Informatica Grafica

Corso di Laurea in Ingegneria Edile – Architettura

Elaborazione di documenti elettronici

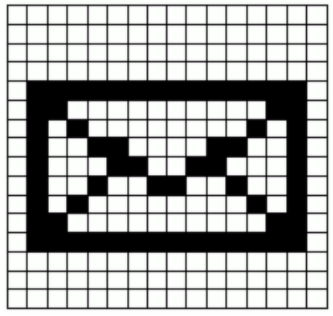
Michele Lombardi

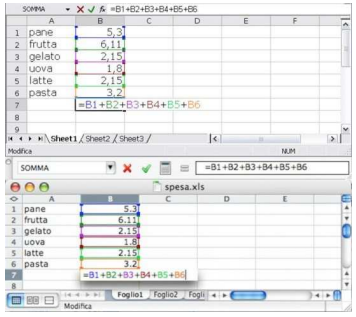
su materiale originario di Paolo Torroni

Dipartimento di Elettronica, Informatica e Sistemistica (DEIS)
Università degli Studi di Bologna

Anno Accademico 2010/2011

Elaborazione di documenti elettronici





- Codifica di **dati semplici** sul calcolatore
  - Numeri
  - Colori
  - Testo
- Rappresentazione di **immagini** ed **elementi grafici**
- Rappresentazione di **testo formattato**
- **Documenti di testo elettronici**
  - Struttura e presentazione
  - linguaggio HTML (pagine web)
  - linguaggio XML (accenni)

Cosa si intende per documento elettronico?

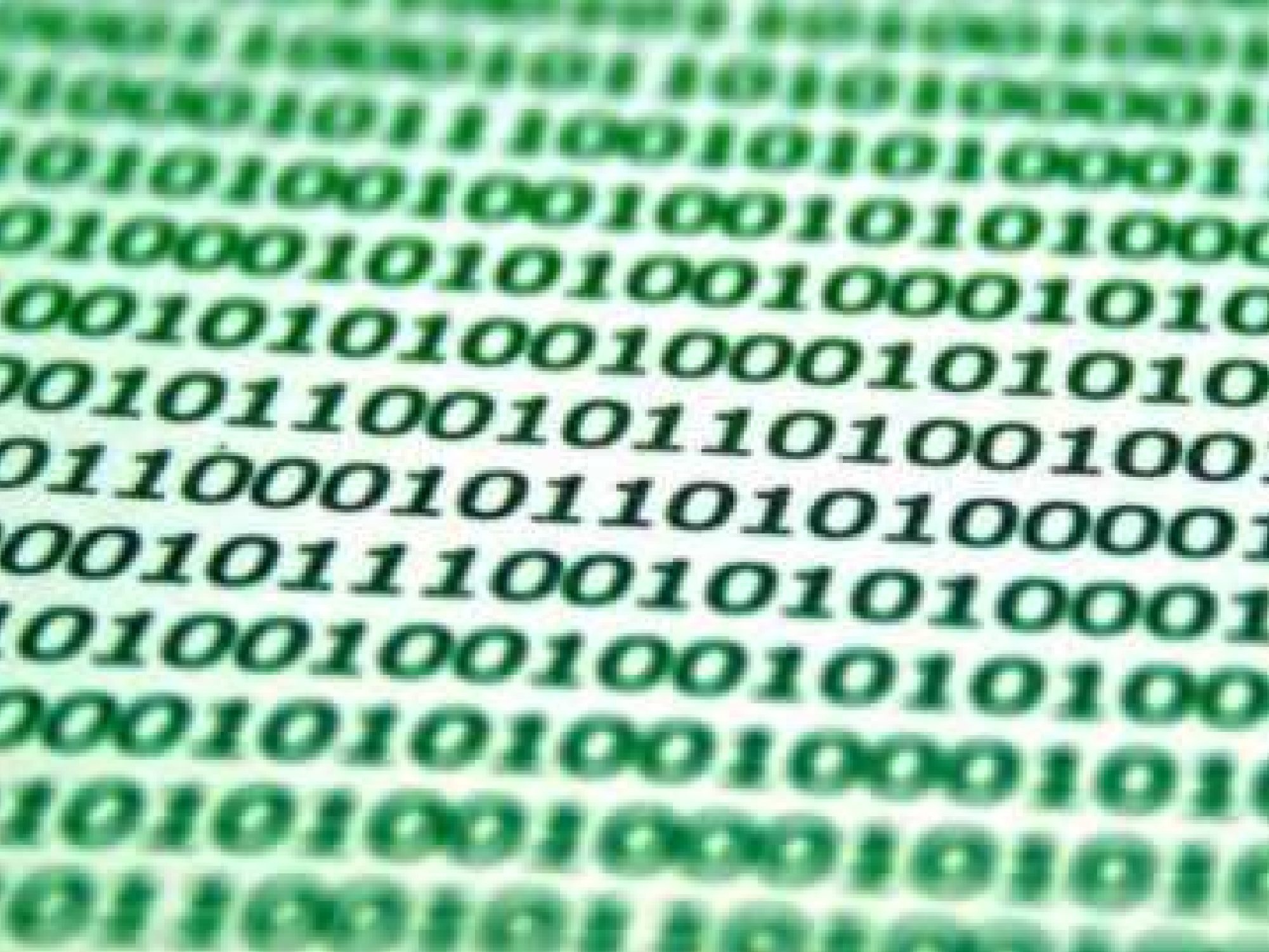
Documento

"Un documento è una frase scritta in un qualche *linguaggio* che abbia un *contenuto*, una *struttura*; può essere *in relazione con altri documenti*."

Documento Elettronico

"Un documento elettronico è un documento la cui rappresentazione fisica è *in forma di bit* all'interno di un sistema informatico."

“In forma di bit”



Un esempio "pratico"

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0
0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0
0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0
0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

- Questa potrebbe essere una porzione di un file sul vostro computer
- **Che cosa rappresenta?**

