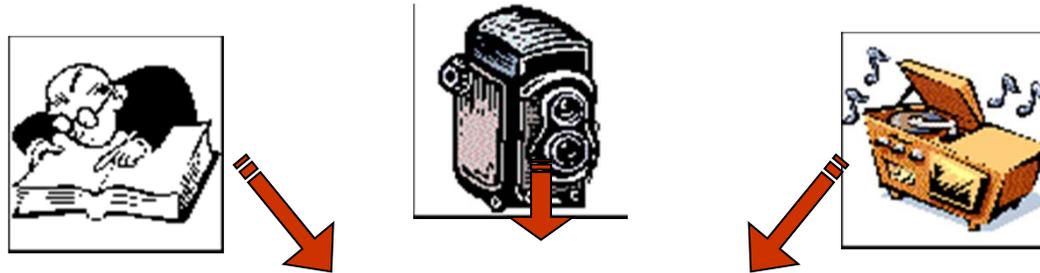


---

*Sistemi di numerazione:  
binario, ottale ed esadecimale*

# Codifica binaria dell'Informazione



01101010

Bit	0/1 (si/no)
Byte	00010010 (8 bit)
Kilobyte	$2^{10} = 1024$ byte
Megabyte	$2^{20} \sim 1.000.000$ byte
Gigabyte	$2^{30} \sim 1.000.000.000$ byte

Concetto di codifica e decodifica delle Informazioni

# *Sistemi di numerazione*

---

- Si chiama **sistema di numerazione** l'insieme di un numero finito di **simboli** e delle **regole** che assegnano uno ed un solo significato ad ogni scrittura formata coi simboli stessi.
- I simboli di un sistema di numerazione prendono il nome di **cifre**.
- Il sistema di numerazione più noto è il sistema **decimale** che si avvale dei dieci simboli (o cifre) 0,1,2,3,4,5,6,7,8,9

# *Sistemi di numerazione*

---

- Una qualunque sequenza di queste cifre permette di rappresentare un ben determinato numero nel sistema di numerazione decimale.
- I moderni sistemi di numerazione sono **posizionali**, cioè tutti i simboli (o cifre) vengono ordinati in modo che ognuno abbia peso maggiore rispetto al simbolo (cifra) precedente: il *valore* del numero rappresentato dipende dalle posizioni relative alle cifre che lo compongono.

# *Sistemi di numerazione*

---

- Il numero delle cifre di cui si avvale un sistema di numerazione prende il nome di **base**.
- In ogni numero decimale (**in base dieci**) la cifra più a destra ha il peso minore (**cifra meno significativa**), quella più a sinistra il peso maggiore (**cifra più significativa**).
- Spostando una cifra di una posizione verso sinistra si moltiplica il suo *valore* per la base (**per dieci nel caso del sistema decimale**).

# *Il sistema di numerazione posizionale decimale*

---

Nella numerazione posizionale ogni cifra del numero assume un valore in funzione della “*posizione*”:

**221** in notazione compatta, cioè

$$2 \times 100 + 2 \times 10 + 1 \times 1$$

ovvero, con la notazione esplicita

$$2 \times 10^2 + 2 \times 10^1 + 1 \times 10^0$$

# *Sistema posizionale*

---

- Ogni numero si esprime come la somma dei prodotti di ciascuna cifra per la base elevata all'esponente che rappresenta la *posizione* della cifra:

$$221 = 2 \times 10^2 + 2 \times 10^1 + 1 \times 10^0$$

# *Sistema posizionale (cont.)*

---

- La *notazione posizionale* può essere usata con qualunque base creando così differenti sistemi di numerazione.
  - Per ogni base di numerazione si utilizza un numero di cifre uguale alla base.
- In informatica si utilizza prevalentemente la numerazione:
  - binaria,
  - ottale,
  - esadecimale.
- Il sistema di numerazione romano non è posizionale:
  - Ad esempio, XIII vs. CXII.

# *Sistema di numerazione decimale*

---

➤ La numerazione *decimale* utilizza una notazione posizionale basata su 10 cifre (da 0 a 9) e sulle potenze di 10.

- Il numero 234 può essere rappresentato esplicitamente come:

$$2 \times 10^2 + 3 \times 10^1 + 4 \times 10^0$$

# *Sistema di numerazione binario*

---

➤ Il sistema di numerazione *binario* utilizza una notazione posizionale basata su 2 cifre (0 e 1) e sulle potenze di 2.

- Il numero 1001 può essere corrisponde al numero decimale:

$$\begin{aligned} 1001_2 &= 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 \\ &= 9_{10} \end{aligned}$$

# *Sistema di numerazione ottale*

---

➤ Il sistema di numerazione *ottale* utilizza una notazione posizionale basata su 8 cifre (da 0 a 7) e sulle potenze di 8.

