



La rappresentazione delle Informazioni

- Nella vita di tutti i giorni siamo abituati ad avere a che fare con vari tipi di informazioni, di natura e forma diversa, così come siamo abituati a diverse rappresentazioni della stessa informazione
 - Esempio: un testo può essere in forma stampata oppure la stessa informazione può essere registrata su un nastro
- Rappresentazioni *equivalenti* della stessa informazione e meccanismi di trasformazione tra differenti rappresentazioni
- La scelta della rappresentazione è in genere vincolata al tipo di utilizzo ed al tipo di operazioni che devono essere fatte sulle informazioni stesse



La rappresentazione delle informazioni

- I computer memorizzano ed elaborano le informazioni sotto forma di **bit (Binary Digit)**
- Un bit è l'unità di informazione base e può rappresentare due informazioni:
 - vero o falso
 - acceso o spento
 -
- Rappresentazione **binaria** (o **digitale**). Il linguaggio di base mediante il quale ogni informazione deve essere codificata è costituito da due soli simboli (**0** e **1**)

La rappresentazione delle informazioni

- Rappresentazione **binaria** (o **digitale**). Il linguaggio di base mediante il quale ogni informazione deve essere codificata è costituito da due soli **simboli (0 e 1)**
- NOTA BENE: i *simboli* che si usano per rappresentare il valore che un bit può assumere (0 e 1) NON devono essere confusi con:
 - i *numeri* 0 e 1 (la tecnica per rappresentare i numeri in formato digitale verrà studiata più avanti)
 - le *cifre del sistema decimale* 0 e 1 (ricordo che il concetto di cifra e quello di numero sono diversi: i numeri sono specificati come sequenza di cifre)
 - i *caratteri* 0 e 1 (i caratteri sono usati per la rappresentazione di testi, come vedremo più avanti)



La rappresentazione delle informazioni

Le ragioni di questa scelta sono prevalentemente di tipo tecnologico e i due simboli corrispondono a:

- due stati di polarizzazione di una sostanza magnetizzabile;
- due stati di carica elettrica di una sostanza;
- al passaggio/non passaggio di corrente attraverso un cavo conduttore;
- al passaggio/non passaggio di luce attraverso un cavo ottico



Codifica binaria

- Per poter rappresentare un numero maggiore di informazioni è necessario utilizzare sequenze di bit.

Per esempio, per rappresentare quattro informazioni diverse possiamo utilizzare due bit che ci permettono di ottenere quattro configurazioni distinte

00 01 10 11

- Il processo secondo cui si fa corrispondere ad un'informazione una configurazione di bit prende il nome di **codifica dell'informazione**

Esempio: un esame può avere quattro possibili esiti: ottimo, discreto, sufficiente, insufficiente

Codifica binaria

- Codifico

ottimo con *00*

discreto con *01*

sufficiente con *10*

insufficiente con *11*

- Con N bit si possono codificare 2^N informazioni differenti

N	Informazioni
2	4
3	8
4	16
5	32
6	64
7	128
8	256

Codifica binaria

- Se invece il mio problema è quello di dover rappresentare M informazioni differenti devo selezionare il numero di N bit in modo tale che

$$2^N \geq M$$

Esempio: per rappresentare 40 informazioni differenti devo utilizzare 6 bit perché $2^6 = 64$, 5 bit non sono sufficienti perché $2^5 = 32$

- Esiste una particolare aggregazione di bit che è costituita da 8 bit ($2^8 = 256$ informazioni) e prende il nome di **byte**



La rappresentazione delle Informazioni

- Noi vedremo le tecniche per rappresentare in formato digitale:
 - i caratteri
 - i numeri naturali
 - le immagini fisse
 - il suono
 - le immagini in movimento
- Parleremo di come si procede per ottenere una rappresentazione digitale (cioè, usando bit) mentre vedremo nella parte dedicata all'architettura di un calcolatore quali sono i dispositivi che materialmente realizzano le tecniche descritte.



La codifica dei caratteri di un testo

- L'obiettivo è quello di comunicare con il calcolatore usando il nostro linguaggio. Dobbiamo rappresentare le lettere dell'alfabeto
- L'insieme di simboli comunemente usati nell'alfabeto anglosassone, incluse le cifre numeriche, lettere maiuscole e minuscole, simboli di punteggiatura, parentesi e operatori aritmetici, può essere codificato usando 7 bit ($2^7 = 128$)
- Il metodo di codifica più diffuso tra i produttori di hardware e di software prende il nome di codice **ASCII** (**A**merican **S**tandard **C**ode for **I**nformation **I**nterchange)



Il codice ASCII

ASCII	Simb.	ASCII	Simb.	ASCII	Simb.
0000000	NUL	0001110	SO	0011100	FS
0000001	SOH	0001111	SI	0011101	GS
0000010	STX	0010000	DLE	0011110	RS
0000011	ETX	0010001	DC1	0011111	US
0000100	EOT	0010010	DC2	0100000	SP
0000101	ENQ	0010011	DC3	0100001	!
0000110	ACK	0010011	DC4	0100010	"
0000111	BEL	0010101	NAK	0100011	#
0001000	BS	0010110	SYN	0100100	\$
0001001	HT	0010111	ETB	0100101	%
0001010	NL	0011000	CAN	0100110	&
0001011	VT	0011001	EM	0100111	'
0001100	NP	0011010	SUB	0101000	(
0001101	CR	0011011	ESC	0101001)



