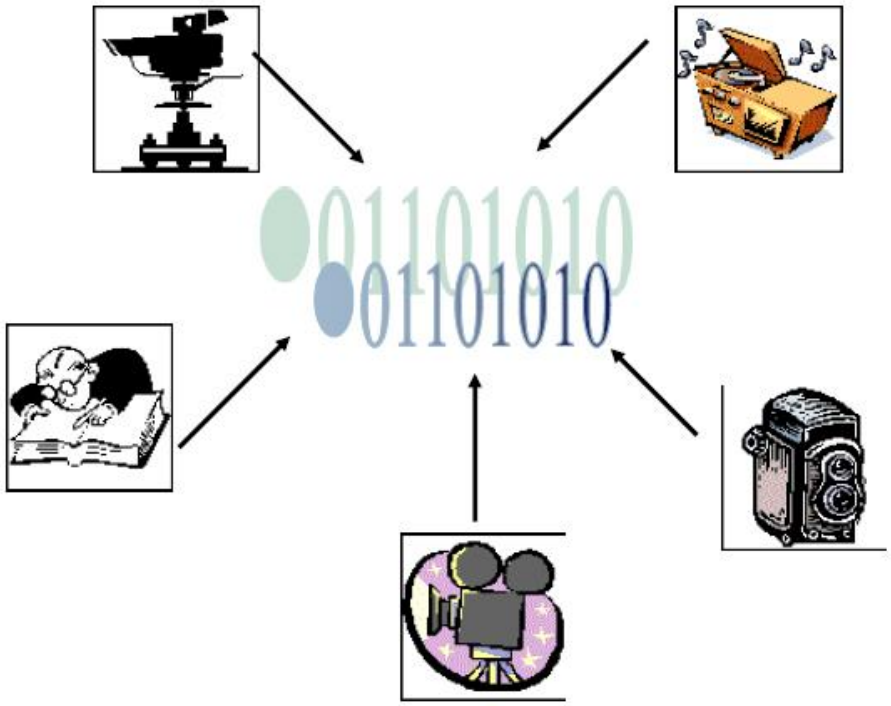
La rappresentazione delle informazioni



In queste pagine cercheremo di capire come sia possibile rappresentare mediante numeri e memorizzare in un file testi, immagini, video, suoni
Il computer per lavorare sui dati, ha bisogna che questi siano espressi come sequenze di 1 e di 0. L'operazione di trasformazione dei dati in sequenze di 1 e di 0, cioè in numeri, prende il nome di ***procedimento di codifica***.

1 Rappresentazione dei numeri

La maniera in cui vengono rappresentati i numeri in un computer l'abbiamo già vista. Si tratta di scriverli in rappresentazione binaria.

2 Rappresentazione di caratteri e stringhe

Dato che qualsiasi informazione memorizzata in un computer o trasmessa in rete è rappresentata da codici numerici, anche i caratteri e le stringhe (insieme di più caratteri) che costituiscono un testo hanno una codifica numerica. La rappresentazione dei caratteri alfanumerici presenti sulla tastiera del computer

avviene mediante dei sistemi che associano ad ogni carattere (lettera, numero, segni di interpunzione e caratteri speciali) un codice binario.

Una dei primi sistemi che vennero usati per trasformare i caratteri in sequenze di 1 e di 0 è la codifica **ASCII** (**A**merican **S**tandard **C**ode for **I**nformation **I**nterchange) a 7 bit. Avendo a disposizione 7 bit per la codifica dei caratteri è possibile codificare 128 combinazioni diverse! Infatti $2^7 = 128$.
La codifica ASCII è una tabella di corrispondenza tra simboli e numeri, viene detta infatti *codifica dei caratteri*. I **primi 32** caratteri del codice ASCII (con codice da 0 a 31) sono **caratteri di controllo, i** caratteri **da 32 a 127** sono **caratteri stampabili**:

Byte	Cod.	Char
00000000	0	Null
00000001	1	Start of heading
00000010	2	Start of text
00000011	3	End of text
00000100	4	End of transmit
00000101	5	Enquiry
00000110	6	Acknowledge
00000111	7	Audible bell
00001000	8	Backspace
00001001	9	Horizontal tab
00001010	10	Line feed
00001011	11	Vertical tab
00001100	12	Form Feed
00001101	13	Carriage return
00001110	14	Shift out
00001111	15	Shift in
00010000	16	Data link escape
00010001	17	Device control 1
00010010	18	Device control 2
00010011	19	Device control 3
00010100	20	Device control 4
00010101	21	Neg. acknowledge
00010110	22	Synchronous idle
00010111	23	End trans. block
00011000	24	Cancel
00011001	25	End of medium
00011010	26	Substitution
00011011	27	Escape
00011100	28	File separator
00011101	29	Group separator
00011110	30	Record Separator
00011111	31	Unit separator

Byte	Cod.	Char
00100000	32	Spc
00100001	33	!
00100010	34	"
00100011	35	#
00100100	36	$
00100101	37	%
00100110	38	&
00100111	39	'
00101000	40	(
00101001	41	)
00101010	42	*
00101011	43	+
00101100	44	,
00101101	45	-
00101110	46	.
00101111	47	/
00110000	48	0
00110001	49	1
00110010	50	2
00110011	51	3
00110100	52	4
00110101	53	5
00110110	54	6
00110111	55	7
00111000	56	8
00111001	57	9
00111010	58	:
00111011	59	;
00111100	60	<
00111101	61	=
00111110	62	>
00111111	63	?