- Il numero 534 corrisponde al numero decimale:

$$534_8 = 5 \times 8^2 + 3 \times 8^1 + 4 \times 8^0 = 348_{10}$$

# *Sistema di numerazione esadecimale*

---

➤ La numerazione *esadecimale* utilizza una notazione posizionale basata su 16 cifre (da 0 a 9 ed i caratteri A, B, C, D, E, F) e sulle potenze di 16.

- Il numero  $\mathbf{B7FC}_{16}$  corrisponde al numero decimale :

$$\begin{aligned} (11) \times 16^3 + 7 \times 16^2 + (15) \times 16^1 + (12) \times 16^0 \\ = 47100_{10} \end{aligned}$$

## *Conversione da base n a base 10*

---

- Per convertire un numero da una qualunque base alla base 10 è sufficiente rappresentarlo esplicitamente:

$$1101_2 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 13_{10}$$

$$710_8 = 7 \times 8^2 + 1 \times 8^1 + 0 \times 8^0 = 456_{10}$$

$$A51_{16} = (10) \times 16^2 + 5 \times 16^1 + 1 \times 16^0 = 2641_{10}$$

## *Conversione da base 10 a base $n$*

---

- Per convertire un numero ad una base  $n$  qualsiasi occorre trovare tutti i resti delle successive divisioni del numero per la base  $n$ .
- Come esempio si vuole trovare il valore binario del numero **210**

# Conversione da base 10 a base 2

---

210	2	resto	0
105	2		1
52	2		0
26	2		0
13	2		1
6	2		0
3	2		1
1	2		1



➤ Leggendo la sequenza dei resti dal basso verso l'alto, si ottiene il numero:

**11010010<sub>2</sub>**

# *Verifica di correttezza*

---

- Per una verifica di correttezza basta riconvertire il risultato alla base 10:

$$\begin{aligned} 11010010_2 &= 1 \times 2^7 + 1 \times 2^6 + 0 \times 2^5 + \\ &\quad 1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + \\ &\quad 1 \times 2^1 + 0 \times 2^0 = 210_{10} \end{aligned}$$

# *Costruzione dei numeri binari*

---

- Per costruire la successione dei numeri binari si può seguire il seguente schema:

0	0	0	0	=	0
0	0	0	1	=	1
0	0	1	0	=	2
0	0	1	1	=	3
0	1	0	0	=	4
0	1	0	1	=	5
0	1	1	0	=	6
0	1	1	1	=	7

# *I primi 32 numeri binari*

---

0	0	0	0	0	0	0	0	=	0
0	0	0	0	0	0	0	1	=	1
0	0	0	0	0	0	1	0	=	2
0	0	0	0	0	0	1	1	=	3
0	0	0	0	0	1	0	0	=	4
0	0	0	0	0	1	0	1	=	5
0	0	0	0	0	1	1	0	=	6
0	0	0	0	0	1	1	1	=	7

0	0	0	0	1	0	0	0	=	8
0	0	0	0	1	0	0	1	=	9
0	0	0	0	1	0	1	0	=	10
0	0	0	0	1	0	1	1	=	11
0	0	0	0	1	1	0	0	=	12
0	0	0	0	1	1	0	1	=	13
0	0	0	0	1	1	1	0	=	14
0	0	0	0	1	1	1	1	=	15

# *I primi 32 numeri binari (cont.)*

---

0	0	0	1	0	0	0	0	=	16
0	0	0	1	0	0	0	1	=	17
0	0	0	1	0	0	1	0	=	18
0	0	0	1	0	0	1	1	=	19
0	0	0	1	0	1	0	0	=	20
0	0	0	1	0	1	0	1	=	21
0	0	0	1	0	1	1	0	=	22
0	0	0	1	0	1	1	1	=	23

0	0	0	1	1	0	0	0	=	24
0	0	0	1	1	0	0	1	=	25
0	0	0	1	1	0	1	0	=	26
0	0	0	1	1	0	1	1	=	27
0	0	0	1	1	1	0	0	=	28
0	0	0	1	1	1	0	1	=	29
0	0	0	1	1	1	1	0	=	30
0	0	0	1	1	1	1	1	=	31

- 
- Con  $n$  bit si possono codificare  $2^n$  numeri decimali.
  - Ad esempio con 4 bit si possono codificare i seguenti  $16=2^4$  numeri:  $0,1,2,\dots,14,15=2^4-1$
  - In generale con  $n$  bit si possono codificare tutti i numeri decimali compresi tra 0 e  $2^n-1$