Parte I

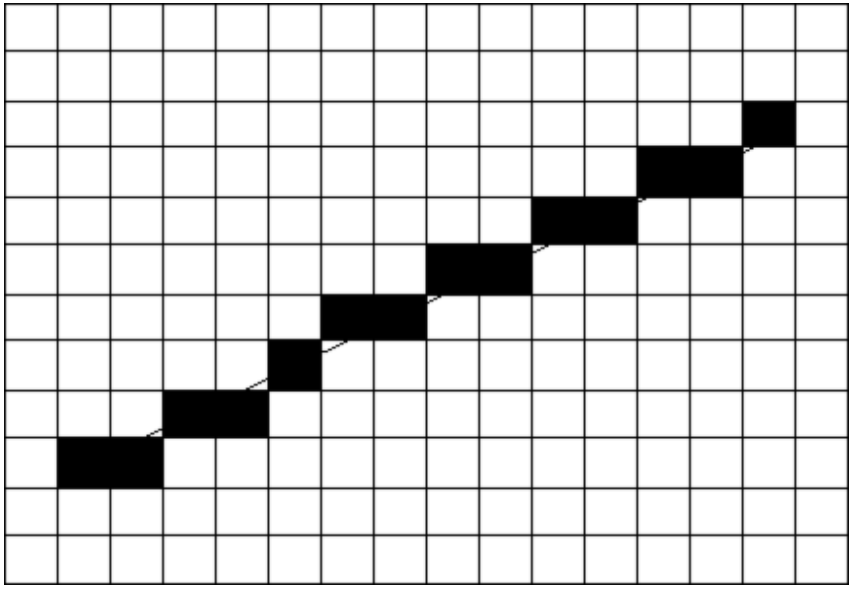
Codifica di dati semplici

Codifica

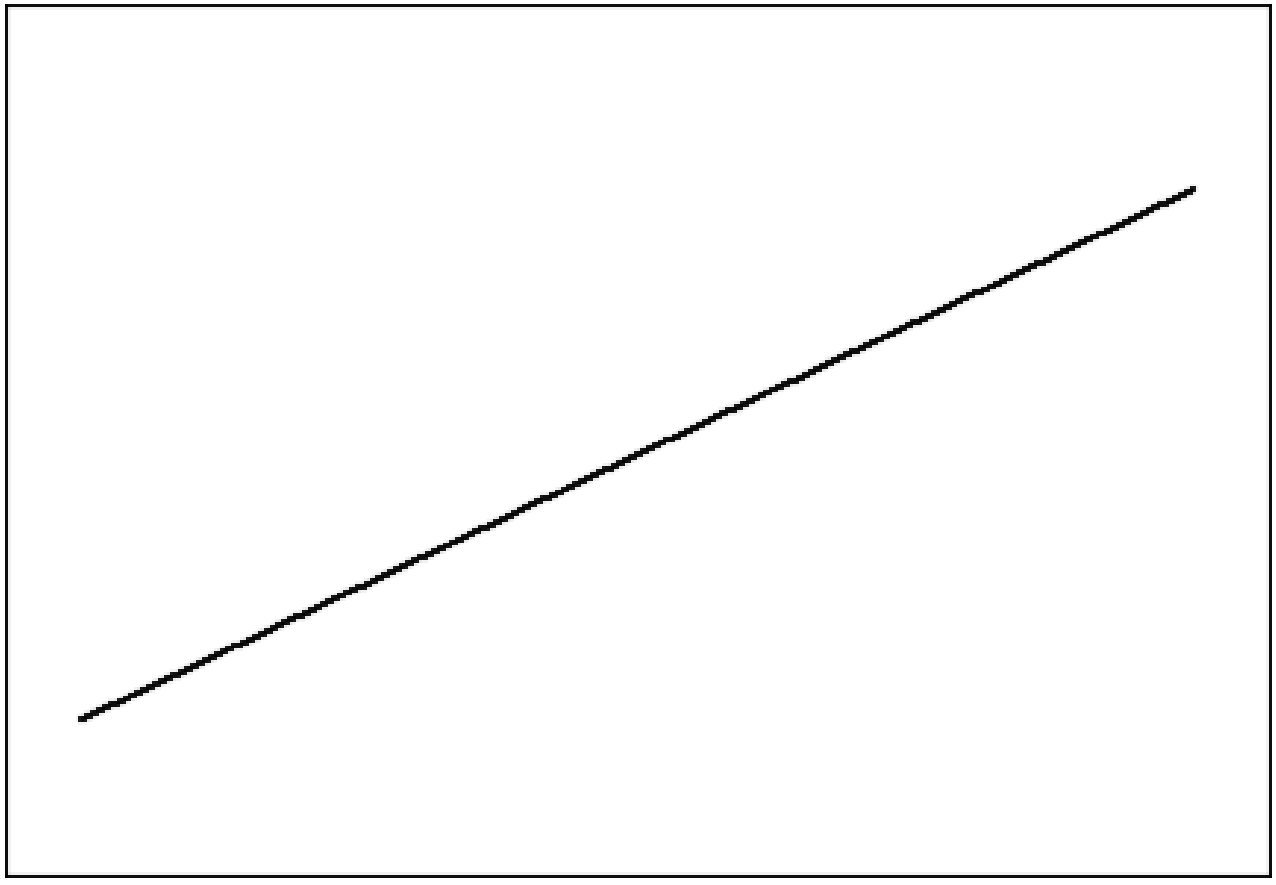
- "L'attività con cui si associa ad un **contenuto informativo** un **codice**" (definizione da wordnet – `http://wordnet.princeton.edu/`)
- ovvero **l'associazione tra tale contenuto informativo ed il codice**
- Nel nostro caso: codice = sequenza di bit

Per esempio:

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0
0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0
0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0
0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



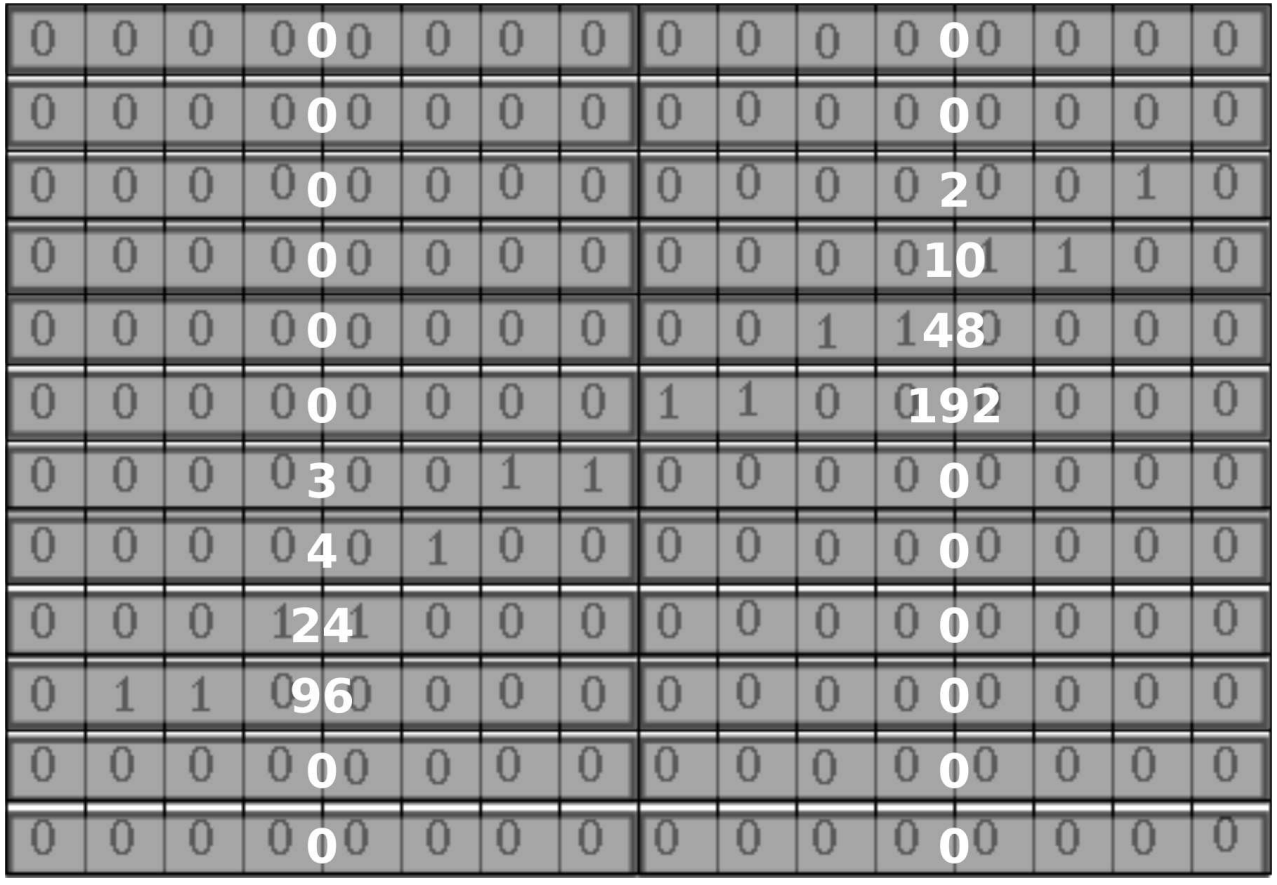
Vista da una certa distanza...



Codifica:

- pixel bianco $\leftrightarrow$ 0
- pixel nero $\leftrightarrow$ 1

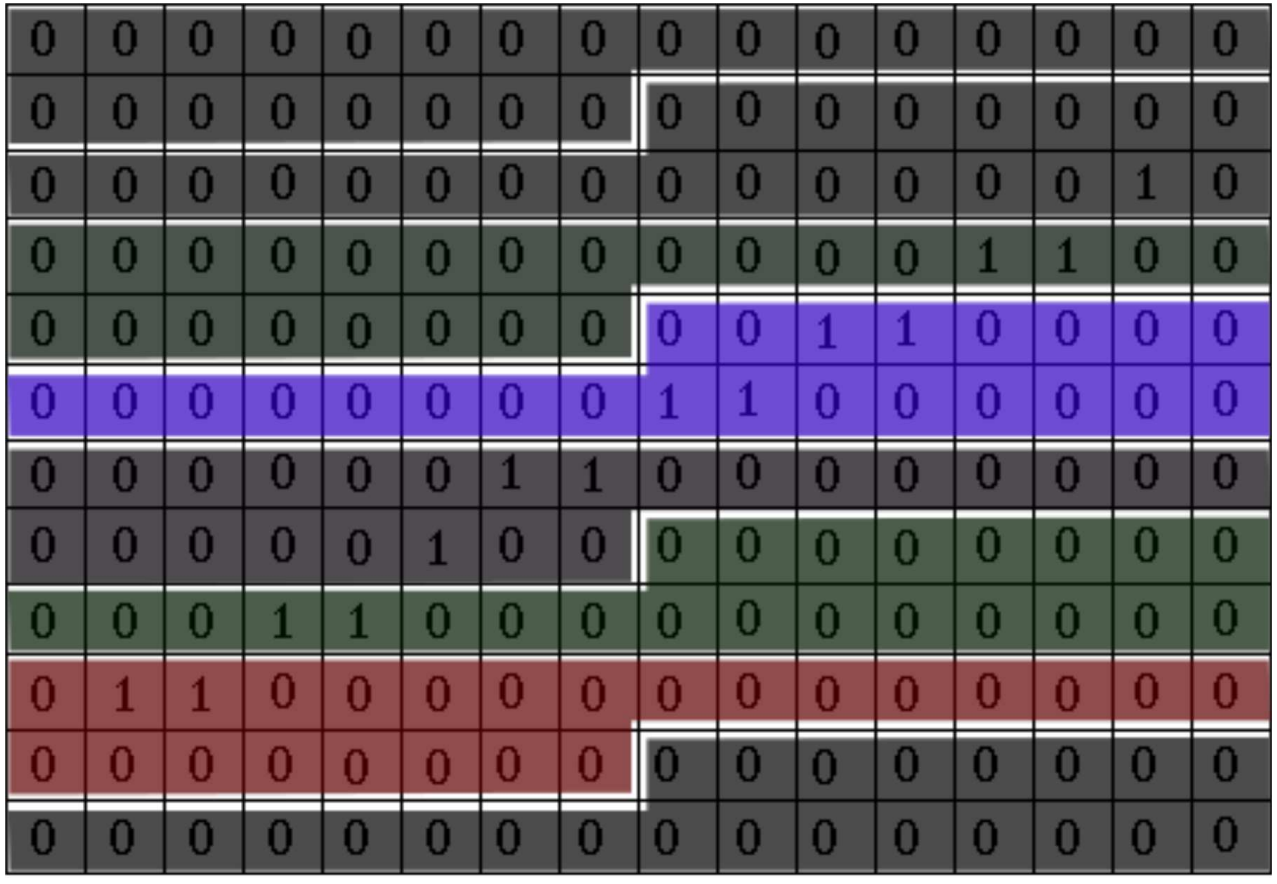
Un altro esempio



Codifica:

