Testo d'esame

L. Console, M. Ribaudo, U. Avalle
F. Carmagnola, F. Cena

Introduzione all' Informatica

UTET Università-2010

Indice

Indice

Informatica

- Informatica - scienza dell'informazione

Inform - automatica

- *Scienza del trattamento razionale, particolarmente con macchine automatiche, delle informazioni considerate come supporto delle conoscenze umane e delle comunicazioni in tutti i settori.*
- Da non confondere con scienza dei calcolatori, ma scienza del trattamento dell'informazione

Informatica

Definizione di Informatica:

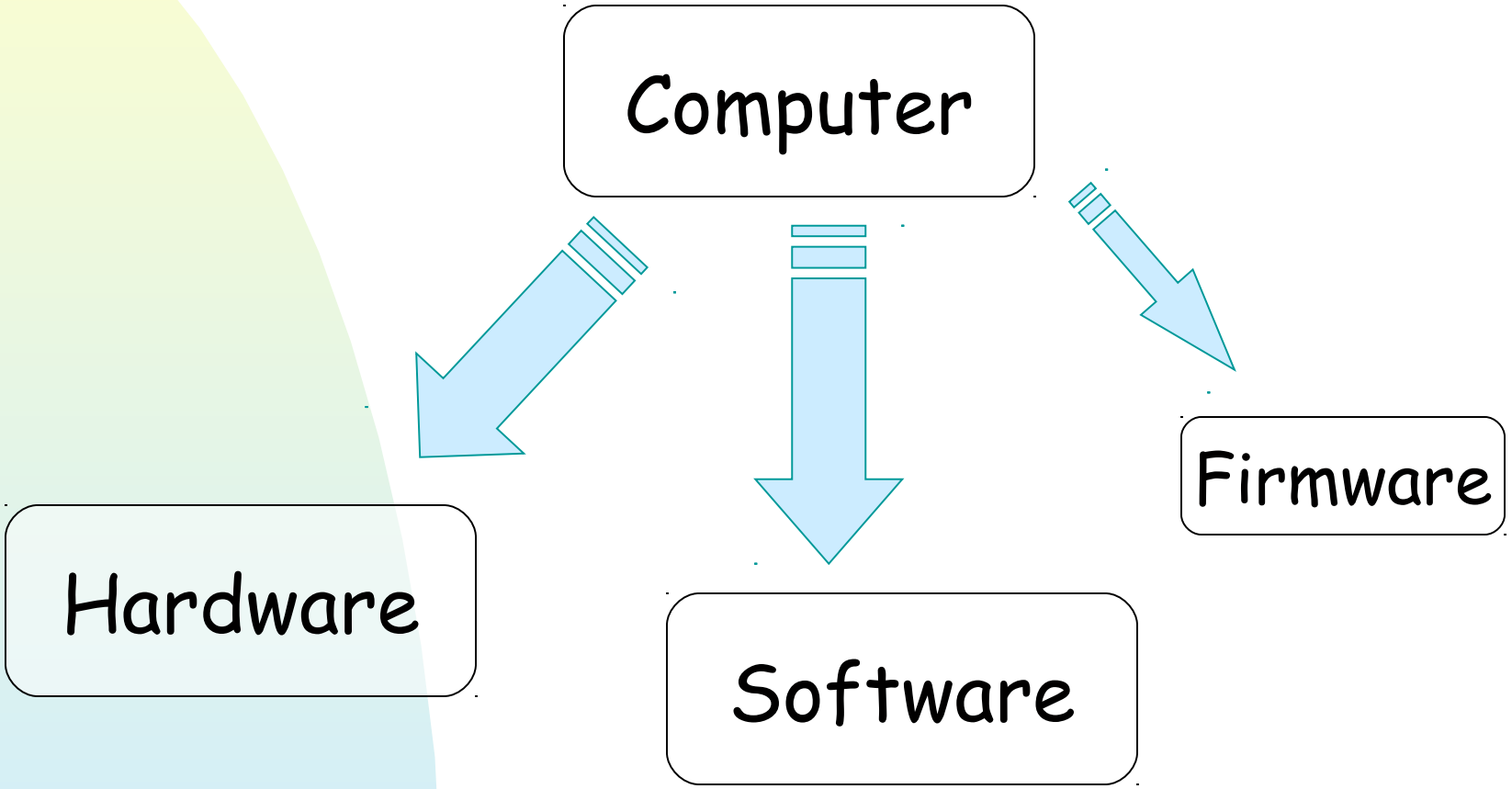
Scienza della rappresentazione e dell'elaborazione dell'informazione

Informatica

L'informatica tratta …

- l'informazione e la sua codifica
- le tecniche per raccoglierla, memorizzarla, distribuirla, trasformarla, …
- il calcolatore, il suo funzionamento, le possibilità che offre per la trasformazione dell'informazione, le tecniche di utilizzo, …
- la comunicazione tra elaboratori, tra persone (mediata dal calcolatore)

Struttura



Hardware

Hardware

Struttura fisica del calcolatore formata da parti meccaniche, elettriche, elettroniche (Video HD, tastiera, modem, CPU, ecc,)

La parte che si può prendere a calci (NON in laboratorio)

Software

Software

Componente del calcolatore costituita dai programmi di base e dai programmi applicativi per la gestione e l'uso del sistema

- Software di sistema
- Software applicativo

la parte contro cui si può solo imprecare

Firmware

Firmware

Software cablato direttamente in chip hardware non facilmente modificabile.

Es. flash Bios e successive soluzioni

la parte contro cui non si può fare proprio nulla

Elaborazione

Elaborare un'informazione significa:

Operare secondo precise modalità su di un insieme di informazioni, al fine di risolvere un determinato problema

Parliamo sempre di informazioni e non di dati

Algoritmo

Per algoritmo intendiamo:

Un testo contenente una successione ben definita di prescrizioni o istruzioni che esprime l'insieme delle azioni da compiere per poter risolvere uno specifico problema

Programma

Programma: codifica di un dato algoritmo in un opportuno linguaggio di programmazione (ossia linguaggio elaborabile da un computer

- ***Grammatica:*** uso di un particolare vocabolario
- ***Sintassi:*** correttezza formale delle "frasi" scritte nel programma, rispetto al linguaggio di programmazione scelto
- ***Semantica:*** correttezza sostanziale del programma in termini di "significato"

Un programma puo' essere corretto sintatticamente ma scorretto semanticamente (non risolve correttamente il problema per cui e' stato ideato)

Processo

Qualsiasi programma o parte di esso opportunamente tradotto in L.M. e caricato in memoria principale diventa un processo