# *Operazioni binarie*

---

10110101+

1000110 =

---

11111011

00110011+

00111000 =

---

01101011

## *Operazioni binarie (cont.)*

---

$$\begin{array}{r} 1101 \text{ x} \\ 11 = \\ \hline 1101 \\ 1101 \\ \hline 100111 \end{array}$$

$$\begin{array}{r} 10011 \text{ x} \\ 10 = \\ \hline 00000 \\ 10011 \\ \hline 100110 \end{array}$$

# *Esercizi*

---

Eseguire le seguenti operazioni direttamente in binario, convertire in decimale e verificare il risultato:

- $110000 + 1001010$ ;
- $1001010 + 1111111$ ;
- $100110 \times 111100$ ;
- $001001 \times 111$ .

---

*Rappresentazione delle informazioni :*

*La codifica dei testi*

---

# *Rappresentazione dei caratteri*

---

➤ Cos'è un carattere ?

- Si tratta di un simbolo, in qualche modo astratto.

– Per esempio una “A” è la rappresentazione grafica convenzionale (detta anche *glifo*) del concetto di carattere “a maiuscola”.

➤ Dobbiamo trovare una “*convenzione*” con cui realizzare una rappresentazione comprensibile al computer dei caratteri.

# Codifica binaria

---

➤ Per poter rappresentare le informazioni è necessario utilizzare *sequenze* di bit.

- Utilizzando due bit si possono rappresentare quattro informazioni diverse:

00    01    10    11

➤ Il processo che fa corrispondere ad una informazione una configurazione di bit prende il nome di *codifica dell'informazione*.

# Sequenze di bit

---

Numero di bit nella sequenza	Informazioni rappresentabili
2	4
3	8
4	16
5	32
6	64
7	128
8	256

In generale,  
con  $n$  bit si  
possono  
rappresentare  
 $2^n$  differenti  
informazioni.

# *Il byte*

---

➤ Un gruppo di 8 bit viene denominato Byte.

- Unità di misura della capacità di memoria.

➤ Si utilizzano i multipli del Byte:

- Kilo      KB       $2^{10}$       ~ un migliaio      (1024)
- Mega     MB       $2^{20}$       ~ un milione      (1024x1024)
- Giga     GB       $2^{30}$       ~ un miliardo     (1MBx1024)
- Tera     TB       $2^{40}$       ~ mille miliardi (1GBx1024)

# *I caratteri utilizzati nella comunicazione scritta*

---

- 52 lettere alfabetiche maiuscole e minuscole
- 10 cifre (0, 1, 2, ..., 9)
- Segni di punteggiatura (, . ; : ! ” ? ’ ^ \ ...)
- Segni matematici (+, -, ×, ±, {, [, >, ...)
- Caratteri nazionali (à, è, ì, ò, ù, ç, ñ, ö, ...)
- Altri segni grafici (©, ←, ↑, □, @, € ...)
- In totale 220 caratteri circa.

