



La codifica delle informazioni

Con le cifre binarie si può utilizzare un codice, cioè un insieme di simboli e di regole per rappresentare informazioni.

Esempio:

Per rappresentare le 26 lettere dell'alfabeto serve un codice di quanti bit?



La codifica delle informazioni (2)

Con **1 bit** posso codificare solo 2 lettere

0 oppure 1

2^1 possibili codici = 2

Con **2 bit** posso codificare solo 4 lettere

00

01

10

11

2^2 possibili codici = 4

La codifica delle informazioni (3)

Con **3 bit** posso codificare solo 8 lettere

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

2^3 possibili codici = 8

Con **4 bit** posso codificare solo 16 lettere

2^4 possibili codici = 16

Con **5 bit** posso codificare 32 lettere

2^5 possibili codici = 32

Quindi la risposta è un codice a 5 bit!



Esercizio:

Nell'alfabeto di Marte sono previsti 400 simboli; quanti bit si devono utilizzare per rappresentarli tutti?

RISPOSTA

Si devono utilizzare **9 BIT**

SPIEGAZIONE:

8 bit non sarebbero sufficienti perché 2^8 risulta uguale a 256 possibili codici ,
Provo con 9 bit ed ottengo
 $2^9 = 512$ possibili codici.



**Dati 12 bit per la codifica, quante
informazioni distinte puoi
rappresentare?**

RISPOSTA

4096 informazioni

SPIEGAZIONE

$$2^{12}=4096$$



IL CODICE ASCII

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

E' uno dei codici più diffusi per la
rappresentazione dei caratteri alfanumerici

Il codice ASCII è un codice a 8 bit

Permette la codifica di 2^8 simboli = 256

| Byte | Cod. | Char | Byte | Cod. | Char | Byte | Cod. | Char | Byte | Cod. | Char |
|----------|------|------------------|----------|------|------|----------|------|------|----------|------|------|
| 00000000 | 0 | Null | 00100000 | 32 | Spc | 01000000 | 64 | @ | 01100000 | 96 | ` |
| 00000001 | 1 | Start of heading | 00100001 | 33 | ! | 01000001 | 65 | A | 01100001 | 97 | a |
| 00000010 | 2 | Start of text | 00100010 | 34 | " | 01000010 | 66 | B | 01100010 | 98 | b |
| 00000011 | 3 | End of text | 00100011 | 35 | # | 01000011 | 67 | C | 01100011 | 99 | c |
| 00000100 | 4 | End of transmit | 00100100 | 36 | \$ | 01000100 | 68 | D | 01100100 | 100 | d |
| 00000101 | 5 | Enquiry | 00100101 | 37 | % | 01000101 | 69 | E | 01100101 | 101 | e |
| 00000110 | 6 | Acknowledge | 00100110 | 38 | & | 01000110 | 70 | F | 01100110 | 102 | f |
| 00000111 | 7 | Audible bell | 00100111 | 39 | ' | 01000111 | 71 | G | 01100111 | 103 | g |
| 00001000 | 8 | Backspace | 00101000 | 40 | (| 01001000 | 72 | H | 01101000 | 104 | h |
| 00001001 | 9 | Horizontal tab | 00101001 | 41 |) | 01001001 | 73 | I | 01101001 | 105 | i |
| 00001010 | 10 | Line feed | 00101010 | 42 | * | 01001010 | 74 | J | 01101010 | 106 | j |
| 00001011 | 11 | Vertical tab | 00101011 | 43 | + | 01001011 | 75 | K | 01101011 | 107 | k |
| 00001100 | 12 | Form Feed | 00101100 | 44 | , | 01001100 | 76 | L | 01101100 | 108 | l |
| 00001101 | 13 | Carriage return | 00101101 | 45 | - | 01001101 | 77 | M | 01101101 | 109 | m |
| 00001110 | 14 | Shift out | 00101110 | 46 | . | 01001110 | 78 | N | 01101110 | 110 | n |
| 00001111 | 15 | Shift in | 00101111 | 47 | / | 01001111 | 79 | O | 01101111 | 111 | o |
| 00010000 | 16 | Data link escape | 00110000 | 48 | 0 | 01010000 | 80 | P | 01110000 | 112 | p |
| 00010001 | 17 | Device control 1 | 00110001 | 49 | 1 | 01010001 | 81 | Q | 01110001 | 113 | q |
| 00010010 | 18 | Device control 2 | 00110010 | 50 | 2 | 01010010 | 82 | R | 01110010 | 114 | r |
| 00010011 | 19 | Device control 3 | 00110011 | 51 | 3 | 01010011 | 83 | S | 01110011 | 115 | s |
| 00010100 | 20 | Device control 4 | 00110100 | 52 | 4 | 01010100 | 84 | T | 01110100 | 116 | t |
| 00010101 | 21 | Neg. acknowledge | 00110101 | 53 | 5 | 01010101 | 85 | U | 01110101 | 117 | u |
| 00010110 | 22 | Synchronous idle | 00110110 | 54 | 6 | 01010110 | 86 | V | 01110110 | 118 | v |
| 00010111 | 23 | End trans. block | 00110111 | 55 | 7 | 01010111 | 87 | W | 01110111 | 119 | w |
| 00011000 | 24 | Cancel | 00111000 | 56 | 8 | 01011000 | 88 | X | 01111000 | 120 | x |
| 00011001 | 25 | End of medium | 00111001 | 57 | 9 | 01011001 | 89 | Y | 01111001 | 121 | y |
| 00011010 | 26 | Substitution | 00111010 | 58 | : | 01011010 | 90 | Z | 01111010 | 122 | z |
| 00011011 | 27 | Escape | 00111011 | 59 | ; | 01011011 | 91 | [| 01111011 | 123 | { |
| 00011100 | 28 | File separator | 00111100 | 60 | < | 01011100 | 92 | \ | 01111100 | 124 | |
| 00011101 | 29 | Group separator | 00111101 | 61 | = | 01011101 | 93 |] | 01111101 | 125 | } |
| 00011110 | 30 | Record Separator | 00111110 | 62 | > | 01011110 | 94 | ^ | 01111110 | 126 | ~ |
| 00011111 | 31 | Unit separator | 00111111 | 63 | ? | 01011111 | 95 | _ | 01111111 | 127 | Del |