- numero intero $\leftrightarrow$ sua rappresentazione binaria

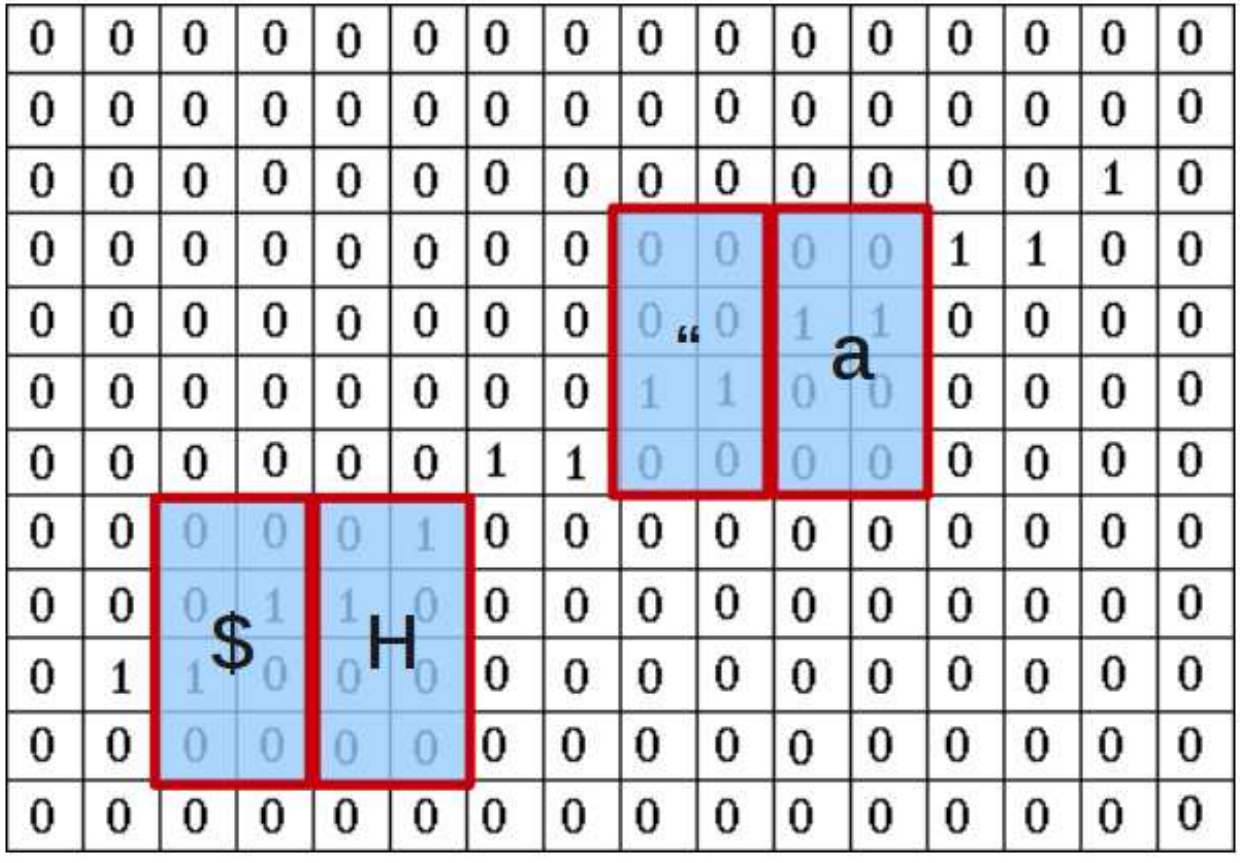
Un altro esempio



Codifica:

- colore $\leftrightarrow$ rappresentazione RGB

Un altro esempio



Codifica:

- lettera $\leftrightarrow$ rappresentazione ASCII-8

Rappresentazione dell'informazione

- Mediante una opportuna codifica è possibile rappresentare diversi tipi di informazione come sequenze binarie
- Nel calcolatore, **ogni** informazione è codificata come sequenza di bit
- Alcuni **esempi importanti**:
  - Numeri
  - Colori
  - Testo

Rappresentazione di numeri

- **Sistemi di numerazione posizionali:**
  - definiti su un *alfabeto* di due o più simboli (detti *cifre*), per una specifica *base* (che coincide con il numero di simboli nell'alftabeto)
  - un numero è rappresentato come una sequenza (*stringa*) di cifre
  - il *peso* di una cifra dipende dalla sua *posizione* nella stringa
- **Sistema decimale:** si basa su un *alfabeto* di 10 cifre:
  - 0, 1, 2, . . . , 9
  - base: 10
  - $0 \rightarrow$ zero
  - $12 \rightarrow$ dodici
  - $459 \rightarrow$ quattrocentocinquantanove
  - etc.

Interpretazione di una stringa decimale

Notazione: $\langle$**stringa**$\rangle_{\langle \textbf{base} \rangle}$

- Esempio: 134_{10} = '1' '3' '4' in base 10
  - la cifra più a destra vale 10^0
  - la seconda cifra più a destra vale 10^1
  - la terza più a destra vale 10^2 e così via
  - Quindi: $134_{10} = 1 \times 10^2 + 3 \times 10^1 + 4 \times 10^0$

Un possibile elemento di confusione:

- per *scrivere* un numero abbiamo bisogno di rappresentarlo
- ...quindi ci risulta molto difficile riferirci al risultato dell'espressione $1 \times 10^2 + 3 \times 10^1 + 4 \times 10^0$ in un modo diverso da "134"!

Sistemi posizionali e sistema binario

In generale:

- Data una sequenza di n **cifre** in base B: $c_{n-1} \dots c_2 c_1 c_0$ il **valore** di questa stringa è pari a:

$$c_{n-1} \times B^{n-1} + c_{n-2} \times B^{n-2} + \cdots + c_2 \times B^2 + c_1 \times B^1 + c_0 \times B^0$$

Nel computer si adotta il **sistema binario (in base due)**

- La cifra è detta **Bit** = **Bi**nary digi**t** e può valere 0 o 1
- Data una sequenza di n **cifre binarie**: $c_{n-1} \dots c_2 c_1 c_0$ il **valore** di questa stringa è pari a:

$$c_{n-1} \times 2^{n-1} + c_{n-2} \times 2^{n-2} + \cdots + c_2 \times 2^2 + c_1 \times 2^1 + c_0 \times 2^0$$

- . . . Perchè proprio questa rappresentazione nei computers?

Codifica esadecimale

- Per convenienza, i bit sono raggruppati in gruppi di 8 (**byte**)
- Altro sistema di numerazione spesso utilizzato: base 16
- Si basa su un alfabeto di 16 cifre
  - 0, 1, 2, . . . , 9, A, B, . . . , F
  - le cifre da A a F hanno valore da 10 a 15
- 16 valori diversi: possibili combinazioni di 4 cifre binarie

Codifica binaria (4 bit)	$\leftrightarrow$	**Cifra esadecimale**	$\leftrightarrow$	**Valore (decimale)**
0000	$\leftrightarrow$	0	$\leftrightarrow$	0
0001	$\leftrightarrow$	1	$\leftrightarrow$	1
0010	$\leftrightarrow$	2	$\leftrightarrow$	2
0011	$\leftrightarrow$	3	$\leftrightarrow$	3
. . .	$\leftrightarrow$	. . .	$\leftrightarrow$	. . .
1110	$\leftrightarrow$	E	$\leftrightarrow$	14
1111	$\leftrightarrow$	F	$\leftrightarrow$	15

Codifica esadecimale

- La codifica esadecimale consente di indicare il contenuto di un byte in modo compatto
- Esempio: `0101 1101` $\leftrightarrow$ 5D

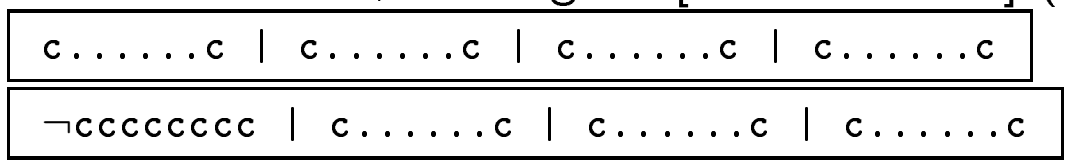
Codifica binaria (4 bit)	$\leftrightarrow$	Cifra esadecimale	$\leftrightarrow$	Valore (decimale)
0000	$\leftrightarrow$	0	$\leftrightarrow$	0
0001	$\leftrightarrow$	1	$\leftrightarrow$	1
0010	$\leftrightarrow$	2	$\leftrightarrow$	2
0011	$\leftrightarrow$	3	$\leftrightarrow$	3
0100	$\leftrightarrow$	4	$\leftrightarrow$	4
0101	$\leftrightarrow$	5	$\leftrightarrow$	5
0110	$\leftrightarrow$	6	$\leftrightarrow$	6
0111	$\leftrightarrow$	7	$\leftrightarrow$	7
1000	$\leftrightarrow$	8	$\leftrightarrow$	8
1001	$\leftrightarrow$	9	$\leftrightarrow$	9
1010	$\leftrightarrow$	A	$\leftrightarrow$	10
1011	$\leftrightarrow$	B	$\leftrightarrow$	11
1100	$\leftrightarrow$	C	$\leftrightarrow$	12
1101	$\leftrightarrow$	D	$\leftrightarrow$	13
1110	$\leftrightarrow$	E	$\leftrightarrow$	14
1111	$\leftrightarrow$	F	$\leftrightarrow$	15