Byte	Cod.	Char
01000000	64	@
01000001	65	A
01000010	66	B
01000011	67	C
01000100	68	D
01000101	69	E
01000110	70	F
01000111	71	G
01001000	72	H
01001001	73	I
01001010	74	J
01001011	75	K
01001100	76	L
01001101	77	M
01001110	78	N
01001111	79	O
01010000	80	P
01010001	81	Q
01010010	82	R
01010011	83	S
01010100	84	T
01010101	85	U
01010110	86	V
01010111	87	W
01011000	88	X
01011001	89	Y
01011010	90	Z
01011011	91	[
01011100	92	\
01011101	93	]
01011110	94	^
01011111	95	_

Byte	Cod.	Char
01100000	96	`
01100001	97	a
01100010	98	b
01100011	99	c
01100100	100	d
01100101	101	e
01100110	102	f
01100111	103	g
01101000	104	h
01101001	105	i
01101010	106	j
01101011	107	k
01101100	108	l
01101101	109	m
01101110	110	n
01101111	111	o
01110000	112	p
01110001	113	q
01110010	114	r
01110011	115	s
01110100	116	t
01110101	117	u
01110110	118	v
01110111	119	w
01111000	120	x
01111001	121	y
01111010	122	z
01111011	123	{
01111100	124	|
01111101	125	}
01111110	126	~
01111111	127	Del

La codifica ASCII inizialmente prevedeva solo i caratteri della lingua inglese e 7 bit erano sufficienti a codificarli tutti infatti l'alfabeto anglosassone {a,b,c, …A,B,C, %, &, (,),…0,1,2,3,..,; ?+,-*,…}, è formato da:
- 26 lettere maiuscole + 26 minuscole
- 10 cifre
- circa 30 segni d'interpunzione
- circa 30 caratteri di controllo (EOF, CR, LF, …)

In seguito il codice ASCII è stato però esteso ad 8 bit, (2^8=256 caratteri) questo codice viene chiamato **ASCII esteso**. I codici che vanno da 0 a 127 che hanno uno 0 iniziale, sono gli stessi del codice ASCII su 7 bit, gli altri (da 128 a 255) che hanno un 1 iniziale sono utilizzati per codificare caratteri speciali. Eccoli di seguito:

Byte	Cod.	Char	Byte	Cod.	Char	Byte	Cod.	Char	Byte	Cod.	Char
10000000	128	Ç	10100000	160	á	11000000	192	+	11100000	224	Ó
10000001	129	ü	10100001	161	í	11000001	193	-	11100001	225	ß
10000010	130	é	10100010	162	ó	11000010	194	-	11100010	226	Ô
10000011	131	â	10100011	163	ú	11000011	195	+	11100011	227	Ò
10000100	132	ä	10100100	164	ñ	11000100	196	-	11100100	228	õ
10000101	133	à	10100101	165	Ñ	11000101	197	+	11100101	229	Õ
10000110	134	å	10100110	166	ª	11000110	198	ã	11100110	230	µ
10000111	135	ç	10100111	167	º	11000111	199	Ã	11100111	231	þ
10001000	136	ê	10101000	168	¿	11001000	200	+	11101000	232	Þ
10001001	137	ë	10101001	169	®	11001001	201	+	11101001	233	Ú
10001010	138	è	10101010	170	¬	11001010	202	-	11101010	234	Û
10001011	139	ï	10101011	171	½	11001011	203	-	11101011	235	Ù
10001100	140	î	10101100	172	¼	11001100	204	¦	11101100	236	ý
10001101	141	ì	10101101	173	¡	11001101	205	-	11101101	237	Ý
10001110	142	Ä	10101110	174	«	11001110	206	+	11101110	238	¯
10001111	143	Å	10101111	175	»	11001111	207	¤	11101111	239	´
10010000	144	É	10110000	176	▒	11010000	208	ð	11110000	240	
10010001	145	æ	10110001	177	▒	11010001	209	Ð	11110001	241	±
10010010	146	Æ	10110010	178	▒	11010010	210	Ê	11110010	242	‗
10010011	147	ô	10110011	179	¦	11010011	211	Ë	11110011	243	¾
10010100	148	ö	10110100	180	¦	11010100	212	È	11110100	244	¶
10010101	149	ò	10110101	181	Á	11010101	213	ı	11110101	245	§
10010110	150	û	10110110	182	Â	11010110	214	Í	11110110	246	÷
10010111	151	ù	10110111	183	À	11010111	215	Î	11110111	247	¸
10011000	152	ÿ	10111000	184	©	11011000	216	Ï	11111000	248	º
10011001	153	Ö	10111001	185	¦	11011001	217	+	11111001	249	¨
10011010	154	Ü	10111010	186	¦	11011010	218	+	11111010	250	·
10011011	155	ø	10111011	187	+	11011011	219	▒	11111011	251	¹
10011100	156	£	10111100	188	+	11011100	220	▒	11111100	252	³
10011101	157	Ø	10111101	189	¢	11011101	221	¦	11111101	253	²
10011110	158	×	10111110	190	¥	11011110	222	Ì	11111110	254	■
10011111	159	ƒ	10111111	191	+	11011111	223	▒	11111111	255	