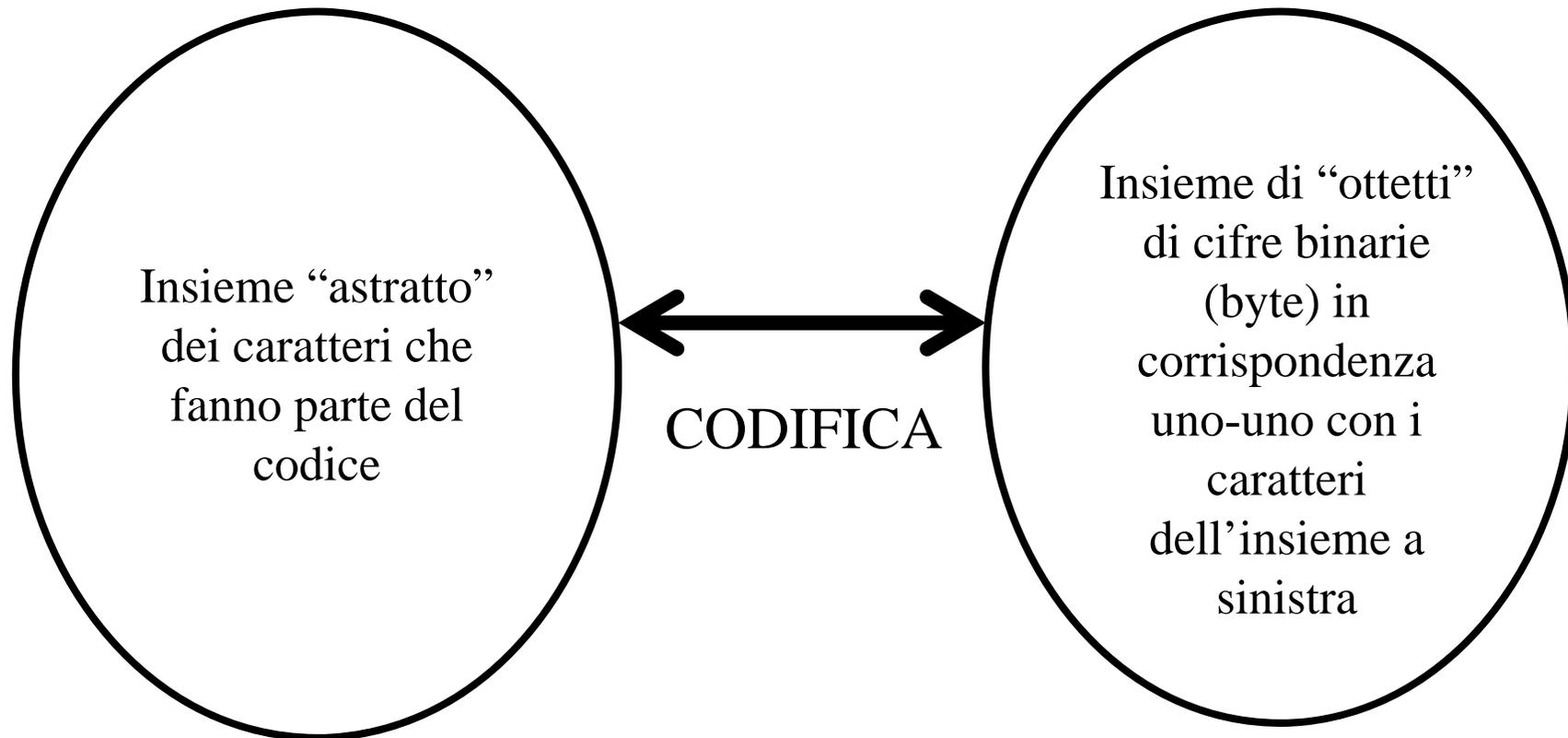
# *Codice*

---

- Si pone quindi la necessità di codificare in numeri binari almeno 220 caratteri.
- La sequenza di bit necessaria a rappresentare 220 simboli deve essere composta da 8 bit e prende il nome di **CODICE.**

# *La fase di codifica*

---



# Il codice *ASCII*

.....			0011 0000	48	0
0100 0001	65	A	0011 0001	49	1
0100 0010	66	B	0011 0010	50	2
0100 0011	67	C	0011 0011	51	3
.....			.....		
0101 1000	88	X	0011 1010	58	:
0101 1001	89	Y	0011 1011	59	;
0101 1010	90	Z	0011 1100	60	<
.....			0011 1101	61	=
0110 0001	97	a	.....		
0110 0010	98	b	1010 0100	164	ñ
0110 0011	99	c	1000 0111	135	ç

**A**merican  
**S**tandard **C**ode  
for **I**nformation  
**I**nterchange

## *Il codice ASCII (cont.)*

---

- I caratteri ASCII da 0 a 127:
  - I primi 32 (numerati da 0 fino a 31) sono “caratteri di controllo” *non stampabili*,
  - I successivi 95 simboli (numerati da 32 fino a 126) sono caratteri *stampabili*,
  - Il 128-esimo simbolo è ancora un “carattere di controllo” *non stampabile*.

Tabella ASCII in notazione binaria: la concatenazione del ‘nibble’ di riga e di quello di colonna dà il codice ASCII in binario.

---

	0000 0	0001 1	0010 2	0011 3	0100 4	0101 5	0110 6	0111 7	1000 8	1001 9	1010 10	1011 11	1100 12	1101 13	1110 14	1111 15
0000 0																
0001 1																
0010 2	spazio	!	“	#	\$	%	&	‘	(	)	*	+	,	-	.	/
0011 3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
0100 4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
0101 5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
0110 6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
0111 7	p	q	r	s	t	u	v	w	x	y	z	{		}	~	

# *Sequenze di caratteri ASCII*

---

Dividendo la sequenza in gruppi di byte è possibile risalire ai singoli caratteri:

01101001 01101100 00100000 01010000 01001111 00101110

01101001 01101100 00100000 01010000 01001111 00101110

i

l

P

O

.

# *Esempi di sequenze*

---

➤ “*Computer*” in ASCII diventa:

- C=67=01000011,                      o=111=01101111,  
  m=109=001101101,                p=112=01110000,  
  u=117=01110101,                t=116=01110100,  
  e=101=01100101,                r=114=01110010.