Valori rappresentabili

- Universo dei valori rappresentabili: **dominio**.
- In un sistema in base B, la larghezza del dominio dipende da:
  - la base (es: $B = 2$, $B = 10$, $B = 16$)
  - Numero n di cifre a disposizione $\Rightarrow$ B^n valori rappresentabili
  - Es: con 3 cifre in base 10
    - $10^3 = 1000$ valori distinti (da 0 a 999)
- Con la codifica binaria, ragiono in termini di bit/byte:
  - con 1 bit: 2 valori distinti
  - con 1 byte: $2^8 = 256$ valori distinti
  - con 4 byte: $2^{4\times 8} = 2^{32} = 4 \times 2^{30} = 4$ miliardi di valori distinti
- Nomenclatura di uso comune:
  - $2^{10} \leftrightarrow$ **K** (Kilo–), $\sim 10^3 \Rightarrow$ migliaia
  - $2^{20} \leftrightarrow$ **M** (Mega–), $\sim 10^6 \Rightarrow$ milioni
  - $2^{30} \leftrightarrow$ **G** (Giga–), $\sim 10^9 \Rightarrow$ miliardi
  - $2^{40} \leftrightarrow$ **T** (Tera–), $\sim 10^{12} \Rightarrow$ migliaia di miliardi

Numeri senza segno, con segno, e decimali

- Rappresentazione dei numeri per un calcoratore
  - La memoria del computer non è infinita (limite fisico)
  - Per rappresentare ciascun numero: numero fisso di byte
  - Massimo intero rappresentabile: dipende dalla scelta fatta
- **Numeri con segno?**
  - Metà dominio per i negativi, metà per i positivi e lo zero
  - Esempio: 32 bit, senza segno: $[0..2^{32}-1]$ (0..4G)
    32 bit, con segno: $[-2^{31}..2^{31}-1]$ (-2G..2G)



- **Numeri decimali?**
  - **Segno, mantissa ed esponente**: $\pm 1.f \times 2^e$
    - $\pm$: segno
    - f: mantissa (precisione)
    - e: esponente (massimo e minimo non nullo)

Precisione singola e doppia

$$\pm 1 \cdot f \times 2^e$$

```
sfffffff | f...f | f...f | e...e
```

```
sfffffff | f...f | f...f | f...f | f...f | f...f | fffff eee | e...e
```

- **Precisione singola**:
  - mantissa 23 bit ($\sim$ 6 cifre decimali significative)
  - esponente 8 bit ($\pm \sim 10^{-44} .. \sim 10^{38}$)
- **Precisione doppia**:
  - mantissa 52 bit ($\sim$ 15 cifre decimali significative)
  - esponente 11 bit ($\pm \sim 10^{-323} .. \sim 10^{308}$)
- Combinazioni di bit riservate
  - $-\infty$, $+\infty$, NaN
- Limiti dell'approccio: overflow/underflow, numeri con infinite cifre (irrazionali o periodici)
- Precisione limitata alle somme di potenze (negative) di 2!

IEEE Standard 754 Floating Point Numbers

Colori

Codifica di colori

Un colore viene codificato nell'ambito di un preciso *spazio dei colori*

Spazio dei colori

Un modello matematico che descrive come rappresentare un colore mediante una n-pla di numeri

Un esempio importante: **spazio dei colori RGB**

- ogni colore viene rappresentato come composizione di tre colori primari
- colori primari: rosso (Red), Verde (Green), Blu (Blue)
- **uno specifico colore $\leftrightarrow$ tre numeri**
- ogni numero specifica l'intensità di ogni colore primario
- i colori primari vengono sommati al nero (modello addittivo)
- utilizzato nei monitor

Perché RGB?

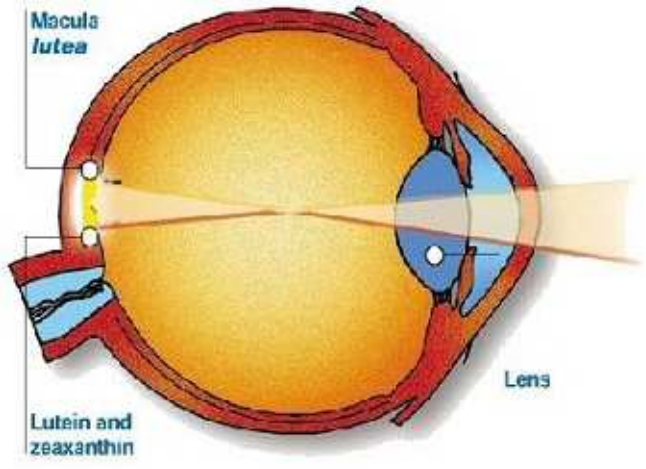
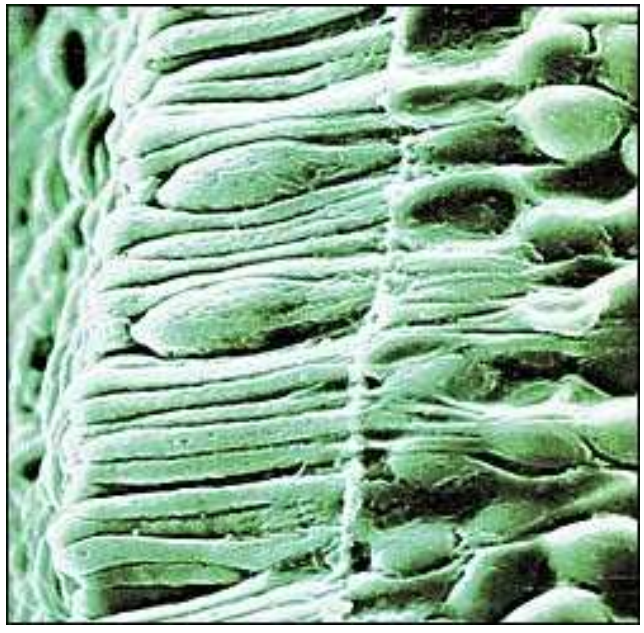


Figure 1: Model of the human eye.



- RGB perché noi uomini siamo **tricromi**
- I coni nella retina del nostro occhio sono sensibili alla luce di quei tre colori
- $\Rightarrow$ mediante composizione di rosso/verde/blu si possono rappresentare (in teoria) tutti i colori visibili

Codifica RGB

- Ogni colore è una **tripla** di numeri reali in $[0, 1]$:
  - (0,0,0) indica il nero, (1,1,1) indica il bianco
- Per limiti di memoria si utilizzano **numeri interi**
  - Il numero di colori rappresentabile dipende dal numero di bytes utilizzato per codificare ogni numero
  - **true color:** un byte per numero (da 0 a 255)
  - $\Rightarrow$ colori rappresentabili $= 2^8 \times 2^8 \times 2^8 \sim 16$ milioni
- Per compattezza quando si scrive un colore RGB si usa spesso la **codifica esadecimale**: `#000000`–`#FFFFFF`
- Esempi: `#FF0000` **rosso**, `#7F0000` **rosso scuro**, `#0000FF` **blu**, `#FF00FF` **viola**, `#7F007F` **viola scuro**, etc.

Altri spazi di colore

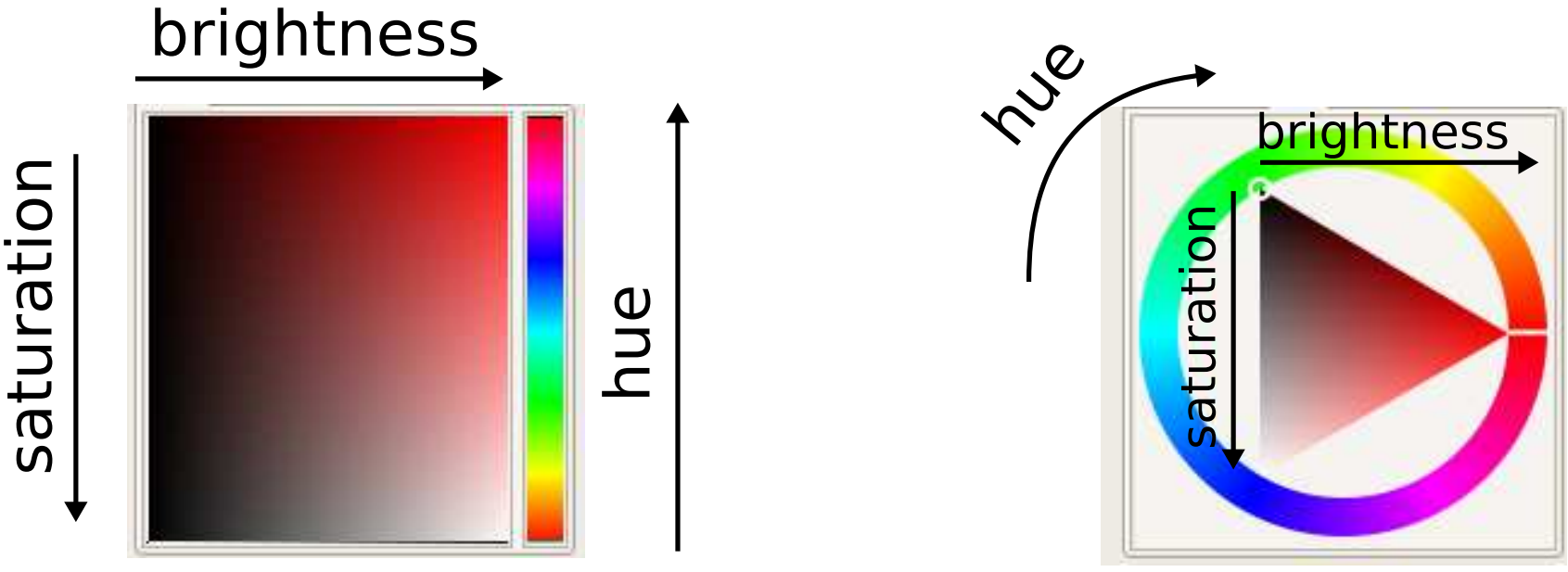
Tonalità, Saturazione e Luminosità

- in Inglese **Hue Saturation Brightness/Value (HSB or HSV)**
- Le tonalità (Hue) sono mescolate al bianco e/o nero
- Hue: nome di un colore puro, da 0 a 360. (0: rosso, 60: giallo, 120: verde, 180: ciano, 240: blu, 300: magenta)
- Saturation: presenza di bianco, da 0 a 100%
- Brightness: complementare della presenza di nero, da 0 a 100%

Perché HSV?