- Processi di sistema
- Processi utente / applicativi

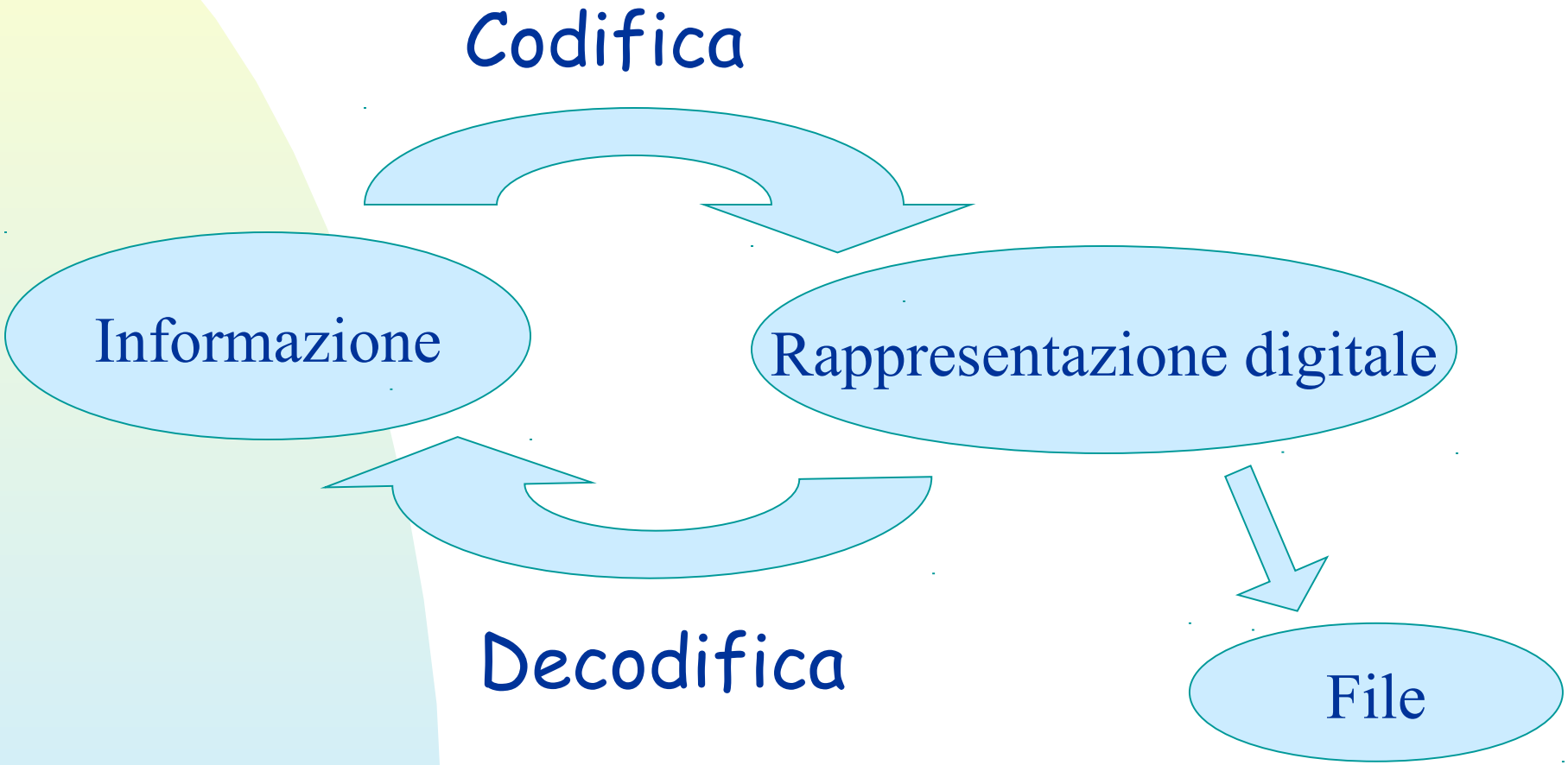
Tipi di Informazione

- Testo (caratteri)
- Numeri
- Suoni
- Immagini

Le informazioni sono concetti astratti che esistono indipendentemente dalla loro rappresentazione

Tutto ciò che ci circonda è informazione

Informazione digitale



Codifica dell'informazione

Idea di base: Usare

- presenza/assenza di carica elettrica
- passaggio/non passaggio di corrente/luce

BInary digiT (cifra binaria): il BIT

Usiamo cioe' una rappresentazione binaria (a due valori) dell'informazione

Codifica dell'informazione

Con 1 bit rappresentiamo solo 2 diverse informazioni:

si/no - on/off - 0/1 - vero/falso

Mettendo insieme piu' bit possiamo rappresentare piu' informazioni:

2 bit = 4 informazioni

00 - 01 - 10 - 11

Codifica dell'informazione

- **Esempio:** un esame può avere quattro possibili esiti
  - ottimo 00
  - discreto 01
  - sufficiente 10
  - insufficiente 11

- Con 2 bit si codificano 4 informazioni (2^2)
- Con 3 bit si codificano 8 informazioni (2^3)
- ……
- Con N bit si codificano 2^N informazioni

Codifica dell'informazione

Se dobbiamo rappresentare più di 4 informazioni occorre aggiungere il 3° bit, tante parole quanti sono i concetti

000, 001,
010, 011,
100, 101
110, 111

Codifica dell'informazione

Se dobbiamo rappresentare 21 concetti diversi mi servono

$2^N >= M$ dove $M=21$

$2^5 = 32$

Alcune sequenze (da 22 a 32) non vengono utilizzate

Codifica dell'informazione

Per rappresentare **57** informazioni diverse dobbiamo usare gruppi di almeno **6** bit.

Infatti:

$$2^6 = 64 > 57$$

Cioe' un gruppo di 6 bit puo' assumere 64 configurazioni diverse:

000000 / 000001 / 000010 …/ 111110 / 111111

Codifica dell'informazione

In generale, con N bit, ognuno dei quali puo' assumere 2 valori, possiamo rappresentare 2^N informazioni diverse

viceversa:

Per rappresentare M informazioni dobbiamo usare N bit, in modo che: $2^N >= M$

Il Byte

In informatica ha assunto particolare importanza il concetto di:

byte = 8 bit = 2^8 = 256 inf. diverse

Il byte e' usato come unita' di misura per indicare le dimensioni della memoria, le dimensioni del disco, la potenza di un elaboratore,

Usando sequenze di byte (e quindi aggregati di 8 bit di bit) si possono rappresentare caratteri, numeri immagini, suoni.

Multipli del Byte

Di solito si usano i multipli del byte

Kilo KB 2^{10} (~ mille byte, 1024)

MegaMB 2^{20} (~ un milione , 1KBx1024)

Giga GB 2^{30} (~ un miliardo, 1MBx1024)

Tera TB 2^{40} (~ mille miliardi, 1GBx1024)

Codifica dei caratteri

E' necessario individuare un codice numerico per rappresentare i caratteri.