- 01000011- 01101111- 01101101- 11100000-01110101-01110100-01100101- 01110010

➤ Esercizio :

- *Scrivere “ASCII” in decimale ed in binario.*

# *Numeri e codice ASCII*

---

- 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

- Con questo tipo di rappresentazione *non* è possibile effettuare operazioni aritmetiche.

# *Rappresentazione di dati alfabetici*

---

## Codifiche standard:

- **ASCII**, 8 bit per carattere, rappresenta 256 caratteri.
- **UNICODE**, 16 bit per carattere
  - ASCII e caratteri etnici ( $2^{16} = 65.536$  simboli).

## ➤ Codifiche proprietarie:

- **MSWindows**, 16 bit per carattere
  - simile ad UNICODE.

# Dieci dita e qualche tasto...

---

- La mia tastiera ha meno di cento tasti.
  - Come ottenere tutti i simboli desiderati?
    - Usando combinazioni di tasti.
      - ❖ Per esempio, <Shift><tasto> dà la versione maiuscola.
    - Digitando la combinazione :
      - ❖ <Alt><codice ASCII in notazione decimale>
        - In questo caso bisogna usare il *tastierino numerico* per inserire il codice !

---

*Rappresentazione delle informazioni :*  
*Codifica di dati multimediali*

---

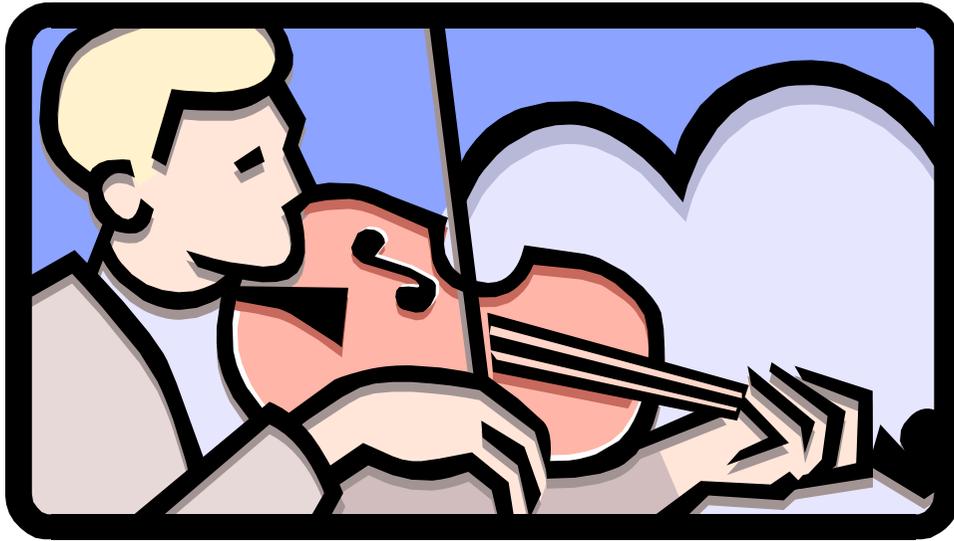
# *Ma il mondo non è tutto “scritto” !*

---

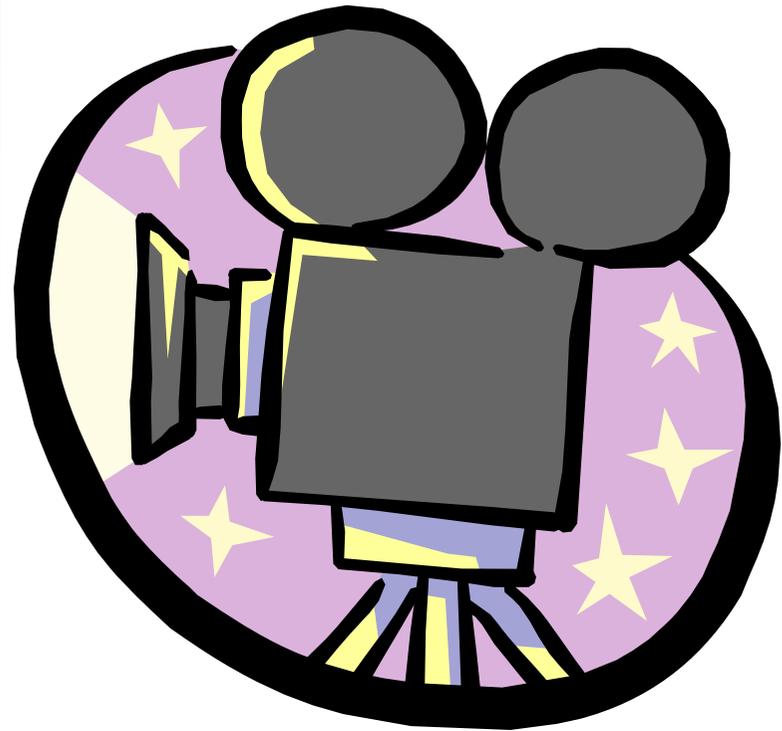
- I caratteri alfanumerici non costituiscono le uniche informazioni utilizzate dagli elaboratori.
  - Le applicazioni *multimediali* utilizzano ed elaborano informazioni contenenti:
    - immagini,
    - suoni,
    - filmati.

