo (informatico);
- L'esecutore deve eseguire (la macchina).

COMPITI DELL' ESECUTORE

- Mettere a punto la sequenza completa, esatta e in ordine cronologico, delle azioni/operazioni => **algoritmo**
- Tradurre l'algoritmo in un linguaggio comprensibile dell'esecutore => **programma**.

COMPITO DEL RISOLUTORE

- **DECODIFICARE** (recepire) le istruzioni impartite dal risolutore.

  Il linguaggio dell'esecutore si chiama "linguaggio macchina"

- **ESEGUIRE** LE ISTRUZIONI

Le doti

Le **doti** richieste ad un "buon" **risolutore** sono:

- (¼) **STRUMENTI**: conoscenze ed abilità specifiche
  per far fronte con efficacia ad un problema bisogna avere un minimo di conoscenze specifiche sull'argomento. Ciò che sappiamo e sappiamo fare sono i nostri strumenti.
- (¼) **RAGIONAMENTO LOGICO**: logica induttiva/deduttiva, creatività
  le capacità logiche e la creatività ci sono indispensabili per rintracciare una qualunque soluzione
- (¼) **METODO:** modo con cui si procede
  anche gli schemi di comportamento che seguiamo intanto che cerchiamo le soluzioni hanno il loro peso
- (¼) **ATTEGGIAMENTO**: coinvolgimento, volontà, curiosità
  è molto importante farsi carico del problema, cioè sentirlo come un nostro problema e non un problema di qualcun altro e volerlo risolvere. Quando non si riesce a risolverlo in poco tempo, non bisogna arrendersi e gettare subito la spugna, ma piuttosto credere nelle proprie possibilità di riuscire.

Anche all'esecutore sono richieste delle "doti":

☺Prontezza e velocità di esecuzione

☺Buona tecnica

☺Precisione

☺Non deve fare domande

☺Non deve prendere iniziative autonome

L'esecutore deve essere innanzi tutto molto veloce e deve avere una buona tecnica di esecuzione. Poi deve eseguire quanto gli è stato detto di fare, senza interrogarsi su ciò che sta facendo e senza prendere delle proprie iniziative che vadano oltre il compito che gli è stato affidato.

LA RISOLUZIONE DI UN PROBLEMA

CONSEGNA: *ad una classe di 2ª media è chiesto di disegnare un quadrato avente area di 36 cm².*

Individuare un procedimento risolutivo informatizzato del problema.

Per affrontare il problema adottiamo un protocollo metodologico suddiviso nelle seguenti fasi:

1) FASE: svolgimento usuale, non informatizzato.
si vede come il problema viene risolto abitualmente, in modo non informatizzato.

2) FASE: analisi generale (*quali sono gli ingredienti* ?)
si esamina la soluzione abituale, non informatizzata e si ricercano gli ingredienti da inserire nella "ricetta".

3) FASE: analisi specifica (*come vanno combinati gli ingredienti*?) → ***algoritmo***
si cerca come comporre gli ingredienti per trovare, se esiste, un procedimento risolutivo. La "ricetta" prodotta è l'algoritmo risolutivo.

4) FASE: sintesi (*implementazione dell'algoritmo risolutivo*) → ***programma***
si traduce l'algoritmo risolutivo in un linguaggio di programmazione

Svolgimento usuale

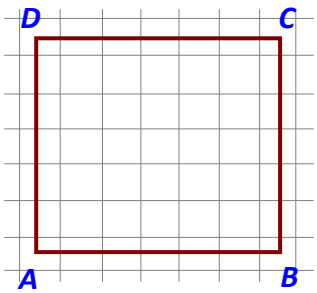
(L'alunno di 2ª media segna il dato dell'area, quindi si trova il lato e disegna il quadrato)

PROBLEMA: disegna un quadrato avente area di 36 cm²

Dati: $\mathcal{A} = 36\ cm^2$
Lato = ?

Trovo il lato:
$L = \sqrt{36} = 6cm$

Disegno il quadrato



analisi generale: quali sono gli ingredienti?