- perché tende ad essere più intuitivo di RGB per un essere umano
- è possibile "tradurre" un colore da uno spazione all'altro (funzione biiettiva)

RGB contro HSV



- I selettori di colori all'interno di programmi di grafica si basano tipicamente sul modello HSV
- È decisamente più difficile capire che colore si va ad ottenre specificando la quantità di rosso, verde e blu

Altri spazi di colore

Cyano Magenta Giallo Nero

- in Inglese Cyan, Magenta, Yellow, blacK (CMYK)
- modello **sottrattivo** (i colori vengono sottratti dal bianco)
- utilizzato nelle stampanti (gli inchiostri *assorbono* luce); di solito vi si riferisce con il termine "quadricromia"
- colore = **quadrupla** di numeri in $[0, 1]$; $(0, 0, 0, 0)$ **è il bianco**

Una nota importante:

- CMYK non è in grado di rappresentare tutti i colori visibili$\Rightarrow$ esistono colori che **non** possono essere stampati in quadricromia
- Uno stesso colore può essere (in linea di principio) codificato in modo diverso (funzione non iniettiva)
  - es. $(0, 0, 0, 1) \leftrightarrow$ nero, $(1, 1, 1, 1) \leftrightarrow$ nero
  - su schermo possono sembrare identici, purtroppo **in stampa non è così**. Codificate **sempre** il nero con $(0, 0, 0, 1)$
  - traduzine RGB $\leftrightarrow$ CMYK più complessa

Caratteri


Tack
Vielen Dank
Obrigado
Merci
ありがとうございます
Bedankt
Takk
感謝您
Terima Kasih
谢谢
Grazie
ขอบคุณ
Спасибо
Thank You
Kiitos
Tak
Teşekkür Ederiz
감사합니다
Gracias
Σας ευχαριστούμε
Dziękujemy

Caratteri

- **Carattere**: qualunque segno grafico utilizzato in tipografia per rappresentare le lettere, i segni di interpunzione, le cifre, etc.
- Tre diversi aspetti da considerare:
  - **Natura**: che cosa rappresenta.
    - 2 caratteri aventi la stessa natura: σ / ς
    - 2 caratteri aventi diversa natura: a / å
    - variazioni della stessa lettera: a / à
  - **Glifo**: la forma ottenuta in stampa
    - diversi glifi, stessa natura: a, a, a, *a*, **a**
  - **Codifica**: scelta della sequenza di simboli binari corrispondenti
    - es: a $\rightarrow$ 1100001
- Per ora ci focalizziamo sulla **codifica**

Codifica dei caratteri

- Come codificare i caratteri?
- Problema semplice da risolvere con un alfabeto:
  - contare quanti lettere e simboli si vogliono rappresentare
    - es: `A`, `B`, ..., `Z`, `a`, `b`, ..., `z`, `0`, `1`, ..., `9`, `!`, , `#`, €, `(`, `)`, `?`, `:`, ... $\Rightarrow$ 100 simboli
  - prendere la minima potenza di due che ne contiene il numero
    - $100 \Rightarrow 2^7 = 128$
  - l'esponente del 2 identifica il numero di bit necessario
    - $128 \Rightarrow 7$
  - stabilire una corrispondenza tra stringhe di (7) bit e caratteri
- Problemi: **armonizzazione** tra codifiche di alfabeti diversi e tra architetture diverse
- Soluzione: definizione di **standard**

ASCII

- **American Standard Code for Information Interchange**:
  - usa 7 bit (128 caratteri)
  - alcune combinazioni sono usate per *caratteri speciali* (a-capo, tabulazione, beep, . . .)
  - orientato alla lingua inglese (alfabeto latino, senza accenti)
- Però:
  - molte altre lingue con diversi alfabeti
  - sforzi simili all'ASCII (arabo, greco, cirillico, cinese, etc.)
  - produttori di hardware: codifiche a 8 bit (256 caratteri, di cui solo i primi 128 standard ASCII)
- Problemi:
  - Interoperabilità nel passaggio di documenti da un sistema operativo a un altro
  - Sbilanciamento verso la lingua inglese
  - Documenti scritti in due lingue?

ISO Latin e Unicode

- **ISO Latin** (ISO Latin-1, ISO Latin-9)
  - codifica a 8 bit
  - standardizzazione della codifica degli alfabeti europei
  - compatibilità con ASCII
- **Unicode: Unified Character Set**
  - UCS-2: codifica a 16 bit
    - di norma: il primo byte identifica l'alfabeto
    - Es: `00.....0` corrisponde a ISO Latin-1
  - UCS-4: codifica a 32 bit
    - usata solo per alfabeti con moltissimi caratteri diversi: cinese letterario, alfabeti antichi (geroglifici, cuneiforme, etc.)
- **Unicode: Unified Transformation Format** (UTF-8)
  - Usa pochi byte per i caratteri più frequenti:
    - 8 bit per ASCII
    - 16 bit per ISO Latin-1 e per alfabeti non latini più comuni
    - 24/32 bit per alfabeti orientali e storici

File di testo

- Un file di testo è u file che contiene **solo caratteri**
- Il suo contenuto è la sequenza delle codifiche di tali caratteri

Esempio: file `ciao.txt`

Con notepad:

```
ciao ragazzi
```

Con un editor binario:

```
63 69 61 6F 20 72 61 67
61 7A 7A 69 0A
```

Per la cronaca:

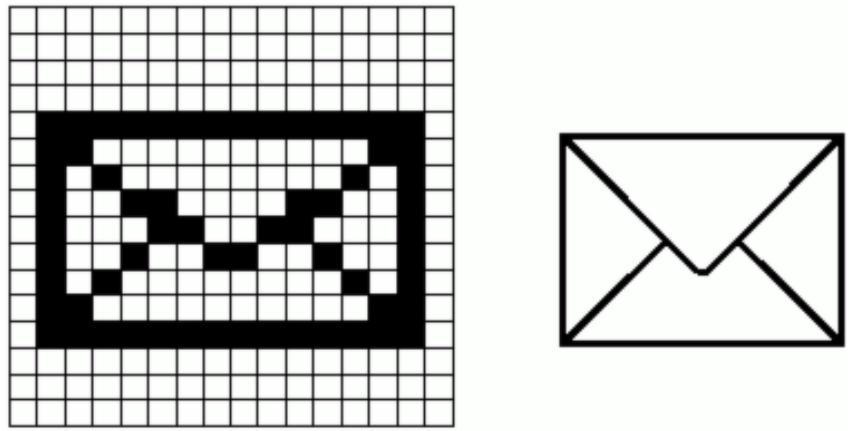
- Codifica ASCII $\Rightarrow$ 7/8 bit per carattere
- 20 è lo spazio
- 7A è la zeta
- 0A indica che il file è finito

Parte II

Immagini ed elementi grafici

Elementi grafici

Modalità di resa grafica digitale



Due grandi famiglie di modalità di resa grafica:

- **Bitmap**, o **raster**, orientata al **pixel**.
  - **immagine = griglia** di punti e colori.
  - adatta ad immagini catturate dalla realtà (es. fotografie)
  - Si prestano a visualizzazione a video.
- **Vettoriale**, orientata alle **linee, curve, aree**.
  - **immagine = insieme di primitive di disegno** (una primitiva è una istruzione per disegnare un oggetto geometrico)
  - adatta ad immagini sintetiche (artificiali)
  - si prestano a *ridimensionamento* e resa di font.

Raster o vettoriale?

Raster o vettoriale?

Bitmap/raster

Immagine memorizzata come griglia di $n \times m$ punti (**dot** o **pixel**)

La **qualità** dipende da:

- **risoluzione** $(n, m) \rightarrow$ quanti pixel compongono l'immagine
- **profondità di colore** $\rightarrow$ quanti colori sono rappresentabili
  - **bicromia** (1 bit per dot)
  - **palette** (1 byte per dot, 256 colori – anche altre possibilità)
  - **true color** (3 bytes per dot, 16ML colori)

La **resa** dipende anche dal **mezzo di output** utilizzato:

- risoluzione del video o della stampante
- misurata in **dot per inch (dpi)**
- da non confondere con la risoluzione spaziale
- dimensioni in stampa = lunghezza (o larghezza) in pixel / dpi

Formati bitmap

Per immagini raster: maggiore qualità = maggiori **dimensioni**

- Compressione per ridurre le dimensioni delle immagini
- Compressine **senza perdita di qualità (lossless)**
  - **BMP**, nessuna compressione: direttamente visualizzabile su schermo
  - **GIF** (Graphic Interchange Format) e **PNG** (Portable Network Graphics), basati su **palette** –GIF usato anche per semplici animazioni
  - **TIFF** (Tagged Image File Format), **contenitore** per vari formati; usato anche per fax, scansioni, etc., anche su più pagine
- Compressione **con perdita di qualità (lossy)**
  - **JPEG** (Joint Photographic Expert Group), usa true color
  - Algoritmo che elimina le sfumature difficilmente percettibili.
  - Possibilià di scegliere il livello di compressione.