Se un carattere non è presente sulla tastiera del computer è possibile inserirlo digitando ALT + numero decimale corrispondente al carattere ASCII (da tastierino numerico). Ad esempio, ALT + 123 consente di inserire il carattere {.
Di seguito è riportato il codice ASCII cin la codifica decimale:

0		32		64	@	96	`	128	Ç	160	á	192	└	224	Ó
1	☺	33	!	65	A	97	a	129	ü	161	í	193	┴	225	ß
2	☻	34	"	66	B	98	b	130	é	162	ó	194	┬	226	Ô
3	♥	35	#	67	C	99	c	131	â	163	ú	195	├	227	Ò
4	♦	36	$	68	D	100	d	132	ä	164	ñ	196	─	228	õ
5	♣	37	%	69	E	101	e	133	à	165	Ñ	197	┼	229	Õ
6	♠	38	&	70	F	102	f	134	å	166	ª	198	ã	230	µ
7	•	39	'	71	G	103	g	135	ç	167	º	199	Ã	231	þ
8	◘	40	(	72	H	104	h	136	ê	168	¿	200	╚	232	Þ
9	○	41	)	73	I	105	i	137	ë	169	®	201	╔	233	Ú
10	◙	42	*	74	J	106	j	138	è	170	¬	202	╩	234	Û
11	♂	43	+	75	K	107	k	139	ï	171	½	203	╦	235	Ù
12	♀	44	,	76	L	108	l	140	î	172	¼	204	╠	236	ý
13	♪	45	-	77	M	109	m	141	ì	173	¡	205	═	237	Ý
14	♫	46	.	78	N	110	n	142	Ä	174	«	206	╬	238	¯
15	☼	47	/	79	O	111	o	143	Å	175	»	207	¤	239	´
16	►	48	0	80	P	112	p	144	É	176	░	208	ð	240	-
17	◄	49	1	81	Q	113	q	145	æ	177	▒	209	Ð	241	±
18	↕	50	2	82	R	114	r	146	Æ	178	▓	210	Ê	242	‗
19	‼	51	3	83	S	115	s	147	ô	179	│	211	Ë	243	¾
20	¶	52	4	84	T	116	t	148	ö	180	┤	212	È	244	¶
21	§	53	5	85	U	117	u	149	ò	181	Á	213	ı	245	§
22	▬	54	6	86	V	118	v	150	û	182	Â	214	Í	246	÷
23	↨	55	7	87	W	119	w	151	ù	183	À	215	Î	247	¸
24	↑	56	8	88	X	120	x	152	ÿ	184	©	216	Ï	248	°
25	↓	57	9	89	Y	121	y	153	Ö	185	╣	217	┘	249	¨
26	→	58	:	90	Z	122	z	154	Ü	186	║	218	┌	250	·
27	←	59	;	91	[	123	{	155	ø	187	╗	219	█	251	¹
28	∟	60	<	92	\	124	|	156	£	188	╝	220	▄	252	³
29	↔	61	=	93	]	125	}	157	Ø	189	¢	221	¦	253	²
30	▲	62	>	94	^	126	~	158	×	190	¥	222	Ì	254	■
31	▼	63	?	95	_	127	⌂	159	ƒ	191	┐	223	▀	255	

Le **parole** sono sequenze di caratteri e come tali sequenze di sequenze di bit. Una stringa di caratteri sarà rappresentata dal computer come una successione di gruppi di 8 bit. Esempi:

"Ciao" = 01000011 01101001 01100001 01101111
"24" = 00110001 00110011
"3 kg" = 00110011 00100000 01101011 01100111
"cane" = 01100011 01100001 01101110 01100101

O	G	G	I		P	I	O	V	E
01001111	01000111	01000111	01001001	00100000	01010000	01001001	01001111	01010110	01000101

Tra i simboli speciali del codice ASCII vi è anche il simbolo spazio bianco "NUL"(codice 00100000), il simbolo di fine riga "CR" (00001101). In questo modo è possibile rappresentare mediante una sequenza di codici ASCII un testo strutturato in righe e pagine

Decodifica: quale testo è codificato da una data sequenza?
- si divide la sequenza in gruppi di otto bit (un byte)
- si determina il carattere corrispondente ad ogni byte

01101001011011000010000001110000011011110010111 0
01101001011011000010000001110000011011110010111 0
i l P o .

La decodifica è possibile perché i caratteri sono codificati con stringhe binarie di lunghezza costante.
Con il codice ASCII è possibile rappresentare i numeri come sequenza di caratteri. Ad esempio il numero 234 sarà rappresentato come: 00110010 00110011 00110100
2 3 4

La tabella ASCII fu creata sostanzialmente per poter scrivere in inglese. Come potete notare infatti mancano molti simboli, per esempio la codifica ASCII non è sufficiente per poter scrivere perfettamente in italiano perché mancano le lettere accentate. Figuriamoci altre lingue come l'arabo, il cinese, il giapponese, ecc...
Per questo motivo, oggigiorno si usa un'altra *codifica dei caratteri*, chiamata **UNICODE** (http://www.unicode.org) che utilizza 16 bit (65536 caratteri). Contiene tutti i simboli per tutte le scritture del mondo (arabo, ebraico, cinese, giapponese,

coreano, tailandese….) e non solo, contiene anche simboli per scrivere la matematica, alfabeti fonetici, lingue morte, ecc…
I primi 128 simboli UNICODE sono identici a quelli dell'ASCII per motivi di compatibilità con il passato i successivi corrispondono ad altri alfabeti. Non riesce in ogni caso a coprire i simboli (oltre 200.000) di tutte le lingue!