Per giungere a disegnare il quadrato l'alunno ha avuto bisogno:

- del dato dell'**Area**: *dato di input*
- del dato del **Lato**: *elaborazione*
- di un **Foglio da disegno** quadrettato: *dato di input*
- di una **Penna**: *dato di input*

Inoltre l'alunno deve sapere due cose:

- come si estrae la **Radice quadrata**
  per calcolare il lato
- come è fatto un **Quadrato**
  per poterlo disegnare

Alla fine l'alunno:

- **Disegna** il **Quadrato** con il lato che si è calcolato sul suo foglio: *dato di output*

Tutti gli ingredienti (**dati/oggetti**) che l'esecutore introduce dall'esterno sono stati classificati come: *dati di input*.

Tutti gli ingredienti (**dati/oggetti**) che l'esecutore elabora a partire da dati che già possiede sono stati classificati come: *elaborazione*.

Tutti gli ingredienti (**dati/oggetti**) che l'esecutore restituisce verso l'esterno sono stati classificati come: *dati di output*.

È importante tenere presente che la ricetta che verrà individuata non è a beneficio del risolutore, ma dell'esecutore (la macchina) e quindi, già da questa fase è bene che il risolutore ne tenga conto.

L'informatico ha la necessità di specificare ***tutti gli ingredienti***, anche quelli banali, che di solito sono lasciati sottintesi, perché si suppone che l'esecutore ***non*** sia in grado di procurarseli autonomamente.

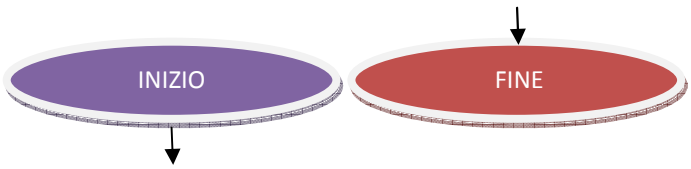
ALLA MACCHINA BISOGNA SPECIFICARE TUTTO

ANALISI SPECIFICA:
Come risolvere un problema, cioè con quale algoritmo

Per descrivere graficamente l'algoritmo si utilizzano i: ***diagrammi di flusso***, comprendenti i seguenti simboli grafici standard.

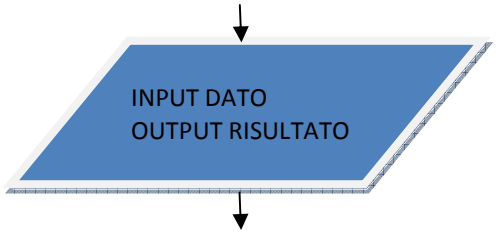
OVALE

Indicando l'inizio o la fine dell'algoritmo



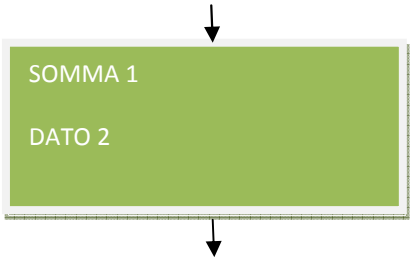
PARALLELOGRAMMA

Indica l'input di un dato o l'output di un risultato



RETTANGOLO

Indica una operazione/ elaborazione specifica



ROMBO

Blocco condizionale

schema dell'algoritmo per il disegno del quadrato

INIZIA DA QUI

INTRODUCI IL DATO DELL'AREA (il dato dell'area è un dato del problema, che deve essere memorizzato).

CALCOLA IL LATO (il lato è dato dalla radice quadrata dell'area).