Il codice ASCII

ASCII	Simb.	ASCII	Simb.	ASCII	Simb.
0101010	*	0111001	9	1000111	G
0101011	+	0111010	:	1001000	H
0101100	,	0111011	;	1001001	I
0101101	-	0111100	<	1001010	J
0101110	.	0111101	=	1001011	K
0101111	/	0111110	>	1001100	L
0110000	0	0111111	?	1001101	M
0110001	1	1000000	@	1001110	N
0110010	2	1000001	A	1001111	O
0110011	3	1000010	B	1010000	P
0110100	4	1000011	C	1010001	Q
0110101	5	1000100	D	1010010	R
0110110	6	1000101	E	1010011	S
0111000	8	1000110	F	1010100	T



Il codice ASCII

ASCII	Simb.	ASCII	Simb.	ASCII	Simb.
1010101	U	1100011	c	1110001	q
1010110	V	1100100	d	1110010	r
1010111	W	1100101	e	1110011	s
1011000	X	1100110	f	1110100	t
1011001	Y	1100111	g	1110101	u
1011010	Z	1101000	h	1110110	v
1011011	[1101001	i	1110111	w
1011100	\	1101010	j	1111000	x
1011101]	1101011	k	1111001	y
1011110	^	1101100	l	1111010	z
1011111	_	1101101	m	1111011	{
1100000	`	1101110	n	1111100	
1100001	a	1101111	o	1111101	}
1100010	b	1110000	p	1111110	~
1111111	DEL				



Il codice ASCII

Sebbene 7 bit siano sufficienti per codificare l'insieme di caratteri di uso comune, in genere il codice ASCII standard utilizza 8 bit, il primo dei quali è sempre 0

Codifica della parola **cane**

01100011	01100001	01101110	01100101
c	a	n	e

Tra i simboli speciali del codice ASCII vi è anche il simbolo spazio bianco "NUL"(codice 00000000), 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



Il codice ASCII

Consideriamo il problema inverso: data una sequenza di bit, il testo che essa codifica può essere ottenuto nel modo seguente:

- si divide la sequenza in gruppi di otto bit (un byte);
- si determina il carattere corrispondente ad ogni byte

Esempio:

01101001 01101100 00000000 01110000 01101111 00101110

01101001 01101100 00000000 01110000 01101111 00101110

i l P o .



La codifica dei caratteri di un testo

- 52 lettere alfabetiche maiuscole e minuscole
- 10 caratteri che denotano le cifre (0, 1, 2, ..., 9)
 - Nota bene: il codice ASCII e gli altri codici per la rappresentazione dei caratteri NON si usano per rappresentare i numeri (lo vedremo più avanti)
- Segni di punteggiatura (, . ; : ! " ? ' ^ \ ...)
- Simboli matematici (+, -, ×, ±, {, [, >, ...)
- Caratteri di alfabeti nazionali (à, è, ì, ò, ù, ç, ñ, ö, ...)
- Altri segni grafici (©, ←, ↑, ⊕, @, €, ...)



La codifica dei caratteri di un testo

- Codifiche standard:
 - **ASCII**, 8 bit per carattere, rappresenta 256 caratteri
 - **UNICODE**, 16 bit per carattere, (ASCII + caratteri etnici)
- Codifiche proprietarie:
 - **MSWindows**, 16 bit per carattere, simile ad UNICODE

La codifica dei numeri

- La rappresentazione dei numeri con il sistema decimale può essere utilizzata come spunto per definire un metodo di codifica dei numeri all'interno degli elaboratori: la sequenza 15 viene interpretato come: 1 decina + 5 unità
- In generale la sequenza $c_n c_{n-1} c_{n-2} \dots c_1 c_0$ (ogni " c_i " è una cifra compresa tra "0" e "9") viene interpretata come:

$$\begin{aligned} & c_0 \times 10^0 + && (c_0 \text{ unità}) \\ & c_1 \times 10^1 + && (c_1 \text{ decine}) \\ & c_2 \times 10^2 + && (c_2 \text{ centinaia}) \\ & \dots\dots \\ & c_{n-1} \times 10^{n-1} + \\ & c_n \times 10^n \end{aligned}$$

Un ripasso di aritmetica: la notazione posizionale

- La numerazione decimale utilizza una **notazione posizionale** basata sul numero 10 (**base**). La sequenza "234" rappresenta il numero $4 \times 10^0 + 3 \times 10^1 + 2 \times 10^2$
- La notazione posizionale può essere utilizzata in qualunque altro sistema di numerazione (con base diversa da 10)
- Nel sistema di numerazione binario i numeri vengono codificati utilizzando le due cifre "0" e "1"
- Nel sistema di numerazione ottale i numeri vengono codificati utilizzando le otto cifre "0", "1",, "7"
- Nel sistema di numerazione esadecimale i numeri vengono codificati utilizzando le sedici cifre "0", "1",, "8", "9", "A", "B", "C", "D", "E", "F"
 - La cifra "A" corrisponde a 10, la cifra "B" corrisponde a 11, la cifra "C" corrisponde a 12, la cifra "D" corrisponde a 13, la cifra "E" corrisponde a 14, la cifra "F" corrisponde a 15,

Un ripasso di aritmetica: La notazione posizionale

- In analogia con il caso decimale la sequenza $c_n c_{n-1} c_{n-2} \dots c_1 c_0$ (ogni " c_i " è la cifra "0" o la cifra "1") rappresenterà, in binario, il numero

$$c_0 \times 2^0 + c_1 \times 2^1 + \dots c_{n-1} \times 2^{n-1} + c_n \times 2^n$$

La sequenza "1011"denota il numero

$$1 \times 2^0 + 1 \times 2^1 + 0 \times 2^2 + 1 \times 2^3 = 11 \text{ (in base 10)}$$