- **7 bit (ASCII standard)**
- **8 bit [1byte] (ASCII esteso)**
- **16 bit [2byte] (UNICODE)**

Domande???

1. Nell'alfabeto di Marte sono previsti 300 simboli; quanti bit si devono utilizzare per rappresentarli tutti?
2. Quanti byte occupa la frase "**biologia marina**" se la si codifica utilizzando il codice ASCII?
3. Quanti byte occupa la stessa frase scritta in codice UNICODE?
4. Dati 12 bit per la codifica, quante informazioni distinte si possono rappresentare?
5. Cosa rappresenta la stringa
"01101001011011100110011001101111011100100110110101110100011000010110100101100011011000 01"?

Risposte!!!

1. L'esercizio richiede di trovare il numero di bit che sono necessari per codificare **300 informazioni diverse**.
   Dobbiamo quindi applicare la formula $\mathbf{2^N >= M}$ e ricavare N.
   $2^N >= 300$ se N=9
2. Poiché sappiamo che ogni carattere in codice ASCII esteso occupa **un byte** dobbiamo **contare il numero di caratteri** (inclusi gli spazi bianchi) che formano la frase "biologia marina" e moltiplicare per 1 byte.
   15 caratteri ->15 x 1 byte = 15 byte
3. 15 caratteri ->15 byte
   Poiché ogni carattere in codice UNICODE occupa **due byte** avremo :
   15 caratteri ->15 x 2 byte = 30 byte
4. In questo caso conosciamo la **lunghezza delle sequenze** di bit che sono usate per la codifica dell'informazione e basterà applicare la formula $\mathbf{2^N}$ per trovare il numero di informazioni distinte che si possono rappresentare $2^{12} = 4096$
5. Dividendo la stringa in gruppetti di 8 bit (parto da sinistra) e determino il carattere corrispondente ad ogni byte, ottengo in questo modo la parola "informatica".

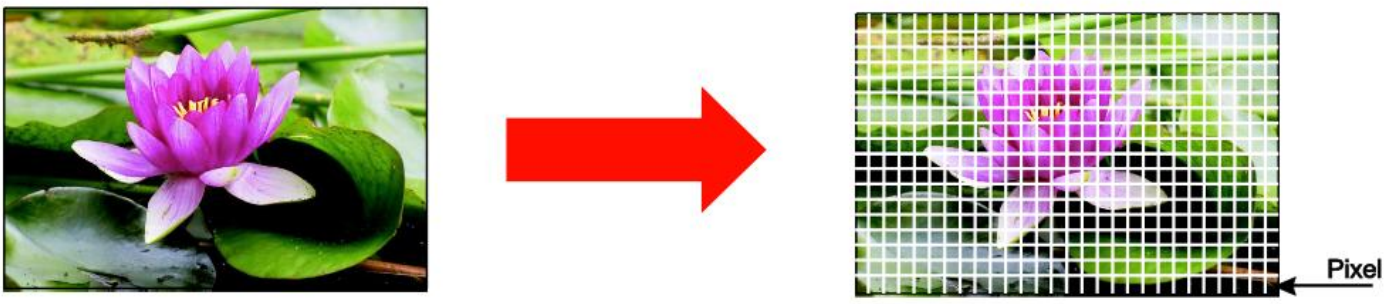
Esercizi:

1. Quanti byte occupa la frase "Ciao Ragazzi!!!" se la codifichiamo utilizzando ASCII?
   Quanti byte occupa se la codifichiamo in ASCII standard?
   Quanti byte occupa se la codifichiamo in UNICODE?
2. Codificate in ASCII esteso la frase "Vado in palestra".
3. Cosa rappresenta la stringa 01100011011000010110111001100101?
4. Nell'alfabeto di Terra Gemella ci sono 340 simboli distinti. Quanti bit si devono utilizzare per codificare tutti i simboli dell'alfabeto? Di quanti bit abbiamo bisogno per codificare un testo di 2500 caratteri scritto nell'alfabeto di Terra gemella?

3 Rappresentazione delle immagini

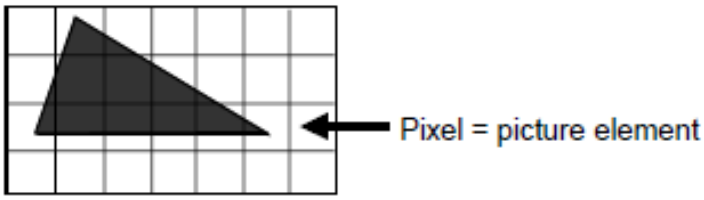
Anche le immagini possono essere codificate mediante una sequenza di 0 e 1., questa operazione si chiama **digitalizzazione**.
La prima cosa da fare è dividere l'immagine in tanti quadratini, che prendono il nome di PIXEL (Picture Element). Una volta specificata la dimensione in pixel dell'immagine è sufficiente elencare in ordine il colore di ogni pixel.

- Per le immagini in bianco e nero si assegna ad ogni pixel un bit
- Per le immagini a colori si assegna ad ogni pixel una sequenza di bit

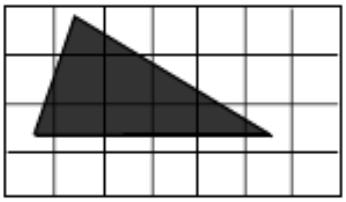


CODIFICA IMMAGINI IN BIANCO E NERO

Consideriamo un'immagine in bianco e nero, senza ombreggiature o livelli di chiaroscuro. L'immagine viene suddivisa mediante una griglia formata da righe orizzontali e verticali a distanza costante.