Il codice **ASCII** (American Standard Code for Interchange Code)

usa i primi 7 bit di ogni byte:

$2^7 = 128$ **caratteri diversi**

Sufficienti per l'alfabeto anglosassone

Codifica dei caratteri

Per la codifica di 127 caratteri standard sono sufficienti 7 bit (ASCII standard) – Talvolta occorre utilizzare

- 8 bit (ASCII esteso) 256 caratteri

Oppure ancora

- 16 bit (UNICODE) 65535 caratteri
- MS Windows usa un codice proprietario a 16 bit per carattere, simile ad UNICODE ma non totalmente standard chiamato multilanguage
- 8 bit (EBCDIC) usato per i display lcd

Codifica ascii dei caratteri

ASCII	Simb.	ASCII	Simb.	ASCII	Simb.
00000000	NUL	00001110	SO	00011100	FS
00000001	SOH	00001111	SI	00011101	GS
00000010	STX	00010000	DLE	00011110	RS
00000011	ETX	00010001	DC1	00011111	US
00000100	EOT	00010010	DC2	00100000	SP
00000101	ENQ	00010011	DC3	00100001	!
00000110	ACK	00010011	DC4	00100010	"
00000111	BEL	00010101	NAK	00100011	#
00001000	BS	00010110	SYN	00100100	$
00001001	HT	00010111	ETB	00100101	%
00001010	NL	00011000	CAN	00100110	&
00001011	VT	00011001	EM	00100111	'
00001100	NP	00011010	SUB	00101000	(
00001101	CR	00011011	ESC	00101001	)

Codifica ascii dei caratteri

ASCII	Simb.	ASCII	Simb.	ASCII	Simb.
00101010	*	00111001	9	01000111	G
00101011	+	00111010	:	01001000	H
00101100	,	00111011	;	01001001	I
00101101	-	00111100	<	01001010	J
00101110	.	00111101	=	01001011	K
00101111	/	00111110	>	01001100	L
00110000	0	00111111	?	01001101	M
00110001	1	01000000	@	01001110	N
00110010	2	01000001	A	01001111	O
00110011	3	01000010	B	01010000	P

Codifica ascii dei caratteri

ASCII	Simb.	ASCII	Simb.	ASCII	Simb.
00110100	4	01000011	C	01010001	Q
00110101	5	01000100	D	01010010	R
00110110	6	01000101	E	01010011	S
00111000	8	01000110	F	01010100	T

Codifica delle parole

Sono sequenze di caratteri

- Esempio: INFORMATICA

01001001	01001110	01000110	01001111	01010010	0100110	01000001
I	N	F	O	R	M	A

01010100	01001001	01000011	01000001
T	I	C	A

Codifica delle parole

Per la lettura di un file ASCII (ad esempio un normalissimo file pippo.txt) il sistema memorizza lunghe sequenze di codici Ascii.

La decodifica consiste nella suddivisione in blocchi di 8 bit e la ricerca del simbolo grafico corrispondente

Codifica dei numeri

- Il codice ASCII consente di codificare le cifre decimali da"0" a "9" fornendo in questo modo un metodo per la rappresentazione dei numeri
- Il numero 324 può essere rappresentato dalla

sequenza di byte: 00110011 00110010 00110100

3 2 4

Ma ATTENZIONE si tratta di rappresentazione di simboli e non del valore numerico 324 da utilizzare nelle operazioni matematiche

Codifica dei numeri (decimale)

Sistema posizionale in cui ogni cifra di un numero assume un valore che dipende dalla sua posizione

Il sistema decimale (base 10):

$245 = \mathbf{2} \times 100 + \mathbf{4} \times 10 + \mathbf{5} \times 1$

$851 = \mathbf{8} \times 10^2 + \mathbf{5} \times 10^1 + \mathbf{1} \times 10^0$

*Si deve fare la somma dei prodotti di ciascuna cifra moltiplicata per la base elevata all'esponente che rappresenta la **posizione** della cifra stessa (partendo da 0)*

Codifica dei numeri (binario)

Utilizza una notazione posizionale basata su 2 cifre (0 e 1) e sulle potenze di **2**

Esempio: 10110 =

$1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = \mathbf{22}$

Esempio: 1010101 =

$1 \times 2^6 + 0 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = \mathbf{85}$

$1*64+0*32+1*16+0*8+1*4+0*2+1*1 = 85$

Codifica dei numeri (Ottale)

Utilizza una notazione posizionale basata su 8 cifre (0,1, …, 7) e sulle potenze di **8**

Esempio: 1101 =

$1 \times 8^3 + 1 \times 8^2 + 0 \times 8^1 + 1 \times 8^0 = \mathbf{577}$

$1*512 + 1*64 + 0*8 + 1*1 = 577$

Per evitare problemi si scrive la base come pedice del numero

Esempio: 11011_2 10011_8 10011_{10}

Codifica dei numeri (Esadecimale)

Utilizza una notazione posizionale basata su 16 cifre (0,1,2,…,9,A,B,C,D,E,F) e sulle potenze di **16**

Esempio: 1011_{16} =

$1x16^3 + 0x16^2 + 1x16^1 + 1x16^0 = \mathbf{4113}$

$1*4096 + 0*256 + 1*16 + 1*1 = 4113$

Esempio: C3B5 =

$12 * 16^3 + 3 * 16^2 + 11 * 16^1 + 5 * 16^0 = 50101$

12*4096 + 3*256 + 11*16 + 5*1 =

49152 + 768 + 176 + 5 = 50101

Conversioni dei numeri (da $_{10}$ a $_{2}$)

Per convertire un numero in base 2 si devono trovare i resti delle divisioni successive del numero per la base 2

Esempio: 157_{10}

157 / 2 resto 1
78 / 2 resto 0
39 / 2 resto 1
19 / 2 resto 1
9 / 2 resto 1
4 / 2 resto 0
2 / 2 resto 0
1 / 2 resto 1

10011101

Conversioni dei numeri (da $_{10}$ a $_{2}$)

Un metodo pratico

Esempio: 157_{10}

128	64	32	16	8	4	2	1
1	0	0	1	1	1	0	1
29			13	5	1		0

Conversioni dei numeri (da $_2$ a $_{10}$)

Un metodo pratico

Esempio: 10011101

128	64	32	16	8	4	2	1
1	0	0	1	1	1	0	1

128 + 16 + 8 + 4 + 1 = 157

Conversioni dei numeri (da $_2$ a $_{16}$)

Un metodo pratico

Esempio: 10011101

8	4	2	1		8	4	2	1	
1	0	0	1	|	1	1	0	1	
	9					D			= 157

Addizione binaria

```
 10111001 +    (185)
 10000100 =    (132)
 ==========
100111101      (317)
```

Overflow = oltre in numero massimo di cifre rappresentabili con un particolare numero di cifre (es Byte)

Interi con segno

111 (neg) = 001 = -1

1010 (neg) = 0110 = -6

0000	+0	**1**000	-0
0001	+1	**1**001	-1
0010	+2	**1**010	-2
0011	+3	**1**011	-3
0100	+4	**1**100	-4
0101	+5	**1**101	-5
0110	+6	**1**110	-6
0111	+7	**1**111	-7

E MI MANGIO SEMPRE UN BIT PER IL SEGNO

Complemento a 2

0000	+0		
0001	+1	**1**001	-7
0010	+2	**1**010	-6
0011	+3	**1**011	-5
0100	+4	**1**100	-4
0101	+5	**1**101	-3
0110	+6	**1**110	-2
0111	+7	**1**111	-1

Complemento a 2

- Si fissa un'ampiezza di *n* bits detta **precisione**
- Il bit piu' significativo (a sinistra) determina il segno: 0 corrisponde a +, 1 a -
- I bits restanti sono la codifica binaria se il numero e' positivo
- Se il num. e' negativo, per calcolarne il valore assoluto si trasforma la sequenza
- si parte da destra verso sinistra e la si lascia invariata fino al primo 1,
- dopodiche' si complementano i bit rimasti (scambio di 0 con 1 e viceversa): la codifica binaria risultante e' il valore assoluto.