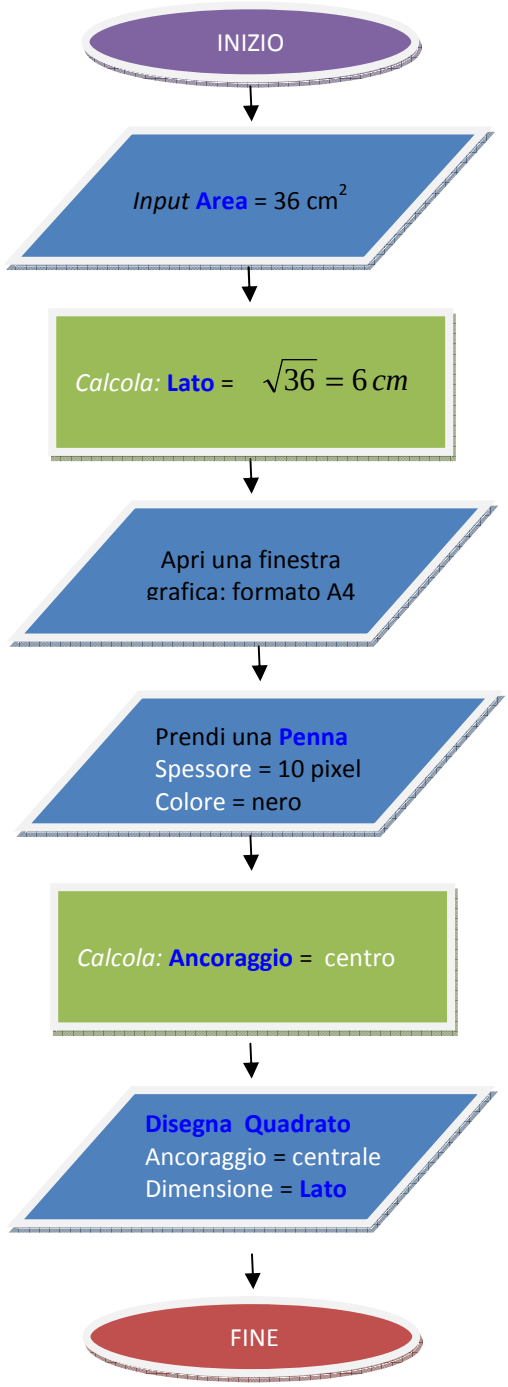
*PRENDI UN FOGLIO DA DISEGNO QUADRETTATO (la finestra grafica ha: un formato dato da **lunghezza** × **larghezza** e un sistema interno di coordinate cartesiane. Può avere un colore di sfondo).*

*PRENDI UNA PENNA (la penna ha un suo **spessore** ed un suo **colore**).*

È NECESSARIO DECIDERE IN QUALE PUNTO DEL FOGLIO INSERIRE IL QUADRATO (punto di ancoraggio).

DISEGNO DEL QUADRATO (è necessario specificare: il punto in cui appendere la figura e le dimensioni dei lati.)

FINE DELL'ALGORITMO



Commento: *inserire una figura su un foglio richiede le stesse azioni richieste per appendere un quadro alla parete: prima va fissato un "gancio" di sostegno nella parete e poi va fissato un "gancetto" anche sul quadro, che andrà infilato nel gancio a parete.*

SINTESI - IMPLEMENTAZIONE CON IL LINGUAGGIO PROCESSING

Una volta realizzato l'algoritmo deve essere tradotto in un **linguaggio di programmazione**.

Usiamo il **linguaggio di programmazione di grafica** PROCESSING

Il programma è **Open Source** e può essere scaricato liberamente dal sito: **http://processing.org/.**

Si presenta con una interfaccia molto semplice, con una finestra (*sketch*) nella quale deve essere inserito il codice per il disegno delle figure. Nella parte inferiore dell'interfaccia grafica sono riportati commenti e debug. Un primo pulsante consente (se il codice è sintatticamente corretto) l'avvio del programma. Durante l'esecuzione si apre una finestra grafica le cui dimensioni possono essere specificate dal programmatore. Un secondo pulsante consente l'arresto dell'esecuzione del programma. In alto è riportata una barra con i principali campi per la gestione dell'applicazione: *File*, *Edit*, *Sketch*, *Tools*, *Help*. Da quest'ultimo si accede al campo *Reference*, che rimanda alla **lista** completa **delle istruzioni**.

Prendiamo dalla lista alcuni COMANDI DI PROCESSING

// qualsiasi testo	*Commento*
size(width, height); size(120,120);	*finestra grafica rettangolare; es(120,120)*
background(B/N); background(R, G, B);	*Sfondo Black / White: Sfondo colore R,G,B*
stroke(Gray); stroke(R,G,B); stroke (R,G,B,T);	*Colore del tratto*
strokeWeight(PixelWeight);	*Spessore in Pixel del tratto*
Smooth(); noSmooth();	*Tratto smussato; non smussato*
fill(Gray); fill(R,G,B); fill (R,G,B,T);	*Riempimento delle figure*
noStroke(); noFill();	*Nessun tratto; nessun riempimento*
line(x_1, y_1, x_2, y_2);	*linea retta da P_1 a P_2*
rect(x_C, y_C, width, height);	*rettangolo (o quadrato) ancorato in P_C*
ellipse(x_C, y_C, width, height);	*ellisse (o cerchio) ancorato in P_C*
rectMode(CENTER); ellipseMode(CENTER);	*Imposta l'ancoraggio*
int x = 25; float z = 23,5;	*dichiarazione di variabile e assegnazione*
sqrt(x);	$\sqrt{x}$

il programma

Con i comandi di processing scriviamo nello **sketch** il codice del nostro algoritmo:

Il **programma** è suddiviso in blocchi ciascuno dei quali svolge il compito indicato nell'algoritmo.