Il simbolo "0" viene utilizzato per la codifica di un pixel corrispondente ad un quadratino bianco (in cui il bianco è predominante), il simbolo "1" viene utilizzato per la codifica di un pixel corrispondente ad un quadratino nero (in cui il nero è predominante).

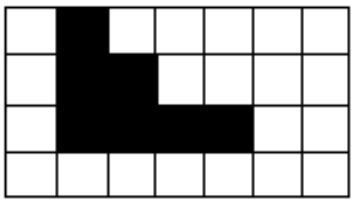


0 22	1 23	0 24	0 25	0 26	0 27	0 28
0 15	1 16	1 17	0 18	0 19	0 20	0 21
0 8	1 9	1 10	1 11	1 12	0 13	0 14
0 1	0 2	0 3	0 4	0 5	0 6	0 7

Si deve definire una convenzione per ordinare i pixel della griglia ipotesi: assumiamo che i pixel siano ordinati dal basso verso l'alto e da sinistra verso destra.
La rappresentazione della figura è data dalla stringa binaria:

0000000 0111100 0110000 0100000

Dato che il contorno della figura non sempre coincide con la griglia si ottiene un'approssimazione della figura originaria. Riconvertendo la stringa si avrà:

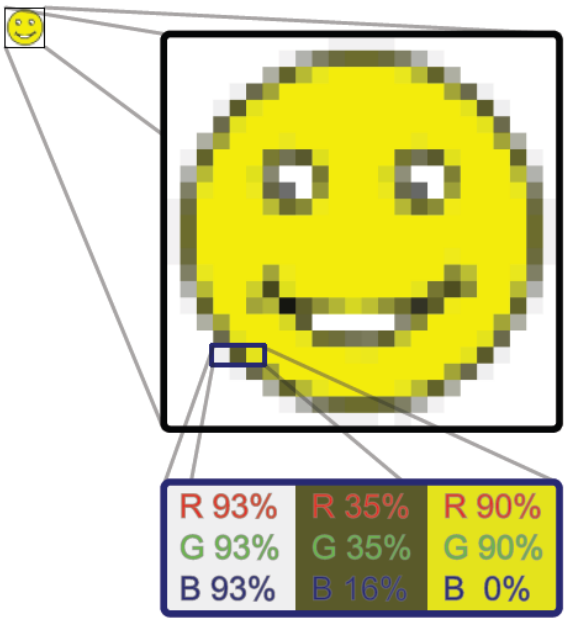


CODIFICA IMMAGINI A COLORI

GRAFICA BITMAP

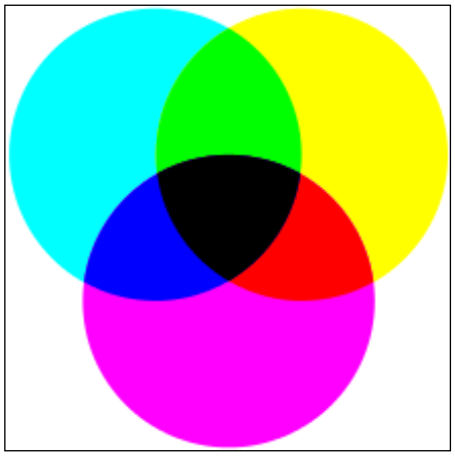
La rappresentazione delle immagini utilizzando la suddivisione in pixel si chiama grafica **bitmap (o raster).** Le immagini di questo tipo sono anche dette bitmap perché i pixel vengono memorizzati all'interno del file mediante una mappa di bit.

Ogni pixel è rappresentato da un insieme di bit che indicano: l'intensità di rosso (red), verde (*green*) e blu (*blu*) che deve essere associata al relativo punto dell'immagine così da poter essere rappresentato correttamente sullo schermo. La soluzione è quindi di dare per ogni pixel la percentuale di rosso, verde e blu necessaria. Questo sistema è chiamato **RGB** (da *red*, *green* e *blu*.)



I colori principali secondo tale codifica, utilizzata per esempio nei monitor, si ottengono per mezzo di combinazioni additive espresse a partire dai colori iniziali. Miscelando i tre colori in proporzioni diverse si ottengono tutti gli altri colori, sommando tutti e tre si ottiene il bianco, il nero è dato dall'assenza di colore.

Quando dobbiamo riprodurre i colori tramite la stampa si fa riferimento al principio della combinazione sottrattiva. I colori principali secondo tale codifica si ottengono per mezzo di combinazioni sottrattiva espresse a partire dai colori iniziali: azzurro (Cyan), violetto (Magenta) e giallo (Yellow). Miscelando i tre colori in proporzioni diverse si ottengono tutti gli altri colori, sommando tutti e tre si ottiene il nero, il bianco è dato dall'assenza di colore. Questo sistema è chiamato **CMYK**.