- In analogia con il caso decimale la sequenza $c_n c_{n-1} c_{n-2} \dots c_1 c_0$ rappresenterà, in esadecimale, il numero

$$c_0 \times 16^0 + c_1 \times 16^1 + \dots c_{n-1} \times 16^{n-1} + c_n \times 16^n$$

La sequenza "1011"denota il numero

$$1 \times 16^0 + 1 \times 16^1 + 0 \times 16^2 + 1 \times 16^3 = 4113 \text{ (in base 10)}$$

Per evitare ambiguità si usa la notazione $1011_2 = 11_{10}$

Per evitare ambiguità si usa la notazione $1011_{16} = 4113_{10}$



Un ripasso di aritmetica: Rappresentazione decimale - limitazioni

- Consideriamo la base dieci: con tre cifre decimali si possono rappresentare i numeri compresi tra 0 e 999, il numero successivo (1000) richiede una quarta cifra di cui non disponiamo

In questo caso si dice che si ha un problema di **overflow**, ossia si eccede il numero di cifre destinato alla rappresentazione, e si genera un errore perché il numero non può essere gestito

Poiché il numero 999 può essere scritto come 10^3-1 (ossia 1000-1), possiamo enunciare la seguente regola:

con N cifre decimali si possono rappresentare
i numeri da 0 a 10^N-1

Un ripasso di aritmetica: Rappresentazione binaria - limitazioni

Consideriamo la base due: con tre cifre binarie si possono rappresentare i numeri compresi tra 0 e 2^3-1 (ossia 8-1), possiamo enunciare la seguente regola:

con N cifre binarie si possono rappresentare i numeri da 0 a 2^N-1

Esempio con N=3:

numero decimale	rappresentazione binaria
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Un ripasso di aritmetica: Rappresentazione esadecimale - limitazioni

Consideriamo la base sedici: con tre cifre esadecimali si possono rappresentare i numeri compresi tra 0 e 16^3-1 (ossia 4096-1).

con N cifre esadecimali si possono rappresentare i numeri da 0 a 16^N-1

Esempio con N=2:

numero decimale	rappresentazione esadecimale
0	00
1	01
.....
10	0A
11	0B
.....
15	0F
16	10
17	11
.....
30	1E
31	1F
32	20
.....

Un ripasso di aritmetica: Rappresentazione binaria - operazioni

- A queste rappresentazioni si possono applicare le operazioni aritmetiche:

$$0+0=0$$

$$1+0=1$$

$$0+1=1$$

$1+1=0$ con riporto di 1 ovvero 10

- $1+1$ in decimale è uguale a 2 ma siamo nella notazione binaria che ha solo due cifre, 0 e 1

Un ripasso di aritmetica: Rappresentazione binaria - operazioni

- A queste rappresentazioni si possono applicare le operazioni aritmetiche:

riporti

1

1 0 +

1 0 =

1 0 0

Un ripasso di aritmetica: Rappresentazione binaria - operazioni

- A queste rappresentazioni si possono applicare le operazioni aritmetiche:

riporti

1

1 1 +

1 0 =

1 0 1

Un ripasso di aritmetica: Rappresentazione binaria - operazioni

- A queste rappresentazioni si possono applicare le operazioni aritmetiche:

riporti

$$\begin{array}{r} 1 \quad 1 \\ \quad 1 \quad 1 \quad + \\ \quad 1 \quad 1 \quad = \\ 1 \quad 1 \quad 0 \end{array}$$

Un ripasso di aritmetica: Rappresentazione binaria - operazioni

- A queste rappresentazioni si possono applicare le operazioni aritmetiche:

riporti

$$\begin{array}{r} 1 \ 1 \ 1 \\ \ 1 \ 1 \ 1 \ + \\ \ 1 \ 1 \ = \\ 1 \ 0 \ 1 \ 0 \end{array}$$

Un ripasso di aritmetica: Conversione dalla base 10 alla base 2

- Dato un numero N rappresentato in base dieci, la sua rappresentazione in base due sarà del tipo:

$c_m c_{m-1} \dots c_1 c_0$ (le " c_i " sono cifre binarie)

- Come possiamo determinare queste cifre?
 - Si deve calcolare la divisione intera di N per 2: $N/2=N'$ con resto R'
 - R' è la cifra più a destra nella rappresentazione binaria di N , cioè $c_0 = R'$
 - Si divide $N'/2$ ottenendo $N'/2 = N''$ con resto R'' e si ha che $c_1 = R''$
 - Si ripete il procedimento fino a quando il risultato della divisione è uguale a 0

Un ripasso di aritmetica: Conversione dalla base 10 alla base 2

- Consideriamo ad esempio il numero 13_{10} e calcoliamo la sua rappresentazione in base due:

$$\begin{array}{rcll} 13/2 & = & 6 & \text{resto } 1 \\ 6/2 & = & 3 & \text{resto } 0 \\ 3/2 & = & 1 & \text{resto } 1 \\ 1/2 & = & 0 & \text{resto } 1 \end{array}$$

- Leggendo i resti dal basso verso l'alto, si ha che la rappresentazione binaria del numero 13_{10} è 1101_2

Un ripasso di aritmetica: Conversione dalla base 10 alla base 2

- Consideriamo ad esempio il numero 42_{10} e calcoliamo la sua rappresentazione in base due:

$$42/2 = 21 \quad \text{resto } 0$$

$$21/2 = 10 \quad \text{resto } 1$$

$$10/2 = 5 \quad \text{resto } 0$$

$$5/2 = 2 \quad \text{resto } 1$$

$$2/2 = 1 \quad \text{resto } 0$$

$$1/2 = 0 \quad \text{resto } 1$$

- Leggendo i resti dal basso verso l'alto, si ha che la rappresentazione binaria del numero 42_{10} è 101010_2