# *La codifica dei dati multimediali*

---



L'informazione per gli umani ha un carattere analogico....



# *In termini tecnici si tratta di* *“SEGNALI”*

---

➤ Un segnale analogico si può:

- **CAMPIONARE**

- Per un suono: misurare l'intensità ogni centesimo di secondo.
- Per una immagine : misurare i colori ogni millimetro quadrato.

- **RAPPRESENTARE** con un numero.

# *La codifica delle immagini (1)*

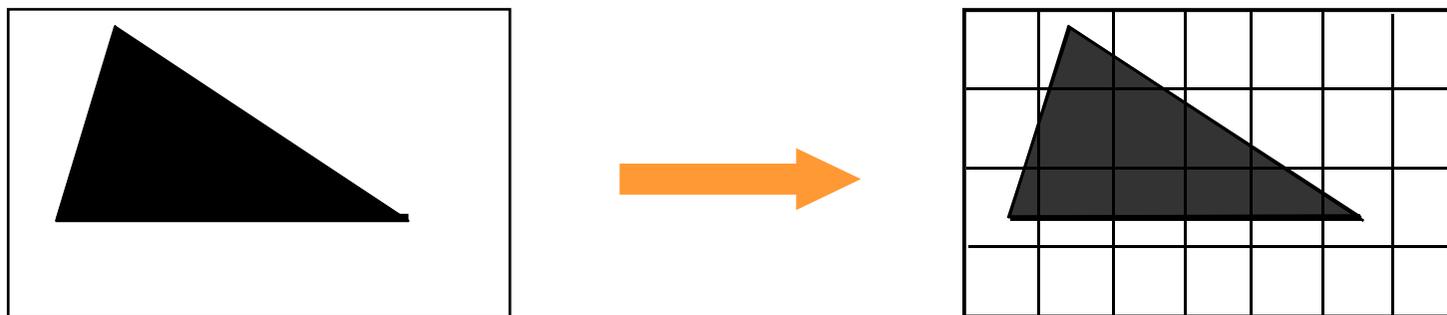
---

- Esistono numerose tecniche che vengono utilizzate per la memorizzazione e l'elaborazione di un'immagine.
- Consideriamo un'immagine in bianco e nero, senza ombreggiature o livelli di chiaroscuro.

## *La codifica delle immagini (2)*

---

L'immagine viene suddivisa mediante una griglia formata da righe orizzontali e verticali a distanza costante.