La codifica delle immagini

Immagine digitale:

insieme di pixel disposti in modo ordinato secondo una griglia costituita da righe e colonne

PIXEL (Picture Elements)

Puntini luminosi e colorati che costituiscono gli elementi fondamentali dell'immagine digitale

Tipi di immagine digitale

IMMAGINE BINARIA

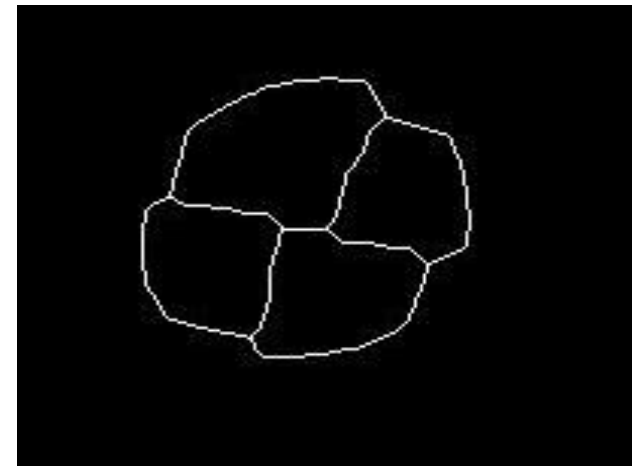
Ogni pixel viene visto come bianco o nero.

Vengono trascurate le altre sfumature.

1 pixel occupa 1 bit di memoria

(0 oppure 1

cioè bianco o nero)



Tipi di immagine digitale (2)

IMMAGINE IN SCALA DI GRIGIO

Ogni pixel corrisponde ad una tonalità di grigio che viene codificata con un certo numero di bit, (da 2 a 8 bit).

Se ogni pixel corrisponde a 4 bit avrò 2^4 possibili tonalità di grigio cioè 16!



Tipi di immagine digitale (3)

IMMAGINE A PSEUDOCOLORI

In questo tipo di immagine si possono avere da 4 a 256 differenti colori ma non è possibile avere tutte le sfumature



IMMAGINE A COLORI

Ogni pixel corrisponde a 3 byte cioè 24 bit!