Un ripasso di aritmetica: Conversione dalla base 10 alla base 2

- Consideriamo ad esempio il numero 345_{10} e calcoliamo la sua rappresentazione in base due:

$345/2$	$=$	172	resto	1
$172/2$	$=$	86	resto	0
$86/2$	$=$	43	resto	0
$43/2$	$=$	21	resto	1
$21/2$	$=$	10	resto	1
$10/2$	$=$	5	resto	0
$5/2$	$=$	2	resto	1
$2/2$	$=$	1	resto	0
$1/2$	$=$	0	resto	1

- Leggendo i resti dal basso verso l'alto, si ha che la rappresentazione binaria del numero 345_{10} è 101011001_2

Un ripasso di aritmetica: Conversione dalla base 10 alla base 16

- Dato un numero N rappresentato in base dieci, la sua rappresentazione in base sedici sarà del tipo:

$c_m c_{m-1} \dots c_1 c_0$ (le " c_i " sono cifre esadecimale)

- Come possiamo determinare queste cifre?
 - Si deve calcolare la divisione intera di N per 16: $N/16=N'$ con resto R'
 - R' è la cifra più a destra nella rappresentazione esadecimale di N , cioè $c_0 = R'$
 - Si divide $N'/16$ ottenendo $N'/16 = N''$ con resto R'' e si ha che $c_1 = R''$
 - Si ripete il procedimento fino a quando il risultato della divisione è uguale a 0

Un ripasso di aritmetica: Conversione dalla base 10 alla base 16

- Consideriamo ad esempio il numero 345_{10} e calcoliamo la sua rappresentazione in base sedici:

$$\begin{array}{rcl} 345/16 = & 21 & \text{resto } 9 \\ 21/16 = & 1 & \text{resto } 5 \\ 1/16 = & 0 & \text{resto } 1 \end{array}$$

- Leggendo i resti dal basso verso l'alto, si ha che la rappresentazione esadecimale del numero 345_{10} è 159_{16}

Un ripasso di aritmetica: Conversione dalla base 2 alla base 10

- Sia $c_m c_{m-1} \dots c_1 c_0$ un numero rappresentato in base 2, per trovare la rappresentazione decimale di questo numero dobbiamo considerare le potenze successive della base 2

$$c_0 \times 2^0 + c_1 \times 2^1 + \dots + c_{m-1} \times 2^{m-1} + c_m \times 2^m = N$$

- Esempio: 101011001_2

$$1 \times 2^0 + 0 \times 2^1 + 0 \times 2^2 + 1 \times 2^3 + 1 \times 2^4 + 0 \times 2^5 + 1 \times 2^6 + 0 \times 2^7 + 1 \times 2^8 = 1 + 8 + 16 + 64 + 256 = 345$$

Un ripasso di aritmetica: Conversione dalla base 16 alla base 10

- Sia $c_m c_{m-1} \dots c_1 c_0$ un numero rappresentato in base 16, per trovare la rappresentazione decimale di questo numero dobbiamo considerare le potenze successive della base 16

$$c_0 \times 16^0 + c_1 \times 16^1 + \dots + c_{m-1} \times 16^{m-1} + c_m \times 16^m = N$$

- Esempio: 159_{16}

$$9 \times 16^0 + 5 \times 16^1 + 1 \times 16^2 = 9 + 80 + 256 = 345$$



Rappresentazione digitale dei numeri

- FINE DEL RIPASSO DI ARITMETICA!!!!
- Come si ottiene la rappresentazione usando bit?
 - si calcola la rappresentazione in base 2 del numero
 - si associa ad ogni cifra binaria della rappresentazione in base 2 un bit
 - se la cifra binaria vale 0 si associa un bit che vale 0
 - se la cifra binaria vale 1 si associa un bit che vale 1

Rappresentazione digitale dei numeri


- Dato che i numeri si possono anche inserire in un calcolatore digitando **caratteri** della tastiera possiamo rappresentare un numero come sequenza di caratteri che rappresentano le cifre che lo compongono.
- Ad esempio:
ASCII: 37 00110011 00110111 (2 byte)
 3 7
BINARIA: 37 00100101 (1 byte)
- Il primo modo è usato per le comunicazioni con l'esterno (input/output:) dato che i numeri si inseriscono digitando **caratteri** della tastiera mentre il secondo modo è usato all'interno del calcolatore per fare i calcoli aritmetici

Rappresentazione digitale dei numeri

- A questo fine non è possibile usare direttamente le codifiche ASCII:

Esempio:	Numero	ASCII
	3 +	00110011
	2 =	00110010
	----	-----
	e	01100101

- Esistono dei programmi di conversione che trasformano i numeri da una codifica all'altra



Rappresentazione di numeri negativi e dei numeri reali

- In realtà, una semplice codifica binaria come quella discussa fino ad ora non è sufficiente, per due motivi:
 - numeri negativi
 - numeri con la virgola
- Per questi numeri vengono utilizzate delle rappresentazioni differenti



La codifica delle immagini

Lettere e numeri non costituiscono le uniche informazioni utilizzate dagli elaboratori ma si stanno diffondendo sempre di più applicazioni che utilizzano ed elaborano anche altri tipi di informazione: **diagrammi, immagini, suoni, filmati**. Spesso in questi casi si parla di applicazioni di tipo