Sottrazione binaria

Si opera con il sistema del complemento a due

```
10111001 - +    (185)
00010100 =      (20)
11101100
====================
110100101       (165)
```

Esercizi

111011	59	3B
101111101	381	17D
110000	48	30
110010101	405	195
101010101010	2730	AAA
101000000001	2561	A01

Esercizi

128	10000000	80
1234	10011010010	4D2
5654	1011000010110	1616
1233	10011010001	4D1
234	11101010	EA
511	111111111	1FF

Cifre rappresentabili

Sist. Decimale = 99…99 = $10^N - 1$

Sist. Binario= 11..11 = $2^N - 1$

Esempio: 11111111 (8 bit)= $2^8-1= 255$.

Per rappresentare il n. 256 serve un bit in più': 100000000 = $1*2^8 = 256$.

Cifre rappresentabili

Fissate quante cifre (bit) sono usate per rappresentare i numeri, si fissa anche il numero piu' grande che si puo' rappresentare

- con 16 bit: $2^{16} - 1 = 65.535$
- con 32 bit: $2^{32} - 1 = 4.294.967.295$
- con 64 bit: $2^{64} - 1 =$ circa $1,84 * 10^{19}$

Si possono rappresentare numeri piu' grandi se si tollera un certo grado di imprecisione.

Overflow

Deciso il numero di cifre a disposizione si fissa anche il **numero massimo** rappresentabile, numeri più grandi causano problemi di **overflow**

Esempio: 4 cifre

in base 10 $9999 + 1 = 10000_{10}$

in base 2 $1111 + 1 = 10000_{2}$ (= 1610)

in base 16 $FFFF + 1 = 10000_{16}$ (= 6553610)

in base 8 $7777 + 1 = 10000_{8}$ (= 409610)

Numeri a virgola mobile (floating point)

Numero 12,52 = 1252/100 = $1252 * 10^{-2}$

Un numero decimale e' rappresentato come un intero moltiplicato per una opportuna potenza di 10, cioe' con una coppia:

<1252; -2>

mantissa esponente

Numeri a virgola mobile (floating point)

Rappresentazioni a 16 bit

- 2 bit per i segni
- 9 bit per il valore assoluto della mantissa
- 5 bit per il valore assoluto dell'esponente

Rappresentazioni a 32 bit

- 2 bit per i segni
- 20 bit per il valore assoluto della mantissa
- 10 bit per il valore assoluto dell'esponente

Numeri a virgola mobile (floating point)

Con lo stesso metodo possiamo rappresentare numeri molto grandi.

Con 8 bit:

5 bit di mantissa: 11111 = 31

3 bit di esponente: 111 = 7

11111111 = $31 * 10^7$ = 310 milioni

Mentre, con la notazione classica, con 8 bit rappresentiamo al massimo il 255

Numeri a virgola mobile (floating point)

Purtroppo però non si può utilizzare sempre perche si perde in precisione

Es. 5 cifre (decimali)

4 per la mantissa, 1 per l'esponente.

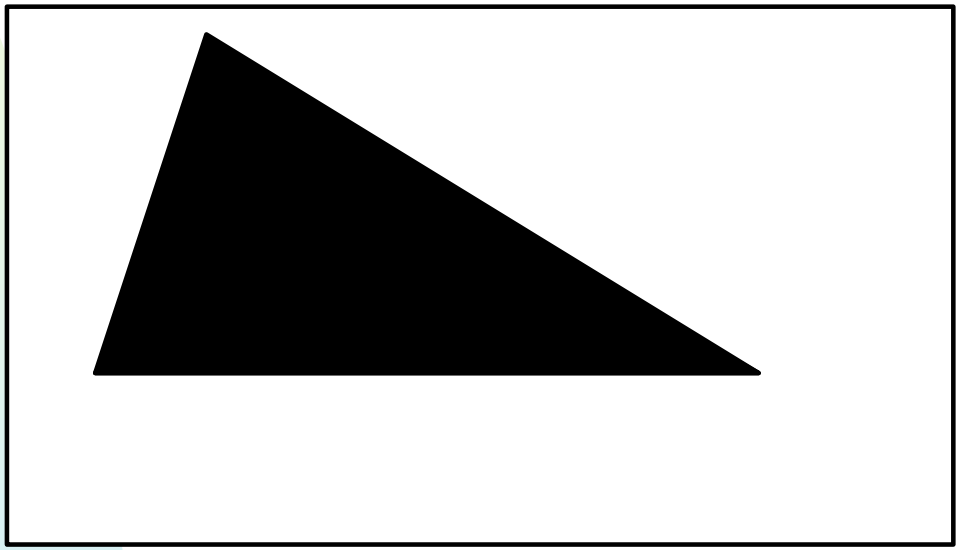
546,768

‹5467; -1› che diventa

Un numero compreso tra 546,7 e 546,8

Rappresentazione delle immagini

Immagini in bianco e nero



Rappresentazione delle immagini

Dividere l'immagine in una griglia a righe orizzontali e verticali

Ogni quadratino della griglia e' un **pixel**

Codificare ogni pixel con:

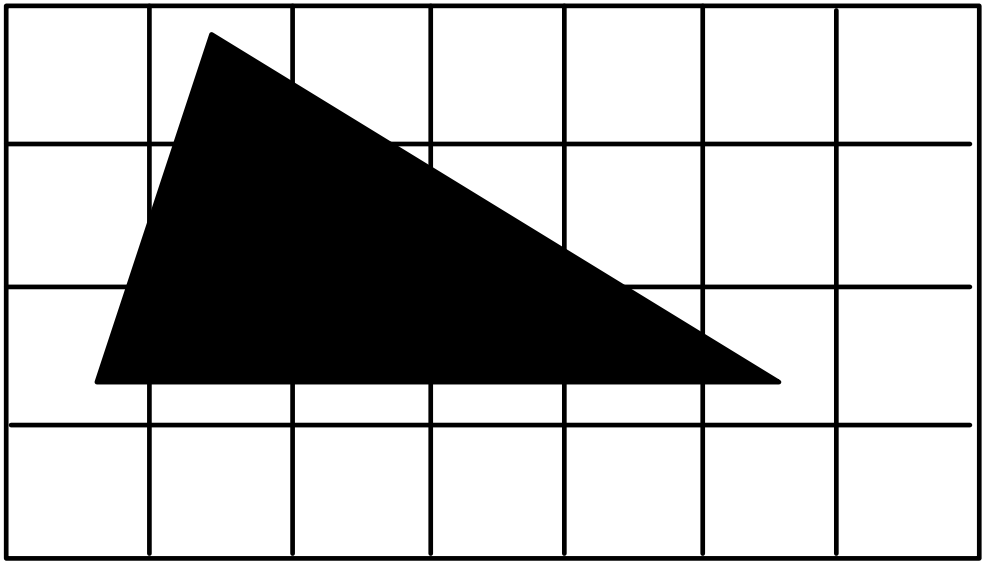
- 0 se il pixel e' bianco
- 1 se il pixel e' nero

Il formato dell'immagine (BMP, Jpeg, Tiff, ecc.) determina **anche** la partenza:

convenzionalmente si parte dal 1° quadratino basso a sinistra.