# *La codifica delle immagini (3)*

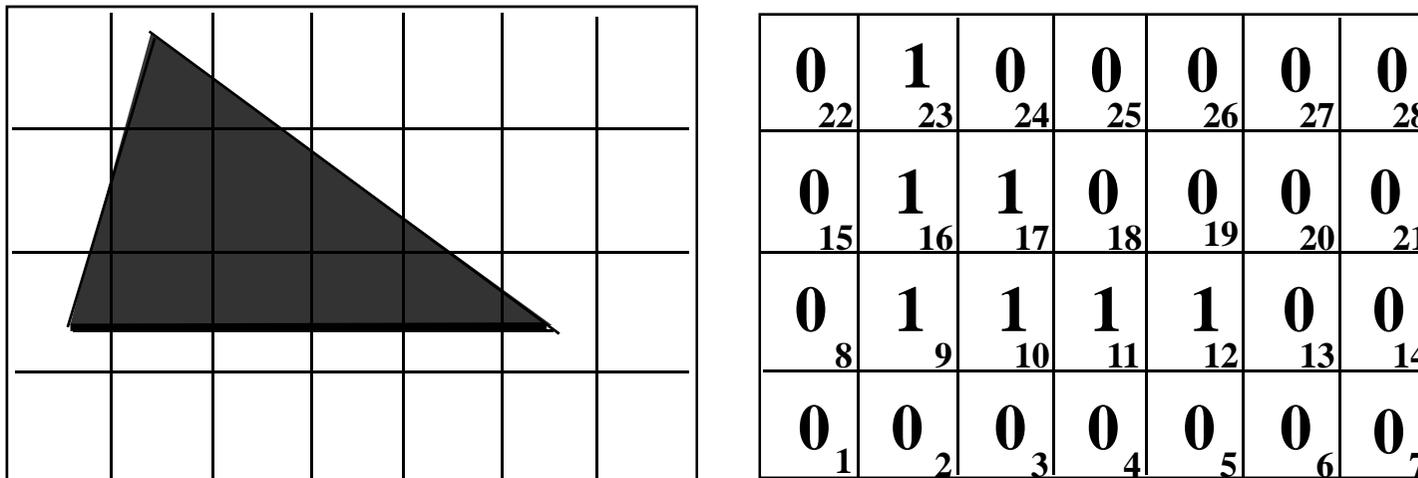
---

- Ogni quadrato prende il nome di **pixel** (**picture element**) e viene codificato in binario secondo la seguente convenzione:
  - Il simbolo 0 viene utilizzato per la codifica di un pixel corrispondente ad un quadrato in cui il bianco è predominante.
  - Il simbolo 1 viene utilizzato per la codifica di un pixel corrispondente ad un quadratino in cui il nero è predominante.

# La codifica delle immagini (4)

---

- Per convenzione la griglia dei pixel è ordinata dal basso verso l'alto e da sinistra verso destra.



La figura sarà rappresentata dalla stringa binaria:

**000000 0111100 0110000 0100000**

# *La codifica delle immagini (5)*

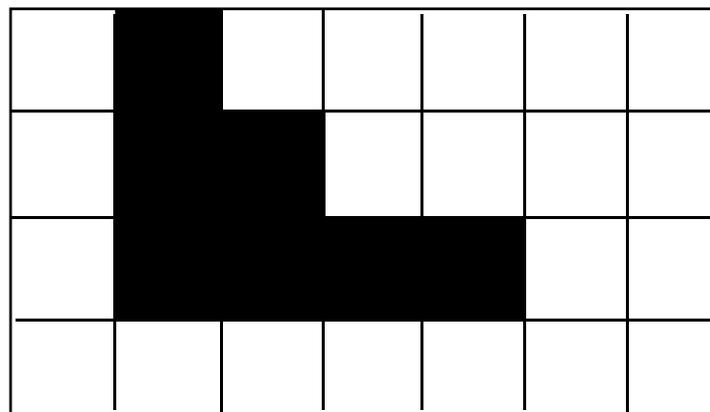
---

➤ Dato che il contorno della figura non sempre coincide con la griglia si ottiene un'approssimazione della figura originaria.

- Riconvertendo la stringa:

– 0000000 0111100 0110000 0100000

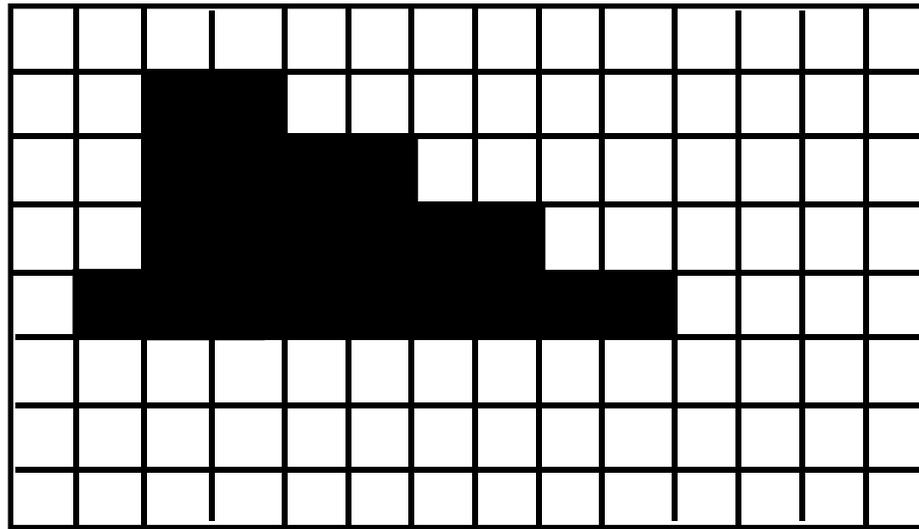
si avrà:



# *La codifica delle immagini (6)*

---

- La rappresentazione sarà più fedele all'aumentare del numero di pixel (ovvero la risoluzione).



- La *dimensione* dell'immagine è espressa come numero di pixel nel formato *righe x colonne*.

# *Gradazioni di grigio*

---

- Per codificare immagini con diversi livelli di grigio si usa una rappresentazione binaria: ad ogni livello di grigio corrisponde una sequenza di bit.
  - Ad esempio, utilizzando quattro bit si possono rappresentare  $2^4 = 16$  livelli di grigio, mentre con otto bit ne possiamo distinguere  $2^8 = 256$ .