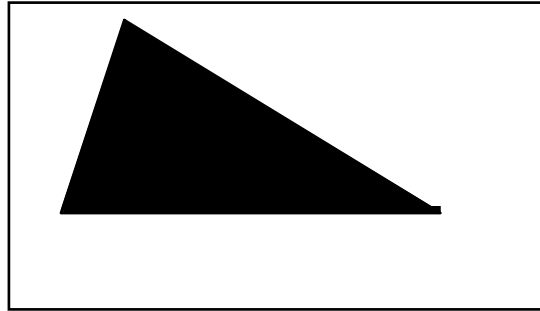
MULTIMEDIALE



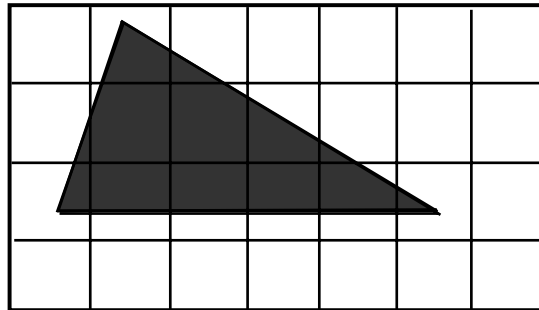
La codifica delle immagini

- Vediamo dapprima il caso delle immagini
- Esistono numerose tecniche che vengono utilizzate per la memorizzazione digitale e l'elaborazione di un'immagine
- Consideriamo un'immagine in bianco e nero, senza ombreggiature o livelli di chiaroscuro

La codifica delle immagini



Suddividiamo l'immagine mediante una griglia formata da righe orizzontali e verticali a distanza costante



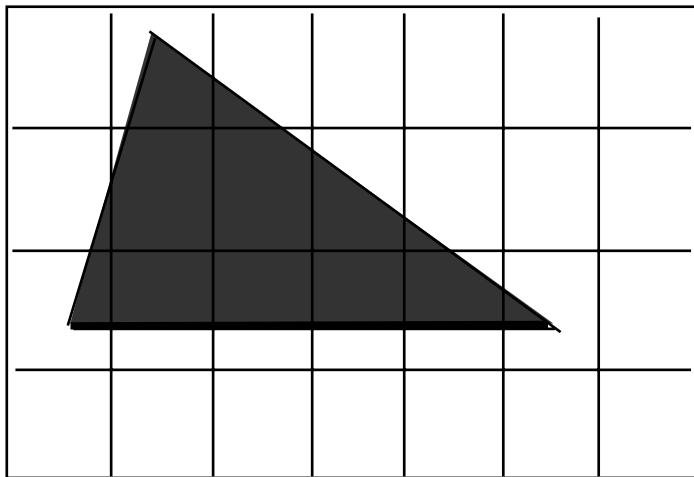


La codifica delle immagini

- Ogni quadratino derivante da tale suddivisione prende il nome di **pixel** (**picture element**) e può essere codificato in binario secondo la seguente convenzione:
 - 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)

La codifica delle immagini

- Poiché una sequenza di bit è lineare, è necessario definire delle convenzioni per ordinare la griglia dei pixel in una sequenza. Assumiamo che i pixel siano ordinati dal basso verso l'alto e da sinistra verso destra



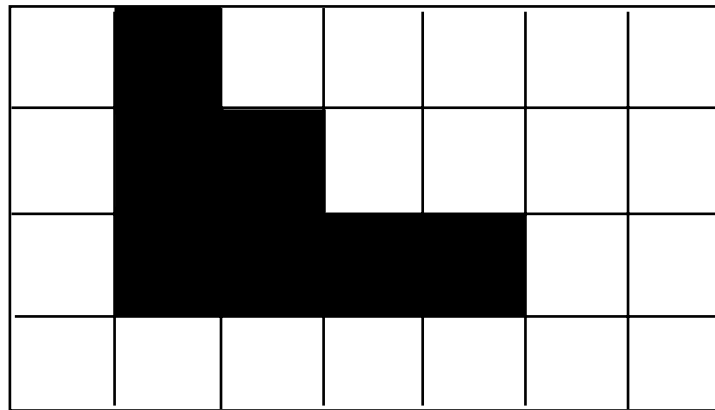
0 ₂₂	1 ₂₃	0 ₂₄	0 ₂₅	0 ₂₆	0 ₂₇	0 ₂₈
0 ₁₅	1 ₁₆	1 ₁₇	0 ₁₈	0 ₁₉	0 ₂₀	0 ₂₁
0 ₈	1 ₉	1 ₁₀	1 ₁₁	1 ₁₂	0 ₁₃	0 ₁₄
0 ₁	0 ₂	0 ₃	0 ₄	0 ₅	0 ₆	0 ₇

Con questa convenzione la rappresentazione della figura sarà data dalla sequenza di bit