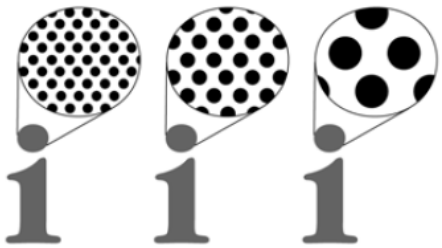


Ritornando al sistema RGB, tale sistema prevede che ogni colore (rosso, verde e blu) sia codificato mediante un numero intero utilizzando 8 bit per il rosso, 8 bit per il verde e 8 bit per il blu. Quindi ogni pixel per essere memorizzato richiede quindi 24 bit (3 byte). Siccome con 8 bit posso rappresentare al massimo 2^8= 256 combinazioni, il numero totale di colori distinti con questo sistema sarà 256x256x256= 16.777.216, cioè più di 16 milioni di colori.

Per poter definire e trattare un'immagine digitale occorre definire e comprendere alcuni termini tecnici:

- La **risoluzione** dell'immagine è il numero di pixel che la costituiscono, espressi in termini di larghezza x altezza (Es 1280x1024 pixel) o dpi (dot per inch, punti

per pollice- 1 inch = 2,54 cm). Un'immagine con una risoluzione di 300 DPI, ad esempio, apparirà più dettagliata (più definita) di una a 150 DPI, in quanto ogni pollice quadrato della sua superficie è composto da un numero di punti molto più elevato (ognuno dotato di specifiche caratteristiche di colore e luminosità) rispetto alla seconda immagine. Naturalmente, però, un'immagine a 300 DPI, essendo composta di più punti rispetto ad una a 150 DPI, necessita per essere "descritta" di un maggior numero di informazioni, e questo si ripercuote direttamente sul suo "peso" in termini di byte (o di "occupazione di spazio di memoria").



- La **profondità** cioè il numero di colori che il pixel può assumere. Dipende dal numero di bit che ho a disposizione per rappresentare il colore del pixel (Es 8 bit-> 256 colori, 16 bit-> 65536 colori, 24 bit-> 16 milioni di colori).

- Il **numero di byte** richiesti per memorizzare una immagine dipende dalla **risoluzione** e dal **numero di colori** che ogni pixel può assumere. Il numero di byte richiesti per memorizzare un'immagine è dato da:

numero di byte = risoluzione x n°colori

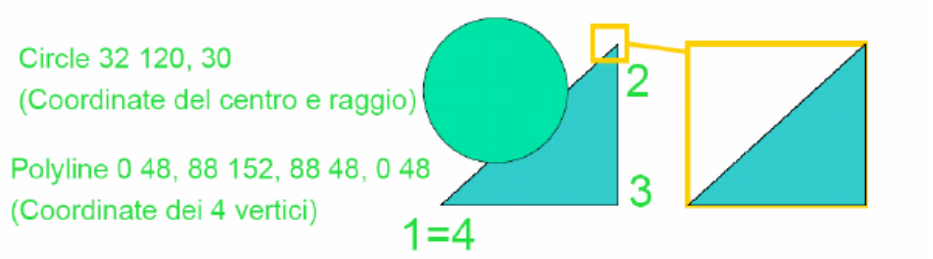
La rappresentazione dell'immagine, e quindi la sua qualità, sarà tanto più fedele all'originale quanto più aumenterà il numero di pixel (ovvero la risoluzione). Chiaramente più pixel userò tanto più grande sarà la rappresentazione di quell'immagine e quindi il file. Stesso discorso con i colori. Tanti più bit userò per definire i colori, tanto più potrò essere preciso ma tanto più sarà grande il file.
Ovviamente se i pixel aumentano in numero, diminuiscono di dimensione. Vediamo un esempio:

I formati BITMAP più comuni:

- Il formato bitmap (**.bmp**) è il formato standard di windows, semplice e con buona velocità di lettura e scrittura su disco, ma occupa molto spazio.
- Le immagini bitmap occupano parecchio spazio, esistono quindi delle tecniche di compressione che permettono di ridurne le dimensioni eliminando i pixel ripetitivi. Ad esempio, se k pixel lungo la stessa riga hanno lo stesso colore, si memorizza il colore una volta sola e il numero k. I file che usano queste tecniche di compressione hanno le estensioni GIF (**.gif**), JPEG (**.jpg**), TIFF (**.tif**).

GRAFICA VETTORIALE

Esiste anche un altro metodo di rappresentare le immagini chiamato **vettoriale**. In questo metodo ogni elemento geometrico viene specificato individualmente.
La codifica vettoriale è utilizzata se le immagini sono regolari in cui non si specificano le informazioni di colore di ogni singolo pixel ma si specificano gli elementi geometrici (punti, segmenti, poligoni,…) che le compongono. Questi elementi geometrici sono rappresentati da formule matematiche (un rettangolo è definito da due punti, un cerchio da un centro e un raggio, una curva da più punti e un'equazione). Sarà il processore ad essere incaricato di "tradurre" queste forme in informazioni interpretabili dalla scheda video. Le dimensioni dei file sono ridotte rispetto alle immagini bitmap. Sono immagini tipiche della progettazione meccanica (CAD).



Vantaggi/ svantaggi delle due grafiche:

Le immagini **bitmap** supportano in genere molti colori e trovano quindi applicazione in tutti quei casi in cui è richiesto un effetto pittorico o fotografico.
Purtroppo l'ingrandimento di un'immagine bitmap, porta inevitabilmente ad un decadimento della qualità dell'immagine stessa (vengono evidenziate in modo più marcato le differenze fra pixel e pixel). I file occupano molto spazio.