Rappresentazione delle immagini

Risoluzione 7 X 4



Rappresentazione delle immagini

0000000 0111100 0110000 0100000

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

Risoluzione in bit B/W

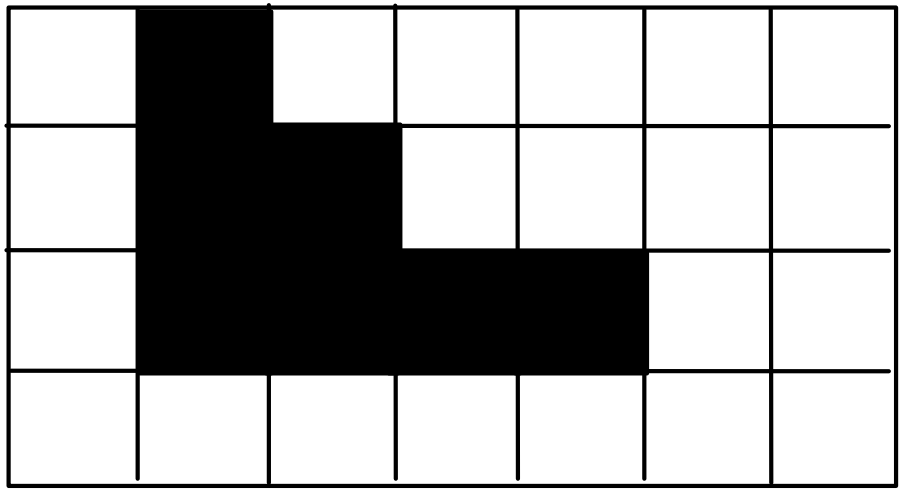
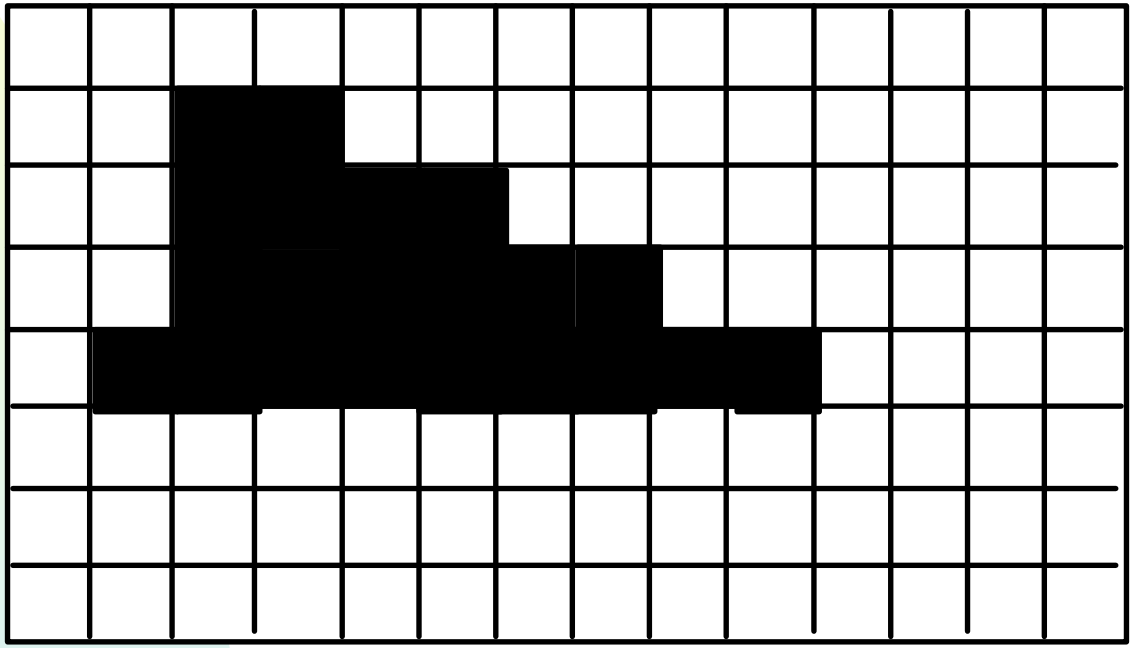


Immagine ricostruita

Rappresentazione delle immagini



Aumentiamo la risoluzione a 14 X 8

Rappresentazione delle immagini

Quindi: le immagini sono rappresentate con un certo livello di approssimazione, o meglio, di **risoluzione**, ossia il numero di pixel usati per riprodurre l'immagine:

- 640 x 480 pixel; 800 x 600 pixel
- 1024 x 768 pixel; 1280 x 1024 pixel
- 1400 x 1280 pixel

É importante anche il **dot pitch**, il grado di definizione del pixel: 0,25 - 0,28 in quanto maggior risoluzione e minor dot-pitch garantiscono immagini migliori

Rappresentazione delle immagini

- Esistono immagini in B/W (1 solo bit per il colore)
- Oppure immagini a livelli di grigio o colori
- Con 8 bit si codificano 256 **livelli di grigio**. (ottenuti regolando la luminosita' del pixel)
- Con 8 bit si rappresentano 256 colori, con 16 bit 64.000 colori, con 24 bit 16 milioni di colori diversi per pixel e con 32 bit sempre 16 milioni di colori in true color

Attenzione : fare sempre attenzione nel calcolo

- a bit e byte 256 colori = 1 byte per pixel

Rappresentazione delle immagini

1 pixel a	2 colori	1 bit
1 pixel a	256 colori	1 byte (1 * 8 bit)
1 pixel a	65535 colori	2 byte (2 * 8 bit)
1 pixel a	16 Mil. di colori	3 byte (3 * 8 bit)

La compressione delle immagini

- **Compressione senza perdita di dati (lossless compression).**

Aree contenenti pixel dello stesso colore vengono codificate in modo compatto; questo permette una compressione limitata ma salvaguarda dalla perdita di informazioni nella fase di decompressione.

- **Compressione con perdita di dati (lossy compression).**

Questa famiglia di tecniche di compressione permette una riduzione nelle dimensioni dell'immagine compressa fino anche a dieci volte rispetto allo schema precedente ma comporta una perdita di informazioni.

La compressione delle immagini

BITMAP (raw): la rappresentazione di una immagine con codifica dei pixel.

Porta via molto spazio:

(immagine 640 x 480 x 32 = Byte)

BITMAP (RLE) – compressa (lossless)

RLE (**R**un **L**ength **E**ncoding)

Esistono vari formati di codifica:

(giff, jpg, pcx, tiff, ecc.)