0000000011110001100000100000

La codifica delle immagini

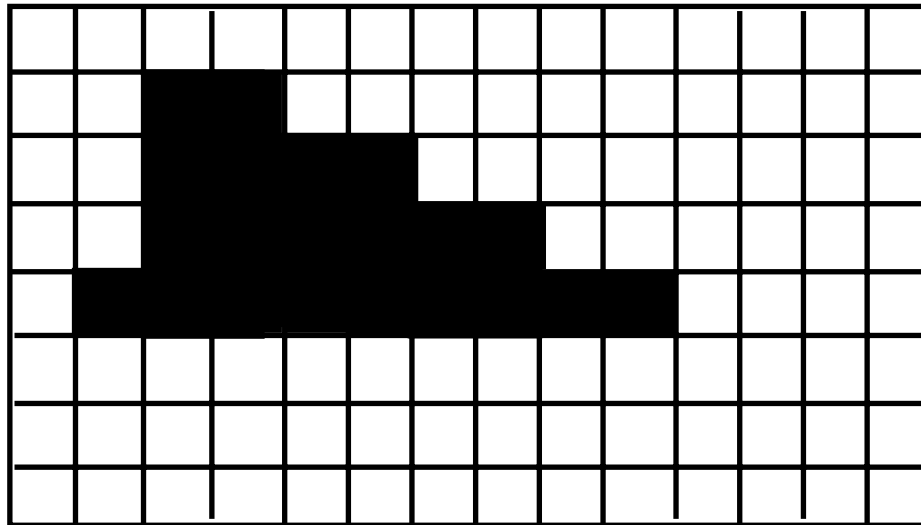
- Non sempre il contorno della figura coincide con le linee della griglia. Quella che si ottiene nella codifica è un'approssimazione della figura originaria
- Se riconvertiamo la stringa **0000000011110001100000100000** in immagine otteniamo



LA DIGITALIZZAZIONE COMPORTA PERDITA
DI QUALITÀ

La codifica delle immagini

- La rappresentazione sarà più fedele all'aumentare del numero di pixel, ossia al diminuire delle dimensioni dei quadratini della griglia in cui è suddivisa l'immagine



AUMENTIAMO LA QUALITÀ AUMENTANDO LA
GRANULARITÀ DELL'IMMAGINE



La codifica delle immagini

- Assegnando un bit ad ogni pixel è possibile codificare solo immagini senza livelli di chiaroscuro
- Le immagini in bianco e nero hanno delle sfumature (diversi livelli di intensità di grigio)
- Per codificare le immagini con diversi livelli di grigio si usa la stessa tecnica: per ogni pixel si stabilisce il livello medio di grigio cui viene assegnata convenzionalmente una rappresentazione binaria
- Per memorizzare un pixel non è più sufficiente un solo bit. Ad esempio, se utilizziamo quattro bit possiamo rappresentare $2^4=16$ livelli di grigio, mentre con otto bit ne possiamo distinguere $2^8=256$, ecc.



La codifica delle immagini

- Analogamente possiamo codificare le immagini a colori. In questo caso si tratta di individuare un certo numero di sfumature, gradazioni di colore differenti e di codificare ognuna mediante un'opportuna sequenza di bit
- Qualsiasi colore può essere rappresentato dalla composizione del rosso, del verde e del blu.
- Quindi, invece che rappresentare alcune sfumature di tanti colori diversi, possiamo rappresentare molte sfumature dei tre colori primari: dalla combinazione di essi otteniamo tanti altri colori.




La codifica delle immagini

- Codifica **RGB** (*Red, Green, Blu* - Rosso, Verde, Blu ovvero i tre colori primari).
- Ogni pixel viene rappresentato con una combinazione dei tre colori
- Per ogni colore primario si usa un certo numero di bit per rappresentarne la gradazione (la "quantità")
- Ad esempio, utilizzando 8 bit per colore primario, otteniamo 256 diverse gradazioni, ovvero $256 \times 256 \times 256 = 16777216$ colori diversi. In questo caso un pixel richiede tre byte di informazione



La codifica delle immagini

Le sequenze di bit relative ad ogni colore primario si possono interpretare come la rappresentazione di una quantità (la gradazione, la sfumatura) quindi si possono esprimere in base decimale:

Ad esempio, se il colore di un pixel fosse  sarebbe espresso con la seguente sequenza di bit: **100010111101001011011000** allora potremmo, più comodamente, scrivere **139 210 216**.

Avendo 8 bit a disposizione per rappresentare la gradazione di un colore fondamentale, tutti e tre i numeri sono compresi tra 0 e 255

Spesso, per comodità di scrittura, tale codifica è espressa in base esadecimale. In questo modo, lo stesso colore dell'esempio sarebbe espresso nel modo seguente: **8B D2 D8** ($8B_{16} = 139_{10}$, e così via...).

100010111101001011011000 equivale a **139 210 216** ed a **8B D2 D8**



La codifica delle immagini

- La rappresentazione di un'immagine mediante la codifica dei pixel, viene chiamata codifica **bitmap** o **raster**
- La **risoluzione** dell'immagine è il numero di pixel che la costituiscono, espressi in termini di *larghezza x altezza*. Ovviamente, aumentando il numero di pixel a disposizione, migliora la qualità dell'immagine.
- La **profondità** dell'immagine è invece il numero di bit che servono per rappresentare un singolo pixel dell'immagine.
- Il numero di bit richiesti per memorizzare un'immagine dipende dalla risoluzione e dalla profondità

numero di bit per immagine = risoluzione x profondità



La codifica delle immagini

- Per distinguere 16777216 colori sono necessari 24 bit per la codifica di ciascun pixel: la codifica di un'immagine formata da 640X480 pixel richiederà 7.372.800 bit (921.600 byte)
- Esistono delle tecniche di compressione delle informazione che consentono di ridurre drasticamente lo spazio occupato dalle immagini
- **codifiche di compressione:** le più famose sono la *CompuServe Graphic Interface (GIF)* e la *Joint Photographic Experts Group (JPEG)*. I file che usano tali codifiche riportano rispettivamente le estensioni .gif e .jpg (o anche .jpeg)



La codifica delle immagini

- Tali formati (detti anche **codec**: *compression/decompression*), usano un sistema per comprimere l'informazione prima di memorizzarla e per decomprimerla prima di visualizzarla.
- entrambi i formati tendono ad eliminare i pixel ripetitivi,
- Entrambe le soluzioni sono *compressioni con perdita* di informazione. Tale perdita non può essere recuperata in alcun modo. La codifica JPEG consente di manipolare tale fattore di compressione.