Ogni riga di codice è **commentata**, per una facile lettura, ma il commento sarà ignorato nell'esecuzione.

```
//Programma Disegna Quadrato
//INIZIO
//Input Area
float Area;                 //Apre un campo dati per l'area – formato numerico in virgola mobile
Area = 36;                  //Assegna all'area il suo valore: 36 cm^2
//Calcolo del lato
float Lato;                 //Apre un campo dati per il lato – formato numerico in virgola mobile
Lato = sqrt(Area);          //Il lato è la radice quadrata dell'area: 6cm
Lato = Lato * 30;           //Conversione del valore in pixel – 30 pixel per centimetro
//Apri la finestra grafica
size(900,600);              //Apre una finestra grafica di 900 × 600 pixel: circa 30 cm × 20 cm
background(255);            //Sfondo della finestra nero
//Prendi una penna
stroke(0);                  //Prende una penna di colore nero
strokeWeight(20);           //e con spessore di 20 pixel – circa 6 mm
fill(255,0,0);              //Le figure disegnate avranno riempimento rosso
smooth();                   //Le linee del disegno saranno smussate
//Ancoraggio nella finestra grafica
float xc = 400;             //Coordinata X del punto di ancoraggio a metà della lunghezza della finestra
float yc = 300;             //Coordinata Y del punto di ancoraggio a metà della larghezza della finestra
//Disegno del quadrato
rectMode(CENTER);           //Fissa il "gancetto" nei rettangoli che disegnerai nel punto centrale
rect(xc,yc,Lato,Lato);      //Disegna il quadrato: il quadrato è un rettangolo con base = altezza = lato
//FINE
```

La grafica interattiva e di animazione

- La **grafica interattiva** è un tipo di grafica in cui **l'utente può modificare parti del disegno**, interagendo con il sistema attraverso: il **mouse**, la **tastiera,** ...
- La **grafica di animazione** è un tipo di grafica in cui **l'immagine si modifica nel tempo**.
- **L'animazione** è ottenuta costruendo la grafica mediante un insieme di **fotogrammi,** o **diapositive,** o "**frame**", che vengono **visualizzati in successione temporale** con una ben determinata velocità di scansione o "**frame rate**".

Il limite fisico del frame rate è legato al tipo di scheda grafica utilizzata dal sistema: 50 fotogrammi al secondo.

Gestione del mouse: (riferimento all'ambiente *processing*)

Gli elementi del mouse che solitamente vengono gestiti sono:

- **Puntatore del mouse** viene gestito attraverso le coordinate x e y del punto di puntamento dello schermo. mouseX e mouseY sono due variabili predefinite di *processing* con formato di tipo: integer (INT)
- **Tasti del mouse**: una prima funzione permette di determinare se il tasto del mouse è stato o no premuto: se un tasto qualsiasi del mouse è premuto la variabile *mousePressed* viene settata a true. Se nessun tasto è premuto allora *mousePressed* è resettata a false.
  La variabile *mouseBotton* consente di discriminare tra tasto destro (right) e tasto sinistro (left).

Gestione della tastiera

- L'evento "tasto premuto" viene rilevato dalla variabile *KeyPressed*:

KeyPressed =
- TRUE => un tasto premuto
- FALSE => nessun tasto premuto

- Quale tasto è stato premuto viene rilevato dalla variabile *Key*. Il formato di *Key* e di tipo: CHR

LA GESTIONE DEI FRAME

Nella **grafica interattiva** e/o **di animazione** è necessario implementare **più fotogrammi** contenenti figure in movimento.

Questo problema viene risolto dalla gran parte dei programmi di grafica interattiva **ridisegnando** ciclicamente **le figure**, con i parametri aggiornati.

Ciò viene fatto ripetendo le istruzioni richieste per il disegno delle figure.

A tale scopo **i programmi di grafica interattiva** comprendono delle specifiche funzioni per la ripetizione ciclica di istruzioni di disegno.

La funzione draw()

In *processing* la funzione che consente la ripetizione ciclica di istruzioni di disegno si chiama: *void draw()*.