# *Esempi di toni di grigio*

---



1 bit  
2 toni



2 bit  
4 toni



8 bit  
256 toni

# *L'uso del colore*

---

- Il colore viene generato dalla composizione di tre *colori primari*: Red, Green, Blue (video RGB)
- Ad ogni colore primario si associa una possibile sfumatura o *gradazione* mediante un'opportuna sequenza di bit.
  - Utilizzando 2 bit per ogni colore primario si possono ottenere 4 gradazioni per ognuno di essi, cioè 64 colori.
    - Un pixel richiede quindi un byte circa (6 bit) in questo caso.

# *L'uso del colore (cont.)*

---

- Utilizzando 4 bit per ogni colore primario si ottengono 16 gradazioni per ognuno di essi, cioè 4096 colori differenti.
  - Un pixel richiede quindi circa due byte (12 bit) di informazione.
- Utilizzando 8 bit per ogni colore primario si ottengono 256 gradazioni per ognuno di essi, cioè 16,8 milioni di colori circa.
  - Un pixel richiede quindi tre byte di informazione.

# Risoluzione

---

- Il numero di pixel per unità di area indica la “*risoluzione*” con cui si è campionata l’immagine.
- Il numero di pixel presenti sul video (*colonne x righe*) prende il nome di **risoluzione dello schermo**.
  - Risoluzioni tipiche sono: 640x480, 800x600, 1024x768, 2048 x 1536
  - La dimensione dell’immagine sarà:

	16 bit	32 bit
640x480	600 KB	~ 1,2 MB
800x600	~ 938 KB	~ 1,9 MB
1024x768	~ 1,6 MB	3MB
2048 x 1536	?	?

# Formati delle immagini

---

- Codifica *raster* o *bitmap*.
  - Ogni punto rappresenta un'informazione indipendente.
    - GIF,
    - JPEG,
    - BMP.
- Codifiche *ibride* (*raster/vettoriale*).
  - Ogni elemento geometrico primitivo viene specificato individualmente.
    - Postscript,
    - PDF (Portable Document Format).

# Codifica di filmati video

---

- Un filmato è una sequenza di immagini statiche (dette **fotogrammi** o *frames*).
  - Codifica normale (dei singoli frames).
    - 3 minuti di video con 24 frames/sec,
      - ❖ minimo 16 frame/s per non percepire i singoli fotogrammi.
    - Risoluzione singolo frame : 200x100, 16 bit/pixel.
    - Memoria necessaria:  $(3*60*24) (200*100*2) \sim 165$  MB.
  - Codifica differenziale.
    - È inefficiente codificare completamente ogni frame.
      - ❖ Alcuni frames si codificano interamente, altri solo nelle parti che differiscono da quelli adiacenti.

# *Formati video*

---

## ➤ MPEG (Moving Picture Experts Group)

- Costituisce uno standard.
- Molto efficiente ma complesso.

## ➤ QuickTime

- Proposto da Apple.
- Meno efficiente ma più semplice.

## ➤ Indeo - AVI

- Proposto da Intel, usato da MicroSoft.
- È il più inefficiente.

# *Codifica di suoni*

---

- Il segnale acustico viene digitalizzato.
  - Dimensioni medie:
    - un minuto di audio con qualità CD musicale stereo occupa da 1MB a 10MB a seconda della codifica impiegata.
- Codifiche standard:
  - WAV (MS-Windows),
  - MIDI
  - MP3

# *Formati sonori*

---

## ➤ MIDI:

- Codifica le note e gli strumenti che devono eseguirle.
- Solo musica, non voce.
- Richiede un sintetizzatore o “campioni” per la riproduzione (non utilizzabile “direttamente”).
- Molto efficiente.

## ➤ MP3:

- MPEG - layer 3: variante MPEG per i suoni.
- Grande diffusione.
- Molto efficiente.

# *Ma quanto spazio ci vuole?*

---

- I bit costano!
- Trasmettere i bit costa!
- Gli uomini però hanno linguaggi “ridondanti”
  - Esempio : Dmn è Psqa e nn c sn lzn d infomtca
    - È un messaggio poco chiaro ma decodificabile...  
E occupa meno caratteri (e meno bit) di “Domenica è Pasqua e non ci sono lezioni di informatica”.
- IDEA: compressione!!!!

# *Compressione*

---

➤ Concetto base:

- AAABBBC → 3A3BC → AAABBBC
- ABCABC → 1A1B1C1A1B1C → ABCABC

Compressione Lossy o Lossless