La codifica delle immagini

- Un'immagine può occupare molto spazio anche se non tutti i 16777216 colori sono CONTEMPORANEAMENTE usati
- Si può usare un sottoinsieme dei colori
- Si considera una tavolozza (*palette*) di colori (codificati con il sistema RGB) da codificare
- Ad esempio, sono comuni palette a 256 colori, ovvero, con profondità dell'immagine a 8 bit
- La palette viene memorizzata insieme al resto dei dati dell'immagine
- Nell'esempio precedente, sarebbero necessari 8 bit per la codifica di ciascun pixel: la codifica richiederà 2457600 bit (307200 byte) per l'immagine più $256 \times 3 = 768$ byte per la palette.

La codifica delle immagini

Palette

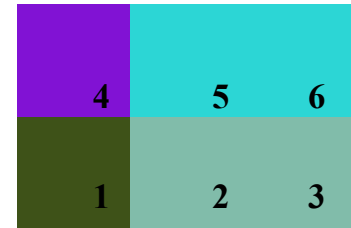
RGB dei colori

81	12	D4
44	D6	D5
3E	52	18
1B	BC	AA

indice nella
tabella (palette)

10 11 11 00 01 01
pixel₁ pixel₂ pixel₃ pixel₄ pixel₅ pixel₆

immagine 3x2



Rappresentazione con palette:

$$\underbrace{24 \times 4}_{\text{palette}} + \underbrace{2 \times 6}_{\text{pixel}} = \underline{108} \text{ bit}$$

Rappresentazione RGB:

$$24 \times 6 = \underline{144} \text{ bit}$$

3E 52 18 1B BC AA 1B BC AA 81 12 D4 44 D6 D5 44 D6 D5
pixel 1 pixel 2 pixel 3 pixel 4 pixel 5 pixel 6

La codifica delle immagini

Palette

81	00	D4
44	00	D5
3E	00	18
1B	00	AA

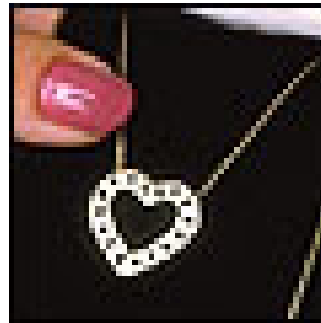
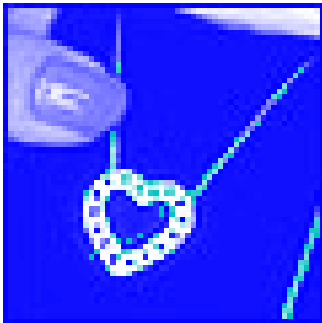
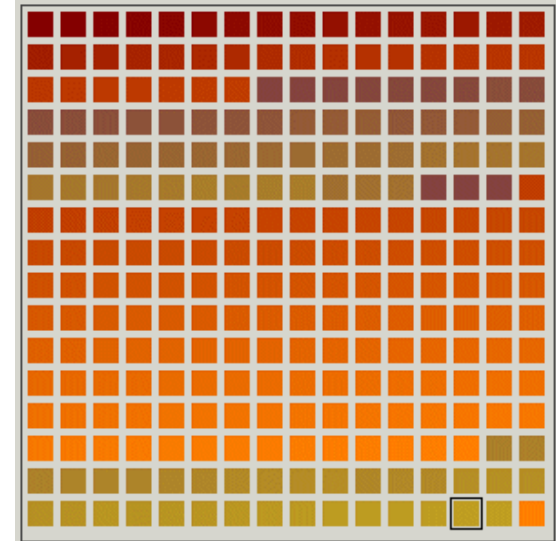
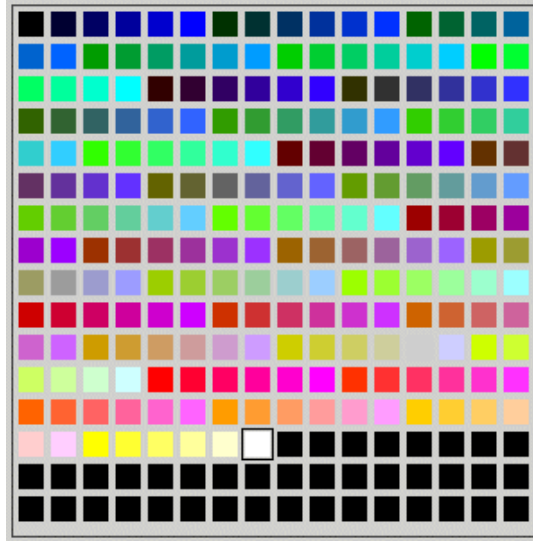
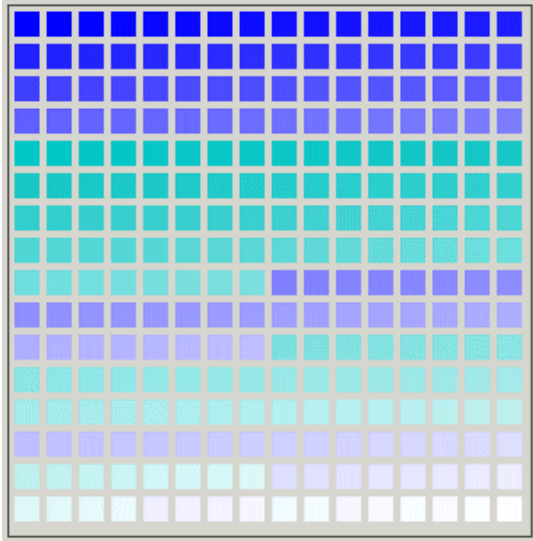
indice nella
tabella (palette)