Un'immagine **vettoriale** ha il vantaggio di non risentire della "sgranatura" negli ingrandimenti. I file vettoriali possono infatti essere ingranditi, rimpiccioliti e deformati a piacimento senza avere nessuna perdita di definizione in quanto una volta noti i dati di base (ad esempio le coordinate del centro e raggio per un cerchio) le funzioni matematiche implementate nel programma permettono di riprodurre con precisione l'oggetto.

Non si presta per rappresentare immagini composte da continue variazioni di colore, quali ad esempio le fotografie. I file occupano poco spazio.

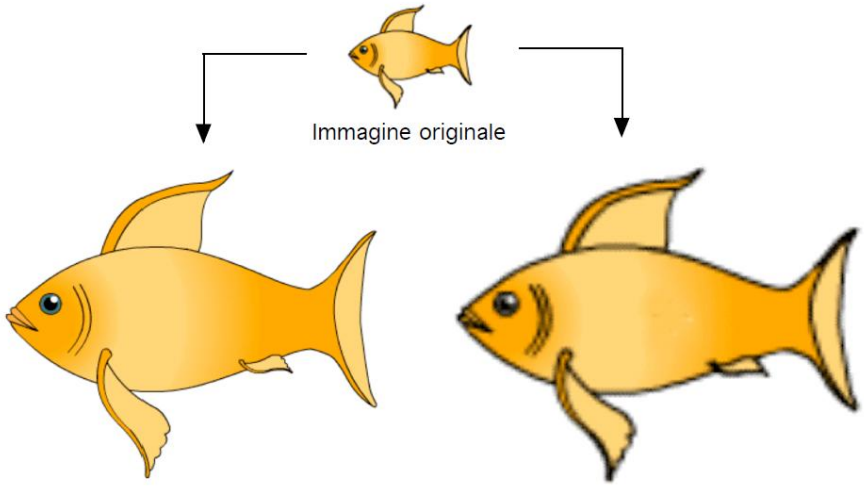


Immagine vettoriale Immagine Bitmap

Domande???

1. Quanti byte occupa un'immagine di 100x100 pixel in bianco e nero?
2. Quanti byte occupa un'immagine di 100x100 pixel a 256 colori?
3. Se un'immagine a 16,7 milioni di colori occupa 2400 byte, da quanti pixel sarà composta?

Risposte !!!

1. Conoscendo la risoluzione dell'immagine possiamo trovare il **numero di pixel** che la compongono: 100x100 = 10000 pixel. Inoltre, nel caso di immagini in bianco e nero basta **un solo bit** per codificare il colore di ogni pixel e quindi saranno necessari 10000 bit per memorizzare l'immagine. Per trovare il numero di byte basta fare 10000 / 8 = 1250 byte.
2. Rispetto all'esercizio precedente, in questo cambia lo spazio occupato da ciascun pixel. Sappiamo che l'immagine è a 256 colori. Per poter rappresentare 256 configurazioni diverse sono necessari 8 bit, ovvero 1 byte. L'immagine occuperà quindi 10000 x 1 byte = 10000 byte.
3. In questo caso le informazioni fornite dall'esercizio sono il numero colori e lo spazio occupato dall'immagine. Dal numero di colori ricaviamo lo spazio occupato da ciascun pixel, calcolando il valore N nell'espressione 2^N>=16,7 milioni. Il risultato è 24 bit, ovvero 3 byte. Se ogni pixel richiede 3 byte e l'immagine occupa 2400 byte, sarà composta da 2400 / 3 = 800 pixel.

Esercizi:

1. Da quanti pixel è composta una immagine b/n che occupa 2400 byte?

2. Spiega cos'è la risoluzione di un'immagine.
3. Un'immagine a 256 colori è formata da 400x400 pixel. Quanto spazio occupa?
4. Occupa più spazio un'immagine di 480x720 pixel a 128 colori oppure un'immagine di 240x360 pixel a 256 colori ?
5. Un'immagine di 300x400 pixel occupa 15.000 byte. L'immagine è a colori oppure in bianco e nero?
6. Hai ricevuto un messaggio di posta elettronica da un amico. Il messaggio contiene:
   - un testo di 300 caratteri scritto in ASCII esteso
   - un'immagine di 120x150 pixel con 1024 colori

   Quanti byte occupa il messaggio ?

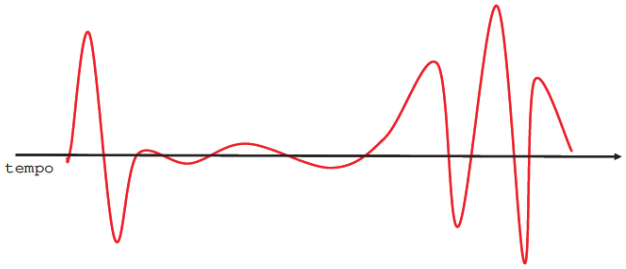
4 I video

Un filmato è una sequenza di immagini (dette fotogrammi o *frames*) che si susseguono rapidamente. Il movimento appare fluido e senza scatti se le immagini si susseguono alla velocità di almeno 25 fotogrammi al secondo.
Per codificare un filmato basta quindi digitalizzare i suoi fotogrammi, il che ci riconduce alle problematiche del paragrafo precedente.
In genere si tratta di sequenze compresse di immagini dove vengono registrate solo le variazioni tra un fotogramma e l'altro.
Vari formati (comprendente il sonoro):