Compressione lossless e lossy

PNG

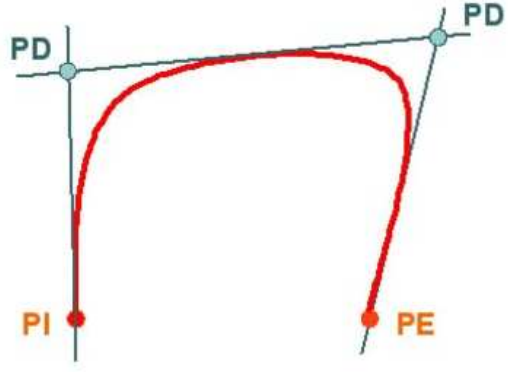
JPG

Immagini vettoriali

Immagine = collezione di elementi geometrici, descritti mediante modelli matematici

- **Vettori** usati per definire gli elementi di base
- Punto di partenza, lunghezza, direzione

- **Curve di Bézier**, per rappresentare oggetti complessi e curvilinei
- Punti di inizio e fine della curvatura (PI, PE)
- Punti di direzione (PD)



Risoluzione di una immagine vettoriale

Una immagine vettoriale può essere visualizzata a **risoluzione arbitraria** (scalabilità)

- la qualità di una immagine vettoriale non dipende dalla dimensione del file
- la dimensione del file riflette solo il numero di elementi matematici nel disegno

Rasterizzazione

- il processo per cui una immagine vettoriale viene convertita in una matrice di punti
- poichè i monitor sono **dispositivi raster** una fase di rasterizzazione è sempre richiesta per la resa a video

Raster o vettoriale?

Caratteri

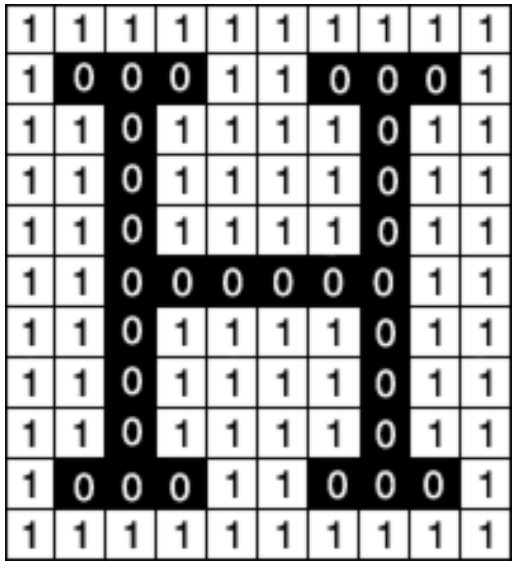
- **Carattere**: qualunque segno grafico utilizzato in tipografia per rappresentare le lettere, i segni di interpunzione, le cifre, etc.
- Tre diversi aspetti da considerare:
  - **Natura**: che cosa rappresenta.
    - 2 caratteri aventi la stessa natura: σ / ς
    - 2 caratteri aventi diversa natura: a / å
    - variazioni della stessa lettera: a / à
  - **Glifo**: la forma ottenuta in stampa
    - diversi glifi, stessa natura: a, a, $\mathbb{a}$, *a*, **a**
  - **Codifica**: scelta della sequenza di simboli binari corrispondenti
    - es: a $\rightarrow$ 1100001

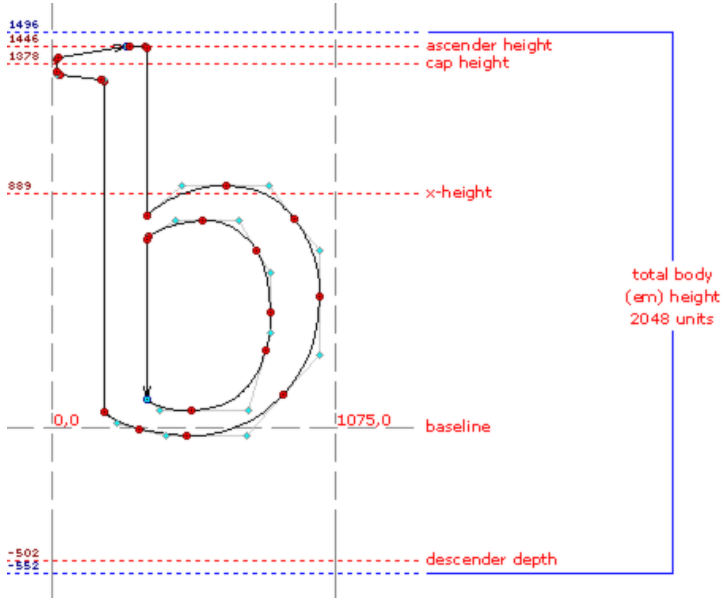
Ora prendiamo in considerazione la rappresentazione del glifo corrispondente ad un carattere

Fonti tipografiche

- **Fonte tipografica digitale**: assortimento completo di caratteri in una certa dimensione e stile
- Specifica per ogni carattere il **glifo** da utilizzare
- **Glifo**: rappresentazione grafica dell'aspetto di un carattere
- Caratteristiche stilistiche di una fonte:
  - Presenza di **grazie** o **serif** (piedini terminali di abbellimento)
  - **Proporzionalià della larghezza** dei glifi
    - Serif, proporzionale:
      Times Roman
    - Sans serif, proporzionale:
      Helvetica
    - Serif, non proporzionale (*monotype*):
      Courier

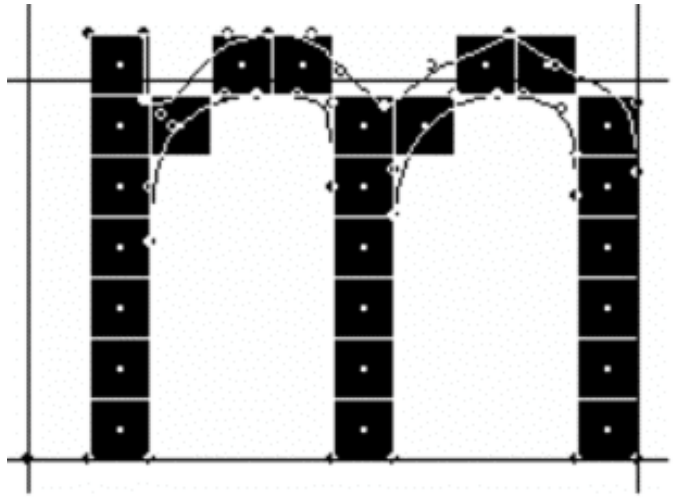
Tipi di fonti tipografiche





- **Bitmap**
  - Glifi rappresentati come **immagini** separate di caratteri.
  - Dimensione precisa (e.g. 12 punti). Scarsa qualità.
  - Non più usate.
- **Vettoriali (scalabili)**
  - Glifi rappresentati come **insiemi di formule**.
  - Rendering grafico attraverso **rasterizzazione**.

Rasterization e anti-aliasing



- **Rasterization**: Scelta dei pixel che devono essere accesi per riprodurre un glifo di una fonte vettoriale.
- **Anti-aliasing**: tecnica per ridurre l'effetto della visualizzazione "scalettata" delle immagini
  - Introduzione di tonalità di colore (grigio)

Parte III

Documenti Strutturati e Linguaggi di Markup

Struttura di un documento

Documento

"Un documento è una frase scritta in un qualche *linguaggio* che abbia un *contenuto*, una *struttura*; può essere *in relazione con altri documenti*."

Fin qui abbiamo visto:

- la modalità di codifica **(linguaggio)** di alcuni tipi di informazione (numeri, caratteri, colori)...
- ...che costituiscono il **contenuto** di un documento

In aggiunta, un documento ha una specifica **struttura**:

- **definita a priori:** per esempio la struttura di un file immagine è definita dallo standard dei vari formati (PNG, JPG...)
- **parte del contenuto del documento stesso:** per esempio i capitoli, sezioni e paragrafi di un libro

Un documento di testo, strutturato in sezioni

Ecloga 1

Strofe 1

Tityre, tu patulae recubans sub tegmine fagi
silvestrem tenui musam meditaris avena;
nos patriae fines et dulcia linquimus arva;
nos patriam fugimus: tu, Tityre, lentus in umbra
formosam resonare doces Amaryllida silvas.

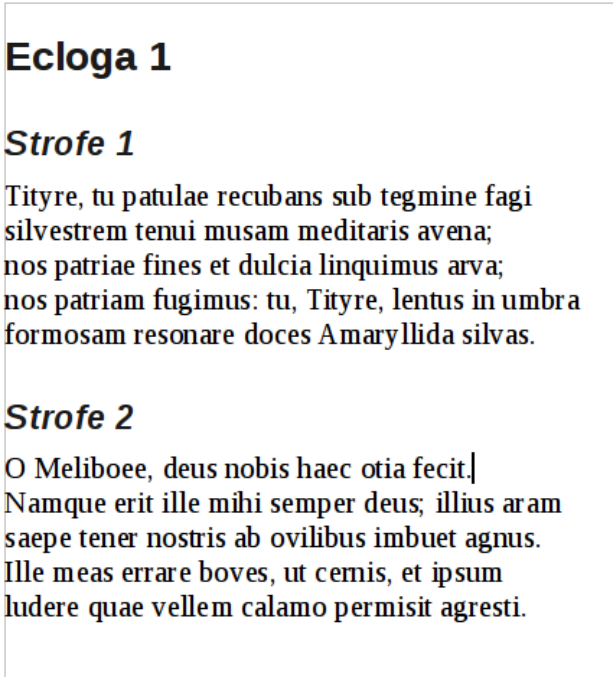
Strofe 2

O Meliboee, deus nobis haec otia fecit.
Namque erit ille mihi semper deus; illius aram
saepe tener nostris ab ovilibus imbuet agnus.
Ille meas errare boves, ut cernis, et ipsum
ludere quae vellem calamo permisit agresti.