10 11 11 00 01 01
pixel₁ pixel₂ pixel₃ pixel₄ pixel₅ pixel₆

immagine 3x2

4	5	6
1	2	3

La codifica delle immagini





La codifica delle immagini

- Quando abbiamo bisogno di colori che non sono presenti in questa tavolozza?
- possiamo sostituire il colore mancante con quello più simile presente nella palette,

oppure

- cambiare palette. In questo modo, cambia l'associazione tra sequenze di bit e colori.
- Di conseguenza, quando si usa un programma di elaborazione dell'immagine (es. paint shop pro, photoshop, etc.), bisogna specificare quale palette si sta usando.



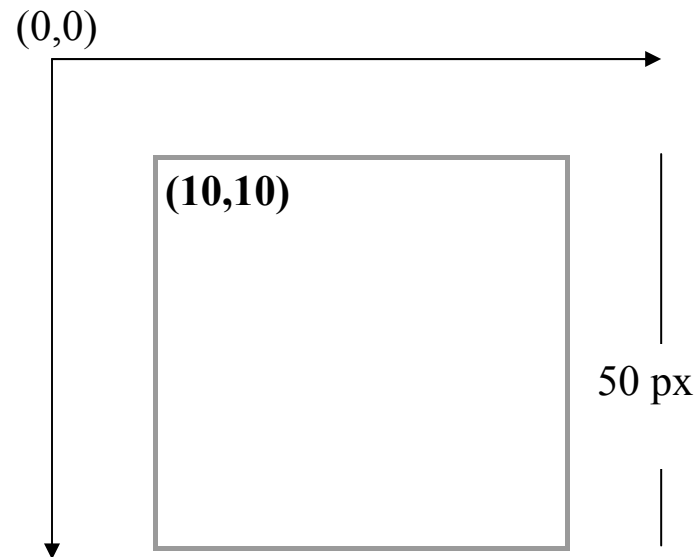
La codifica delle immagini

- **Grafica vettoriale:** *descrizione di elementi geometrici primitivi, i quali vengono specificati individualmente. Non si descrivono i pixel singolarmente.*
- si definiscono le curve e tutti gli elementi geometrici che compongono l'immagine memorizzando solo le loro coordinate
- un programma che gestisce immagini in grafica vettoriale dovrà prima leggere le coordinate e riprodurre pixel per pixel le curve
- formato testuale (si crea e modifica con un editor di testi)
- meno occupazione di memoria + elaborazione per la riproduzione

La codifica dell'immagine

- individuare un punto di riferimento (che può essere il vertice in alto a sinistra del quadrato)
- lunghezza del lato
- origine degli assi cartesiani

Rectangle(10,10, 50, 50)





La codifica delle immagini

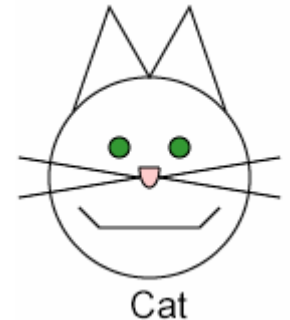
- **Osservazione:** è chiaro che in un file in formato vettoriale bisognerebbe memorizzare opportunamente anche la risoluzione dell'immagine, lo spessore in pixel delle linee, i colori delle linee e dei riempimenti delle figure, etc. Un approfondimento della codifica vettoriale in tal senso è al di fuori degli scopi di questo corso.

La codifica delle immagini: formato SVG

- SVG (Scalable Vector Graphics) è un linguaggio di grafica vettoriale bidimensionale

```
<svg width="140" height="170">
<title>Cat</title>
<desc>Stick Figure of a Cat</desc>

<circle cx="70" cy="95" r="50" style="stroke: black; fill: none;"/>
<circle cx="55" cy="80" r="5" stroke="black" fill="#339933"/>
<circle cx="85" cy="80" r="5" stroke="black" fill="#339933"/>
<g id="whiskers">
  <line x1="75" y1="95" x2="135" y2="85" style="stroke: black;"/>
  <line x1="75" y1="95" x2="135" y2="105" style="stroke: black;"/>
</g>
<use xlink:href="#whiskers" transform="scale(-1 1) translate(-140 0)"/>
<!-- ears -->
<polyline points="108 62, 90 10, 70 45, 50 10, 32, 62"
  style="stroke: black; fill: none;" />
<!-- mouth -->
<polyline points="35 110, 45 120, 95 120, 105, 110"
  style="stroke: black; fill: none;" />
<!-- nose -->
<path d="M 75 90 L 65 90 A 5 10 0 0 0 75 90"
  style="stroke: black; fill: #ffcccc"/>
<text x="60" y="165" style="font-family: sans-serif; font-size: 14pt;
  stroke: none; fill: black;">Cat</text>
</svg>
```





La codifica delle immagini

- Immagini complesse od irregolari: codifica *raster* o *bitmap*
 - Codifiche standard: **GIF, JPEG, BMP**
- Immagini regolari: codifica *vettoriale*
 - Codifiche standard (proprietarie): **CGM, DWG, DXF**
 - **Macromedia FLASH**
- Codifiche *ibride (raster/vettoriale)*:
 - Codifiche standard (proprietarie): **Postscript, PDF** (Portable Document Format)