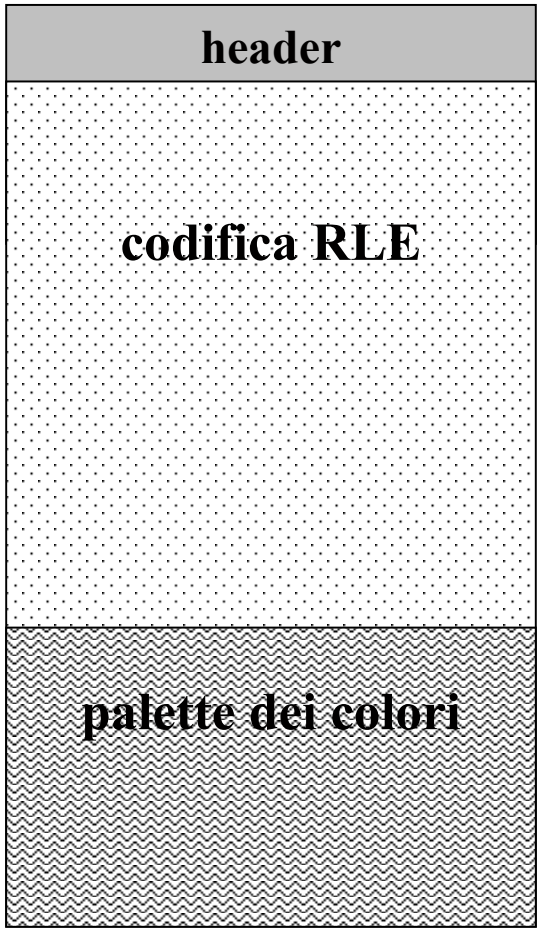
con diversi fattori di compressione in base alla tecnica utilizzata.

La compressione delle immagini

La struttura di un file PCX

lossless

con riduzione di spazio del 30-40%



La compressione delle immagini

Altri protocolli di conversione

GIF (**G**raphics **I**nterchange **F**ormat), GIF87a o GIF89a

Usa l'algoritmo LZW - tipico di Internet, immagini *interlacciate*, sequenze di caricamento ma limitato a 256 colori.

JPEG (**J**oint **P**hotographic **E**xperts **G**roup)
JPEG–LS (lossless)

PNG (**P**ortable **N**etwork **G**raphics) lossy

TIFF è strutturato a blocchi chiamati tag

Le immagini vettoriali

- Se le immagini sono regolari si può usare una codifica di tipo **vettoriale** in cui non si specificano le informazioni di colore dei singoli pixel ma ogni elemento geometrico primitivo viene specificato individualmente
- Le immagini vengono costruite a partire dalla descrizione degli elementi che le compongono mediante un linguaggio testuale
- Spesso occupano meno spazio rispetto alle immagini bitmap

La codifica dei filmati

Sono sequenze di immagini **compresse**: (ad esempio si possono registrare solo le variazioni tra un fotogramma e l'altro)

Utilizzano tutti la tecnica del CODEC (COmpress od DECompress)

- Esistono vari formati per codificare filmato e sonoro :
  - mpeg (il piu' usato)
  - avi (microsoft)
  - quicktime (apple) – mov

La codifica dei filmati

CODEC (COmpress od DECompress)
Programmi per la codifica – decodifica flusso dati

MPEG – (Moving Picture Experts Group) fino a 200:1

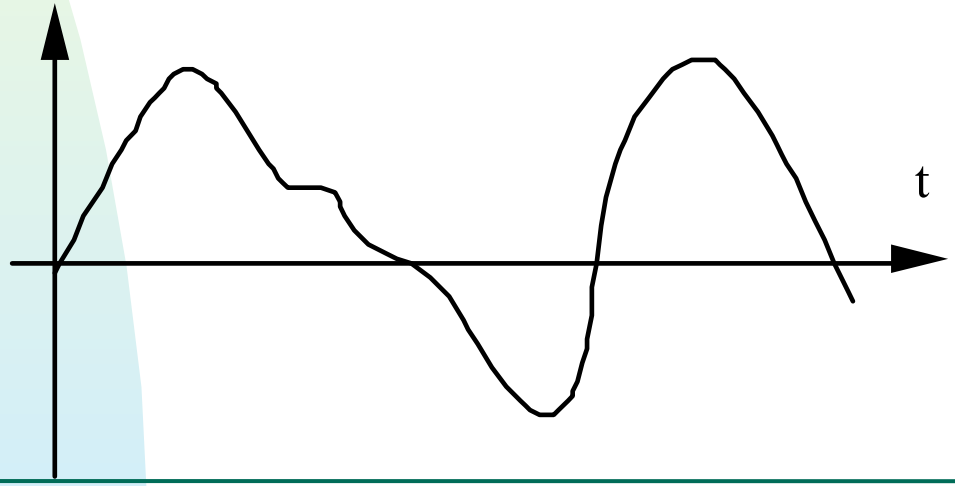
DIVX

MOTION JPEG (Max 100:1)

Il suono

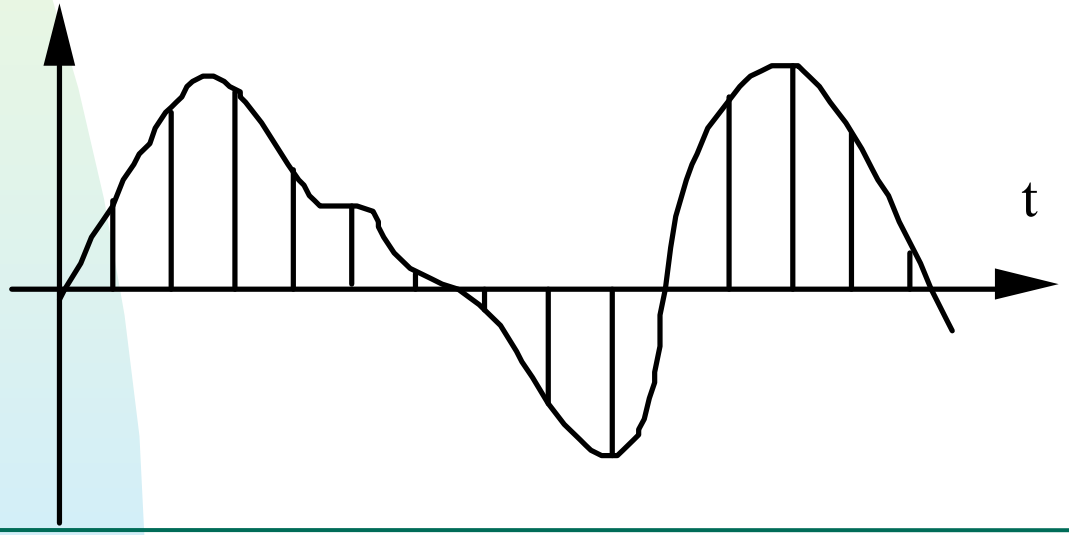
Fisicamente un suono è rappresentato come un'onda che descrive la variazione della pressione dell'aria nel tempo (onda sonora)

Sull'asse delle ascisse viene posto il tempo t e sull'asse delle ordinate la variazione della pressione corrispondente.