Struttura e presentazione

Le informazioni sulla struttura di un documento sono contettualmente **distinte** da quelle sulla sua presentazione

- che una porzione di testo “strofe 1” sia una sezione non dice **nulla** sul modo in cui vada visualizzata
- alcuni programmi per la composizione di documenti di testo permettono di visualizzare la struttura del documento



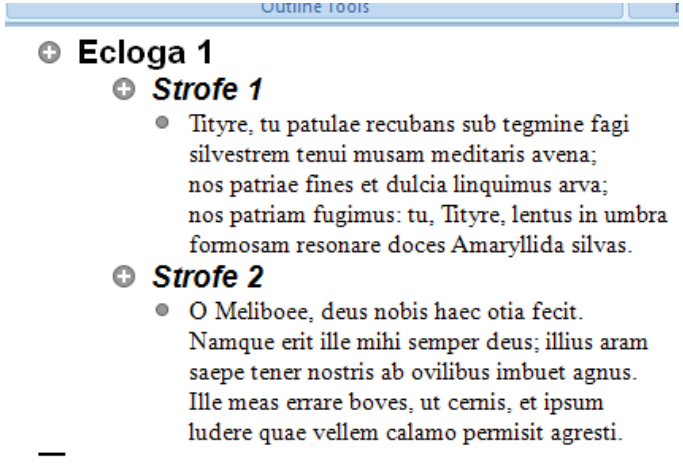
Ecloga 1

Strofe 1

Tityre, tu patulae recubans sub tegmine fagi
silvestrem tenui musam meditaris avena;
nos patriae fines et dulcia linquimus arva;
nos patriam fugimus: tu, Tityre, lentus in umbra
formosam resonare doces Amaryllida silvas.

Strofe 2

O Meliboee, deus nobis haec otia fecit.
Namque erit ille mihi semper deus; illius aram
saepe tener nostris ab ovilibus imbuet agnus.
Ille meas errare boves, ut cernis, et ipsum
ludere quae vellem calamo permisit agresti.



Struttura e presentazione

Questa **separazione tra informazioni di struttura e di presentazione** è alla base del meccanismo degli **stili** (HTML, MS Word, OpenOffice Writer...):

- per ogni elemento di struttura (capitolo, sezione...)
- l'utente può specificare una modalità di presentazione (grassetto, sottolineato, dimensione del carattere...)

Il metodo può essere generalizzato ad informazioni che non specificano struttura, ma neanche una modalità di presentazione:

- "testo importante", "citazione", "riferimento bibliografico"

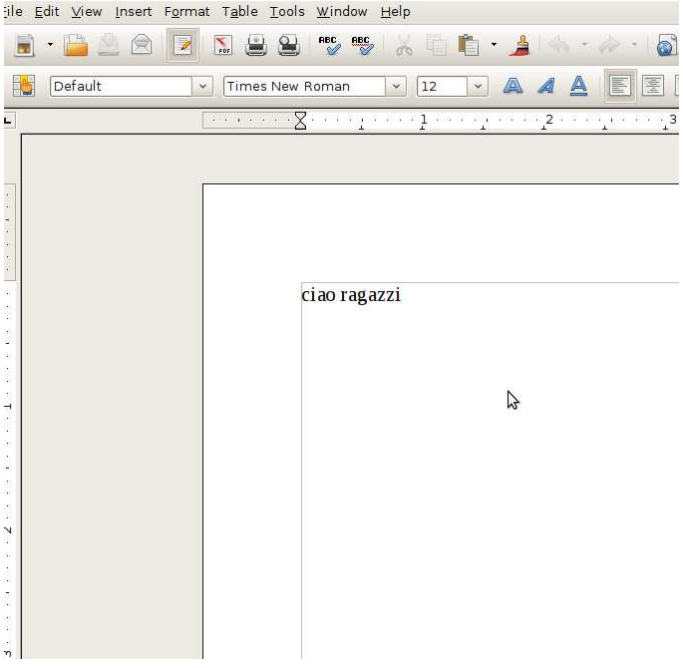
Il markup

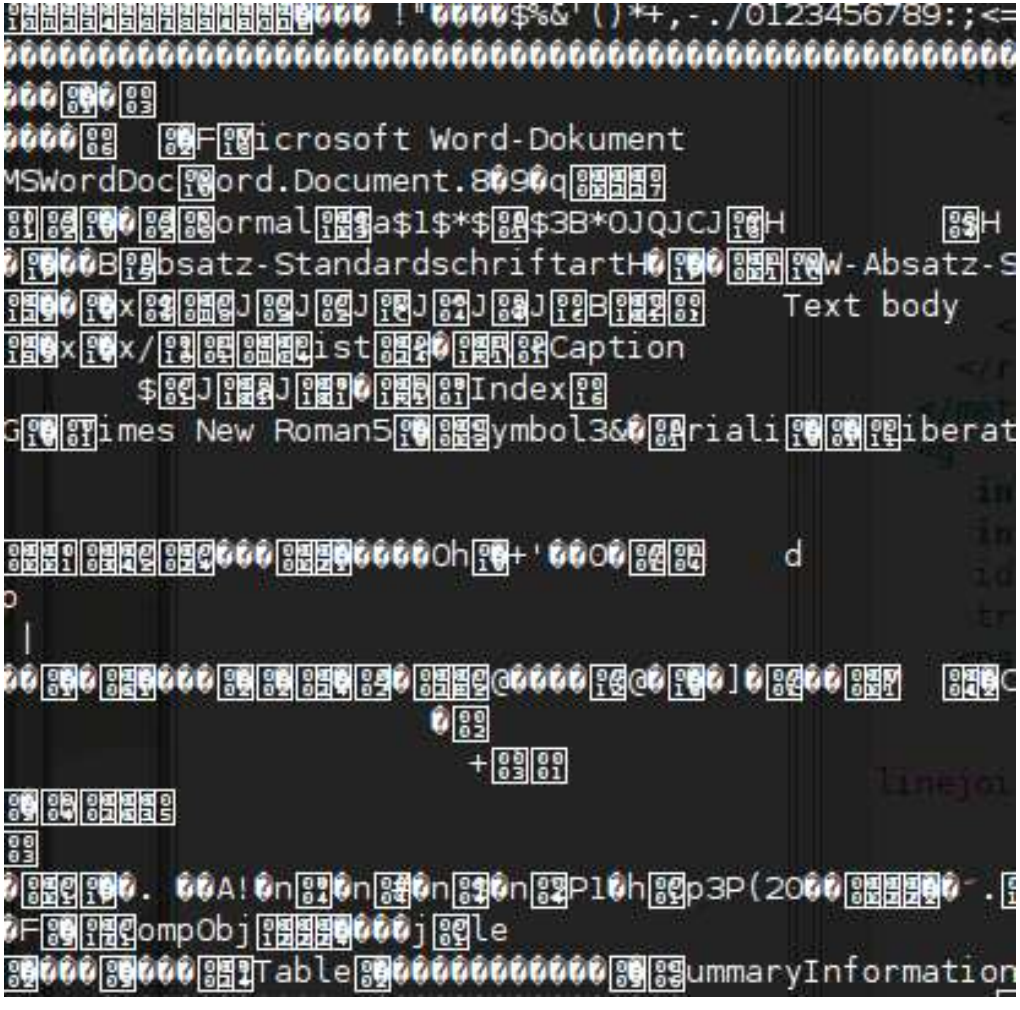
- Markup: qualunque informazione si aggiunga ad un contenuto per renderlo più comprensibile o utilizzabile:
  - nel testo: neretto, corsivo, divisione in pagine, uso dei margini, struttura del documento etc.
- **Preparare un documento con un programma applicativo $\Rightarrow$ aggiungere markup al contenuto**
- Ciascun programma applicativo propone un **formato dati** per indicare il markup e associarlo al contenuto
- Esempi di linguaggi di markup:
  - **HTML**, XML (W3C)
  - Formati proprietari:
    - Word, Excel, Power Point (Microsoft, Office), RTF
    - Pages, Numbers, Keynote (Apple, iWork)
    - StarOffice, OpenOffice (SUN)
    - WordPerfect (Corel), etc.
  - Linguaggi di programmazione: PostScript, PDF (Adobe), LaTeX