Essa va utilizzata nel seguente modo:

```
void draw() {  // inizio della funzione con aperta parentesi graffa
istruzione 1 …  // sequenza di istruzioni di disegno
            }   // fine della funzione con chiusa parentesi graffa
```

Le istruzioni da ripetere ciclicamente vanno inserite tra le due parentesi graffe.

Ogni volta che la funzione *draw()* viene messa in esecuzione, dopo aver eseguito il codice contenuto tra le due parentesi il programma esce dalla funzione, salta, e rientra nuovamente nella funzione.

Per evitare che quando rientra nella funzione non ridisegni esattamente le stesse figure è fondamentale che i parametri delle figure vengano modificati.

Ciò viene fatto in due modi:

- *Primo modo* (grafica interattiva): alcuni dei parametri sono associati a mouse e tastiera.
- *Secondo modo* (grafica di animazione): i parametri vengono modificati o subito prima o subito dopo il codice per il disegno delle figure.

La funzione setup()

Di solito però, **nella grafica interattiva** non tutte le figure sono in movimento, ma alcune parti del disegno, a volte la gran parte, sono fisse.

Per evitare che anche alcune parti fisse del disegno vegano inutilmente ripetute, il relativo codice, invece di inserirlo all'interno della funzione *draw()*, conviene inserirlo in funzioni che lo eseguono una sola volta.

In *processing* la funzione che consente la esecuzione di istruzioni di disegno una sola volta si chiama: *void setup()*.

Essa va utilizzata nel seguente modo:

```
void setup()   {   //inizio della funzione con aperta parentesi graffa
Istruzione 1 …      //sequenza di istruzioni di disegno
               }   //fine della funzione con chiusa parentesi graffa
```

Le istruzioni da eseguire una sola volta vanno inserite tra le due parentesi graffe.

Per consentire a tutte le funzioni di operare con i parametri delle figure, i parametri devono essere introdotti esternamente alle funzioni, di solito all'inizio.

conclusione

In conclusione, per realizzare una **immagine interattiva** o di **animazione** bisogna suddividere il codice in tre parti:

- **La prima parte** contenente il codice per l'introduzione dei dati pubblici delle figure va inserita all'inizio.
- **La seconda parte** contenente il codice per il disegno delle parti fisse, va inserita nella funzione *setup()*.
- **La terza parte** contenente il codice per il disegno delle parti in movimento, va inserita nella funzione *draw()*.

Il programma interattivo

ESEMPIO: *modifichiamo il programma che disegna un quadrato per renderlo interattivo.*

Vogliamo che l'utente possa decidere con il mouse dove posizionare il quadrato.

```
//Programma Disegna Quadrato Interattivo
//INIZIO
//PRIMA PARTE: dati del quadrato
//Input Area
float Area = 36;          //Apre il campo dati per l'area ed assegna all'area il suo valore: 36 cm2
//Input Lato
float Lato;               //Apre un campo dati per il lato  (qui non si può calcolare il valore)
//SECONDA PARTE: parte fissa del disegno
Void setup() {            //Inizio parte fissa
//Apri la finestra grafica (si fa una sola volta)
size(900,600);            //Apre una finestra grafica di 900 × 600 pixel: circa 30 cm × 20 cm
background(255);          //Sfondo della finestra nero
//Prendi una penna (si fa una sola volta)
stroke(0);                //Prende una penna di colore nero
strokeWeight(20);         //e con spessore di 20 pixel – circa 6 mm
fill(255,0,0);            //Le figure disegnate avranno riempimento rosso
smooth();                 //Le linee del disegno saranno smussate
//calcola lato (si può fare qui)
Lato = sqrt(Area);        //Il lato è la radice quadrata dell'area: 6cm
Lato = Lato * 30;         //Conversione del valore in pixel – 30 pixel per centimetro
//Ancoraggio nella finestra grafica (si fa nel punto puntato dal mouse, quindi non serve)
}                         //fine parte fissa
//TERZA PARTE: parte variabile del disegno
Void draw() {             //Inizio parte variabile
//Disegno del quadrato (si fa più volte)
rectMode(CENTER);         //Fissa il "gancetto" nei rettangoli che disegnerai nel punto centrale
if (mousePressed)         //Solo se un tasto del mouse è premuto – la condizione va messa tra parentesi
rect(mouseX, mouseY, Lato, Lato); //Disegna il quadrato: base = altezza = lato
}                         //fine parte variabile
//FINE
```