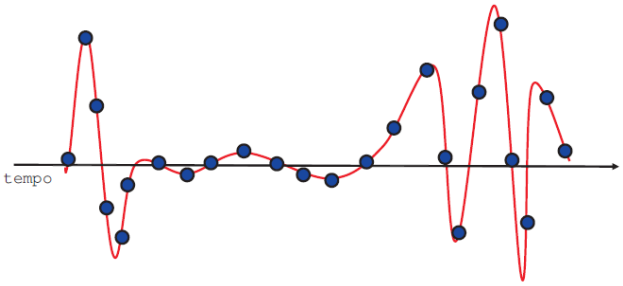
- .avi (Audio Video Interleave, Microsoft)
- .mov
- .mpeg (il più usato) compresso
- Quicktime (Apple)

5 I suoni

Il suono, come è noto, è una vibrazione. Fisicamente un suono è rappresentato come un'onda che descrive la variazione della intensità nel tempo (onda sonora). Graficamente un'onda sonora può essere rappresentata riportando sull'asse delle x il tempo e sull'asse delle y il valore dell'intensità:

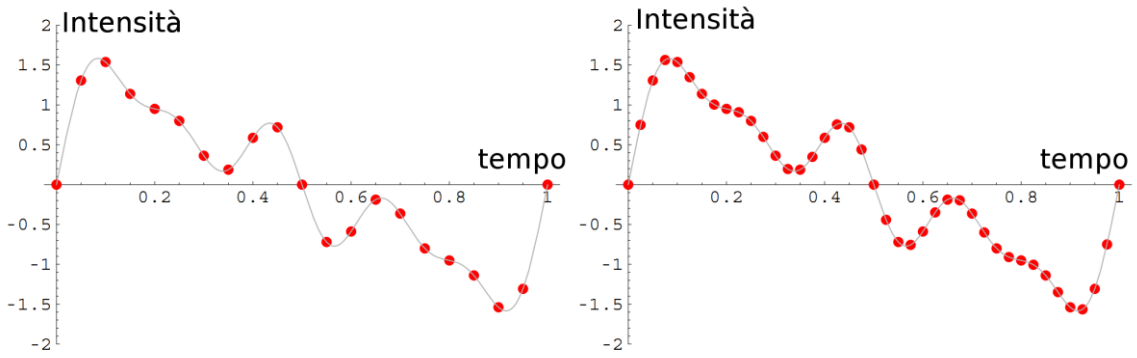


Per codificare i suoni si effettuano dei **campionamenti** sull'onda cioè si misura l'ampiezza dell'onda a intervalli di tempo costanti e si codificano in forma digitale le informazioni estratte da tali campionamenti:



La sequenza dei valori numerici così ottenuta può essere facilmente codificata.
Minore è l'intervallo che intercorre tra una misurazione e la successiva e maggiore sarà la qualità del suono. La **frequenza di campionamento** indica il numero di rilevazioni effettuate in un secondo. L'unità di misura della frequenza è l'Hertz (Hz) e in genere si utilizzano frequenze da 8 a 48 kHz (kilohertz = 1000 Hz). Se un suono è campionato a 200 Hz significa che in un secondo di suono vengono presi 200 campioni.

Esempio di suono preso da un microfono:



Si noti l'intensità in funzione del tempo. I pallini rossi rappresentano i momenti scelti per essere memorizzati (campionamento). A destra la frequenza di campionamento è più alta rispetto a sinistra e questo farà sì che il suono digitalizzato sarà più simile al suono originale.

Per codificare un suono si deve definire anche la risoluzione, cioè il livello di accuratezza per rappresentare il singolo campione, in genere da 8 a 32 bit. Più bit userò per rappresentare il campione tanto più rispondente al vero sarà il suono digitalizzato ma anche tanto più grande sarà lo spazio occupato. Utilizzando 8 bit si avranno 2^8= 256 dati per campione, con 16 bit se ne avranno 2^{16}= 65535, e così via.

Alcuni formati:
.mov, .avi, .ogg senza compressione

.mp3 , .wma formato compresso
.midi usato per 'elaborazione della musica al PC

Nei CD musicali: 44000 campionamenti al secondo, 16 bit per campione
Un minuto di audio con qualità CD musicale stereo occupa da 1MB a 10MB a seconda della codifica impiegata.

Domande ???

1. Quanto spazio occupa un suono della durata di 8 secondi campionato a 200 Hz, in cui ogni campione occupa 8 byte?
2. Un suono viene campionato a 512 Hz e occupa 1KB. Quanto occupa ogni campione? Quanti valori distinti si possono avere per i campioni?

Risposte !!!

1. La frequenza di campionamento dice quanti campioni di suono vengono memorizzati in un secondo. Avendo 8 secondi:
   8 x 200=1600 campioni
   Poiché ogni campione richiede 4 byte: 1600 x 4 = 6400 byte
2. Se il campionamento è a 512 Hz significa che in un secondo ci sono 512 campioni. Se l'occupazione totale è 1 KB (1024 byte) ogni campione occupa: 1024/ 512= 2 byte cioè 16 bit. Di conseguenza per ogni campione si possono avere 2^{16}= 65536 valori distinti.

Esercizi:

1. Quanto spazio occupa un suono della durata di 5 secondi campionato a 250 Hz, in cui ogni campione occupa 16 byte?
2. Un suono viene campionato a 200 Hz e occupa 2KB. Quanto occupa ogni campione?
3. Un suono viene campionato a 300 Hz e occupa 800 byte. Quanti valori distinti si possono avere per i campioni?