Il campionamento dei suoni

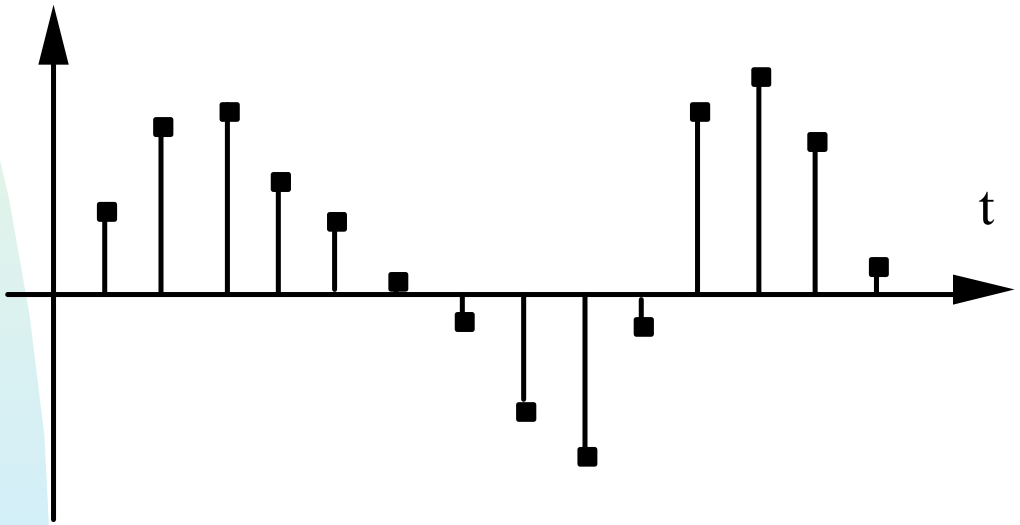
- Si effettuano dei campionamenti sull'onda (cioè si misura il valore dell'onda ad intervalli di tempo costanti) e si codificano in forma digitale le informazioni estratte da tali campionamenti.
- La sequenza dei valori numerici ottenuta dai campioni può essere facilmente codificata



Il campionamento dei suoni

- Quanto più frequentemente il valore dell'onda viene campionato, tanto più precisa sarà la sua rappresentazione

- Il numero di campioni raccolti per ogni secondo definisce la frequenza di campionamento che si misura in Hertz (Hz)



Il campionamento dei suoni

L'onda sonora viene misurata (campionata) ad intervalli regolari

Minore e l'intervallo di campionamento e maggiore e la qualita' del suono

La risoluzione standard è a 16 bit

Es. I CD musicali: 44000 campionamenti al secondo, 16 bit per campione.

La codifica dei suoni

Alcuni formati:

- .aiff
- .wav (wave)
- .mpeg
- .cda
- . MP3
- **midi** usato per l'elaborazione della musica al computer

Archiviare le informazioni

Tutte le informazioni digitalizzate vengono archiviate in sequenze di bit

Sequenze di bit più o meno lunghe si chiamano **file** identificati da un nome di file e un'estensione.

pippo.doc

Convenzionalmente l'estensione identifica il formato di aggregazione (non sempre)

FILE

il cognome
il nome
l'indirizzo
il numero di conto
la disponibilità sul conto
la foto.

Il campo

Campo:

***Un campo è costituto da un insieme di byte* e serve per codificare una singola informazione che può essere:**

- *numerica*: si parla in questo caso di campo numerico, costituito da un insieme di byte per la codifica dei numeri;

- *alfabetica (alfanumerica)*: si parla di campo alfabetico (alfanumerico), costituito da una sequenza di byte che codificano i caratteri della stringa alfabetica (alfanumerica);

- un' *immagine* o un *suono*.

Il record

Un record è costituito da un insieme di campi che sono logicamente correlati tra di loro e che costituiscono, nel loro insieme, un'informazione complessa.

- un *campo alfabetico* per il nome;
- un *campo alfabetico* per il cognome;
- un *campo alfanumerico* per l'indirizzo;
- un *campo numerico* per il numero del conto;
- un *campo numerico* per la disponibilità sul conto;
- un *campo immagine* per la foto (tale campo conterrà la codifica digitale dell'immagine).

Il file

***Un file** è costituito da una sequenza di record.* Le informazioni riguardanti la banca possono essere viste come un file costituito da un insieme di record, uno per ogni cliente.

-***file strutturati*** che sono effettivamente sequenze di record;

-***file non strutturati*** in cui l'informazione è semplicemente una sequenza di caratteri (byte) che costituiscono un testo; si può pensare che il record coincida con il singolo carattere e non sia quindi possibile suddividerlo ulteriormente in campi.

File con record a lunghezza costante

File con *record a lunghezza costante*

114561	51352,50	Massimiliano	Rossi	Via Milano 151
271564	231474,32	Cristina	Bianchi	Corso Venezia 1
102574	567899,80	Elena	Rossi	Via Milano 151
143256	21452,34	Ada	Bo	Via Po 1

File con record a lunghezza variabile

File con *record a dimensione variabile*

114561	51352,50	Massimiliano	Rossi	Via Milano 151
271564	231474,32	Cristina	Bianchi	Corso Venezia 1
102574	567899,80	Elena	Rossi	Via Milano 151
143256	21452,34	Ada	Bo	Via Po 1

File con record a lunghezza variabile

...&1146561#51352,50#Massimiliano#Rossi#ViaMilano151&271564#231474,32#Cristina#Bianchi#Corso Venezia 1&...

Servono due caratteri speciali:

- # fine campo
- & inizio/fine record

Accesso ai dati – accesso sequenziale

File a dimensione costante – *si avanza di un numero di byte prestabilito. Il record successivo inizia al termine della somma di tutti i campi.*

File a dimensione variabile – *ci posizioniamo sul punto di inizio del primo record (segnalato dal carattere "&") e cominciamo a leggere i vari caratteri fino al terzo simbolo "#"*

L'accesso sequenziale non è efficiente perché ogni volta la ricerca viene effettuata a partire dall'inizio del file.

Accesso ai dati – accesso diretto

Il punto di inizio di un record (campo) all'interno di un file prende il nome di **indirizzo** del record (campo).

Poiché un file è, in ultima analisi, un insieme di byte, possiamo definire come indirizzo di un byte il numero che corrisponde alla posizione del byte nella sequenza (per cui il primo byte ha indirizzo 0, il secondo ha indirizzo 1 e così via); l'indirizzo di un campo è quindi l'indirizzo del primo byte del campo e l'indirizzo di un record è l'indirizzo del primo campo del record.

Recuperare il record in *n*-esima posizione richiede il posizionamento al byte che si trova in posizione 50*(n-1) do 50 è la lunghezza ipotetica del record in byte.

L'accesso diretto ai record non è possibile con file a lunghezza variabile perché in questo caso non è possibile calcolare l'indirizzo.

Accesso ai dati – accesso con chiave

Per poter realizzare l'accesso in modo veloce si mantengono le informazioni sugli indirizzi al di fuori del file.

Campo chiave : il campo chiave ci permette di individuare univocamente tutti i record del file.

Una volta scelto il campo chiave si costruisce un nuovo file che prende il nome di **file delle chiavi** (o **tabella delle chiavi**) ed è costituito da record a dimensione costante formati da due campi:

- un campo per contenere il **valore della chiave**
- un campo per contenere **l'indirizzo del record** all'interno del file.
- il file delle chiavi ha **tanti record quanti sono i record** del file originario.