Caratteristiche dei linguaggi di markup a confronto

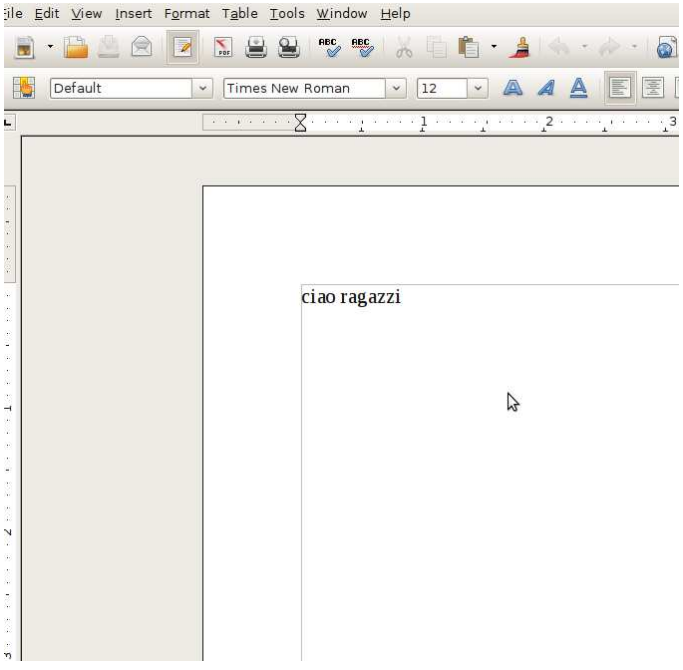
- Leggibilità
  - **Formato binario**: uso di rappresentazioni interne non testuali, per codifica diretta dei dati
  - **Formato leggibile**: uso di caratteri speciali: `<...>`, `{...}`, ...
- Tipo di informazioni
  - **Presentazionale**: contiene istruzioni di presentazione tipografica (orientato alla visualizzazione)
  - **Descrittivo**: descrive il ruolo dei vari elementi del documento (*include* le informazioni di struttura)
- Proprietà del linguaggio di markup
  - **Proprietario**: le caratteristiche del linguaggio sono controllate da un'impresa commerciale (minore interoperabilità, aggiunta più rapida di nuove funzionalità)
  - **Non proprietario**: organizzazione senza fini di lucro o consorzio (apertura, pubblicazione, interoperabilità)

Un linguaggio di markup binario: ".doc" di MS Word





Un linguaggio di markup testuale: Adobe PostScript




```
%%IncludeFeature: *PageSize A4
} stopped cleartomark
[{
%%IncludeFeature: *InputSlot Default
} stopped cleartomark
%%EndSetup
%%Page: 1 1
%%PageOrientation: Portrait
%%PageBoundingBox: 18 36 577 806
%%BeginPageSetup
%
%%EndPageSetup
gsave
[0.12 0 0 -0.12 18 806] concat
gsave
323 266 moveto
0 setgray
(LiberationSerifFID182HGSet1) cvn findfont 100 -100 matrix scale makefont setfont
<6369616F20726167617A7A69>
[44 29 43 51 25 33 44 50 45 44 44 0]
xshow
grestore grestore
showpage
%%PageTrailer

%%Trailer
%%BoundingBox: 0 0 595 842
%%Orientation: Portrait
%%Pages: 1
%%EOF
```

- Il linguaggio PostScript è uno standard de-facto per la stampa digitale
- Un file postscript dice alla stampante cosa deve fare per rappresentare il documento

HTML

HTML è un linguaggio di markup testuale

- nato per essere prodotto a mano ($\Rightarrow$ è *davvero* leggibile)
- il testo a cui si riferiscono informazioni di markup è delimitato da **tag** di apertura e di chiusura
- i tag hanno un nome e sono individuati tramite l'uso delle parentesi uncinate: `<,>`
- per ogni tag di apertura `<nometag>` il corrispondente tag di chiusura è `</nometag>`
- esistono tag che non si riferiscono ad alcun testo; si presentano come `<nometag/>`

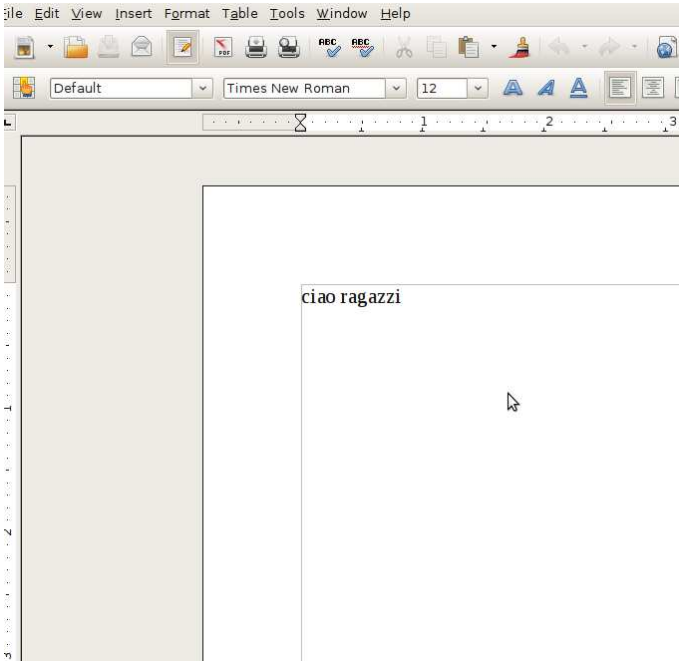
Tag in HTML

- 4 categorie di tag
  - **strutturali** (titoli, sottotitoli, elenchi, tabelle, etc.)
  - **presentazionali** (grassetto, corsivo, etc.)
  - **ipertestuali** (link)
  - **multimediali** (immagini e altri oggetti multimediali)
- Alcuni esempi:
  - *titolo:* `<h1> ... </h1>`
  - *elenchi numerati:* (ordered list & list items)

```
<ol>
    <li> ... </li>
    ...
    <li> ... </li>
</ol>
```

  - *grassetto:* `<b> ... </b>`
  - *URI:* `<a src="..."> ... </a>`
  - *immagine:* `<img src="..."> ... </img>`
- URI, immagini, etc. mettono il documento HTML in relazione con altri documenti

Un esempio di documento HTML




```
<html>
<head>
<title>Ciao ragazzi</title>
</head>
<body>ciao ragazzi</body>
</html>
```

- HTML nasce con scopi diversi dal formato doc, o ps e non ha le stesse potenzialità di resa su stampa
- orientato alla descrizione di **ipertesti**
- un documento HTML non è sempre così "simpaticamente" compatto

Fogli di stile

- Per definire la presentazione (grafica o attraverso media alternativi) si possono usare **fogli di stile**.
- In HTML sono realizzati secondo uno standard che si chiama **Cascading Style Sheet** (CSS).
- un CSS **specifica come visualizzare determinati tag** (o gruppi di tag)
- esempio:

```
h1 { font-family:  Arial; font-size:  120%; color:#666699; }
```

- 2 possibili modi di specificare un CSS:
  - **Dentro l'HTML**, mediante il tag <style> nell'header;

```
<style>
   h1    { font-family: Arial; font-size: 120%; color:#666699; }
   ol    { font-family: Arial; color: #6666FF; }
   p     { font-family: Arial; font-size: 100%; color:#333333; }
</style>
```

  - **Come file esterno**, inserendo un <link> nell'header.

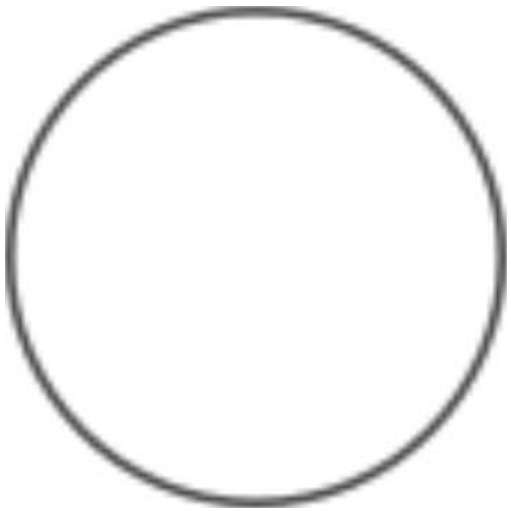
```
<link media="screen" href="sample-style.css" rel="stylesheet">
```

Definire un linguaggio di markup

- Esistono molti linguaggi di mark-up, progettati con caratteristiche diverse a seconda dell'applicazione
- Per **descrivere** in modo rigoroso un linguaggio, bisogna usare un **meta-linguaggio**
- **XML** è un meta-linguaggio di markup molto usato
- notazione simile ad HTML (uso delle parentesi uncinate <,>)
- **Document Type Definition** (DTD)
  - documento scritto in un meta-linguaggio, es: XML
  - elenca gli elementi di markup e le regole di strutturazione che descrivono le caratteristiche di un nuovo linguaggio di markup
- DTD XML usati per definire versioni e varianti di HTML
  - HTML 4.01 Strict
  - HTML 4.01 Transitional
  - XHTML 1.0 Strict, etc.
- `http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd`

Un linguaggio basato su XML: SVG

- **SVG = Scalable Vector Graphics** è un formato per la descrizione di grafica vettoriale
- un file SVG contiene di fatto testo in XML



```
<g
 inkscape:label="Layer 1"
 inkscape:groupmode="layer"
 id="layer1"
 transform="translate(-49.002552,-40.199482)">
<path
   sodipodi:type="arc"
   style="opacity:0.7;fill:none;stroke:#000000;
          stroke-width:2;stroke-linecap:round;
          stroke-linejoin:round; stroke-miterlimit:4;
          stroke-opacity:1;stroke-dasharray:none;
          stroke-dashoffset:0"
   id="path2983"
   sodipodi:cx="101.52033"
   sodipodi:cy="92.717262"
   sodipodi:rx="51.51778"
   sodipodi:ry="51.51778"
   d="m 153.03811,92.717262 c 0,28.452488 -23.06529,51.517778
      -51.51778,51.517778 -28.452482,0 -51.517778,-23.06529
      -51.517778,-51.517778 0,-28.452484 23.065296,-51.51778
      51.517778,-51.51778 28.45249,0 51.51778,23.065296
      51.51778,51.51778 z" />
</g>
```