File delle chiavi

chiave	*indirizzo del record*
102574	222
114561	135
143256	259
271564	189

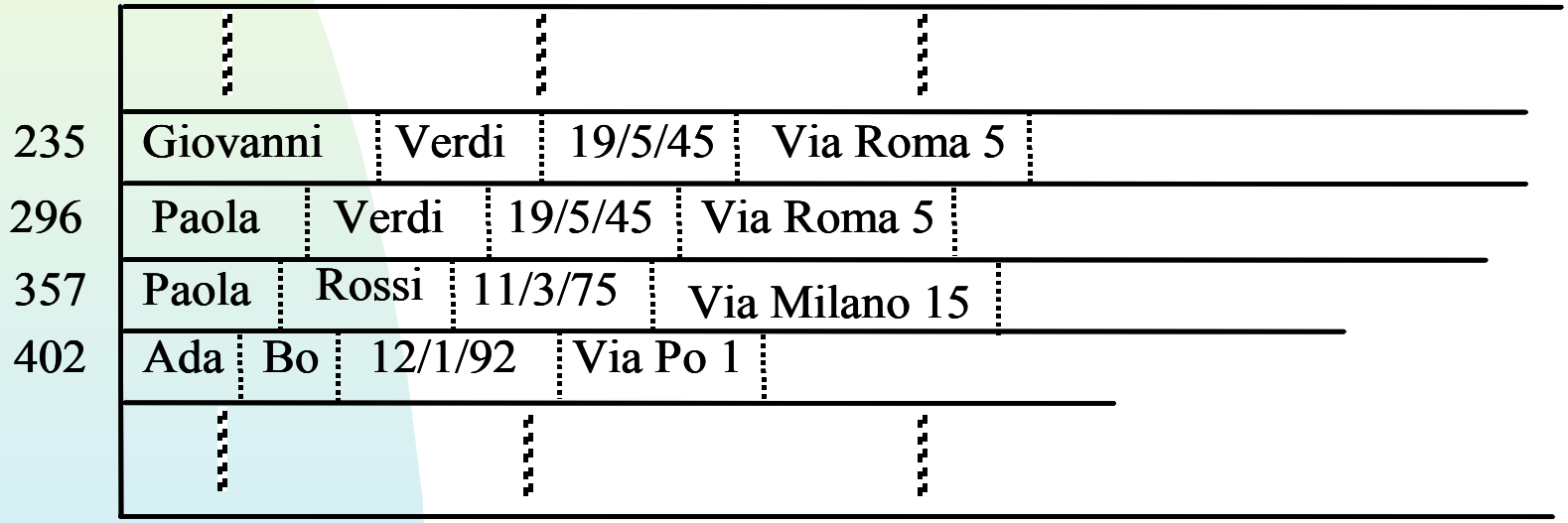
File delle informazioni

135	114561	51352,50	Massimiliano	Rossi	Via Milano 151
189	271564	231474,32	Cristina	Bianchi	Corso Venezia 1
222	102574	567899,80	Elena	Rossi	Via Milano 151
259	143256	21452,34	Ada	Bo	Via Po 1

File delle chiavi

chiave		*indirizzo del record*
Bo	Ada	402
Rossi	Paola	357
Verdi	Giovanni	235
Verdi	Paola	296

File delle informazioni



Strategie di ricerca: SEQUENZIALE

Occorre definire delle **strategie di ricerca** efficienti all'interno del *file delle chiavi*, per poi procedere con accesso diretto nella *tabella informazioni*.

La ricerca più semplice è quella ***sequenziale***: si tratta cioè di lettura un accesso sequenziale alla tabella delle chiavi, fino a quando non si trova il record con la chiave desiderata.

Ovviamente nessun vantaggio

Strategie di ricerca : BINARIA

Indispensabile mantenere le chiavi **ordinate**.

Si prende il record che si trova a metà nella tabella e si legge il valore della chiave. Sia k' il valore della chiave che si trova in questa posizione:

1 se k'=k, ossia l'elemento cercato è proprio quello di metà, allora abbiamo finito e abbiamo trovato l'elemento che stiamo cercando.

2 se k<k' allora l'elemento cercato è nella prima metà della tabella (poiché la tabella è ordinata) e si può proseguire la ricerca ripartendo dall'inizio ma considerando la tabella che contiene solo i primi N/2 elementi.

3 se k>k' allora l'elemento cercato è nella seconda metà della tabella e si può proseguire la ricerca ripartendo dall'inizio ma considerando la tabella che contiene solo gli elementi da N/2 fino alla fine.

Strategie di ricerca : BINARIA

Indispensabile mantenere le chiavi **ordinate**.

chiave	*indirizzo del record*
Bianchi	402
Bruni	296
Gialli	98
Rosa	701
Rossi	546
Russo	357
Verdi	235

Esercizi 1

-Abbiamo un'immagine 10 X 20 pixel con 200 colori

- -Quanti BIT occupa ?
- -Quanti bit occuperebbe se raddoppiassimo i colori ?
- -Quanti bit occuperebbe se raddoppiassimo i pixel ?

Esercizi 1s

Per 200 colori occorrono 8 bit (256)

Quindi 10 X 20 = 200 pixel

- Per ogni pixel 8 bit, quindi 200 X 8 = **1600 bit**

-Se raddoppio i colori (da 200 a 400) occorrono 9 bit (512)

-Quindi 200 X 9 = **1800 bit** (quindi **non** raddoppia lo spazio occupato)

- Se raddoppio i pixel (da 200 a 400)

allora 400 X 8 = 3200 bit (raddoppia lo spazio)

Esercizi 2

-10 secondi di suono campionati a 16 bit occupano 2000 byte

-Quale la frequenza di campionamento ?

-E se raddoppia la frequenza quale occupazione in bit ?

Esercizi 2s

2000 byte = 16000 bit (2000*8) in 10 secondi

In un secondo 1600 bit (16000/10)

- Se usiamo 16 bit per campione la frequenza è di **100 Hz**

-Se la frequenza diventa 200 Hz allora

200 * 16 = 3200

3200 * 10 = 32000 / 8 = **4000 byte**

Esercizi 3

Un' immagine 20 * 30 pixel occupa 2400 bit.

- Quanti i colori possibili ?

Esercizi 3 s

Totale pixel 20 *30 = 600 pixel

2400 bit diviso i 600 pixel = 4

Quindi 4 bit per pixel che contengono max **16 colori**

Esercizi 4

Abbiamo 5 secondi di suono campionati ad una frequenza di 20 Hz. Con campioni da 2 Byte.

- Quale occupazione in bit

- Se i secondi raddoppiano e la frequenza si dimezza quale occupazione ?

Esercizi 4s

- 5 secondi * 20 Hz = 100 campioni

100 * 16 = 1600 bit

Oppure

16 * 5 = 80 e poi 80 * 20 = 1600 il risultato è il medesimo.

- 10 secondi * 10 Hz = 100 campioni – Non cambia il risultato finale