CODIFICA BINARIA

Cosa vuol dire **binario**?

Binario ad esempio del treno? Due rotaie su cui viaggia il treno.

Binario, Bicicletta, Bipede, c'è sempre di mezzo il 2.

Il computer capisce solo ACCESO o SPENTO, di un circuito. Questa minima informazione si chiama BiT (**bi**nary digi**t**).

Per capire per noi essere umani, diciamo 0 (spento) e 1 (acceso).

Questo è l'alfabeto del computer, ha solo due 'lettere': 0 e 1.

Consideriamo invece il nostro alfabeto: a, b, c.....z, almeno 26 lettere, in più le maiuscole, A, B, ...Z.

Ma il computer è limitato a 0 e 1.

Dalla combinazione di 0 e 1 deve esprimere magari la parola 'cane' o 'gatto'.

Come fare??

Come dall'alfabeto italiano, combinando le lettere posso esprimere, nomi, verbi o aggettivi così con l'alfabeto del computer (binario, solo 0 e 1) posso esprimere le parole.

Se io volessi esprimere tutte le lettere dell'alfabeto con una sequenza di 0 e 1 come potrei fare??

Facciamo, all'inizio per semplificare che dobbiamo esprimere solo la 26 lettere miniscole: a, b, c.....z

Ragioniamo: con **1 bit** possimo esprimere solo 2 informazioni: vero/falso ad esempio. Equivale $2^1 = 2$

Con 2 bit, già possiamo esprimere di più: $2^2 = 2 * 2 = 4$ informazioni.

Infatti, abbiamo queste possibili informazioni:

- 0 0
- 0 1
- 1 0
- 1 1

andando così avanti con 3 bit abbiamo $2^3 = 2 * 2 * 2 = 8$ diverse possibilità:

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

Con 3 bit ad esempio potremmo esprimere tutti i giorni della settimana (sono 7 e noi con 3 bit abbiamo 8 combinazioni).
Ad esempio potremmo codificate 000 come LUNEDI', 001 come MARTEDI' fino a 110 come DOMENICA (avanzeremmo anche la sequenza 111 se i giorni della settimana fossero 8).

Se qualcun'altro conoscesse il nostro codice segreto (ci saiamo prima messi d'accordo) e scrivessimo 010, l'altra persona leggerebbe MERCOLEDI', avremmo creato un nostro linguaggio. Da notare che **è una convenzione stabilita tra chi scrive e chi legge**.

Tornando a bomba, se dobbiamo invece esprimere 26 lettere alfabeto quanti bit ci servono se a ogni combinazione gli possiamo assegnare una lettera dell'alfabeto?

Deve essere 2^x > di 26, con l'X, l'intero, minore possibile, che soddisfi l'equazione: vediamo che con $2^3 = 8$ (< di 26), $2^4 = 16$ (< di 26) ma $2^5 = 32$ che è > di 26.

Quindi con 5 bit possiamo esprimere tutte le lettere dell'alfabeto. Stabiliamo ad esempio:

- 00000 -> *A*
- 00001 -> *B*
- 00010 -> *C*
- 00011 -> *D*
- 00100 -> *E*
- ..... fino a *Z*

come già accennato, dobbiamo stabilire con chi legge la convenzione di questa codifica, se no chi legge non capisce nulla: se uno legge *00010* e non conosce la codifica da noi stabilita, non può sapere che si voleva significare C.

Chi legge deve avere come una tabella che associa alla sequenza **00000** la lettera ***A***, e così via. E chi scrive deve conoscere la tabella e se vuole significare ***B*** deve scrivere **00001**.

Tutto questo poi perchè l'**alfabeto dei computer è fatto di sole due lettere 0 e 1**.

CODICE ASCII

I pionieri nel campo dell'informatica, parliamo dei primi anni 60 del secolo scorso, si sono travati di fronte bene o male al problema che abbiamo descritto noi, un po' esteso: dovevano codificare i la lettere dell'alfabeto (maiscole e minuscole), le cifre dei numeri (0, 1, 2....9) e tutti i segni di punteggiatura (virgola, parentesi, duepunti, ?, !....) e vari simboli, come ><. Il tutto per 128 caratteri: 2^7 che equivale a 128, quindi con 7 bit si potrebbero codificare tutti i 128 simboli. Si è scelto di usare 8 bit, di avere altri 128 posti a disposizione per codificare altri caratteri speciali, specifici di ogni alfabeto dei paesi occidentali (nelle slide 9 c'è la tabella ASCII STANDARD, i primi 128 caratteri comuni a tutti e slide 10, gli altri 128 dell'ASCII Esteso).

Le lettere dell'alfabeto della tabella ASCII sono quelle dell'alfabero Inglese (non cisono ad esempio le nostre 'à' o 'ò' ad esempio). Gli bastavano 128

caratteri e quindi 7 bit, ma un altro bit (l'ottavo) è stato utilizzato per codificare le lettere di altri alfabeti, che avevano dei propri caratteri speciali, come le a accentate dell'italiano, o caratteri speciali del francese o tedesco, per un totale di 256 caratteri (di qui gli 8 bit necessari = 2^8 = 256). I primi 128 comuni a tutti (con il primo bit a 0), gli altri 128 (quelli con il primo bit = 1) specifici per ogni alfabeto dei paesi occidentali (ITALIANO, TEDESCO, FRANCESE, OLANDESE ....).

Per il fatto di utilizzare 8 bit come base per codificare i caratteri, ha fatto si che l'insieme di 8 bit, chiamato **Byte**, sia il formato l'unità di base per l'architettura del computer.

-> **Byte** sta per "**BinarY octetTE**", ovvero "ottetto binario".

Come potete vedere ad ogni carattere corrisponde un byte

Se devo scrivere "gatto", diventa:

01100111	**01100001**	**01110100**	**01110100**	**01101111**
g	a	t	t	o

Questo è quello che viene salvato su disco fisso, cioè la sequenza di 0 e 1 di 5 byte.

Quando un programma come world, legge un file da disco fisso, legge la sequenza di 0 e 1, e semplicisticamente possiamo dire che li raggruppa in sequenze di 8 bit, cioè di byte, e converte queste parole di 0 e 1 nel corrispettivo carattere che viene poi presentato a video. Ad esempio, la sequenza di bit (4 byte, 32 bit):

01000011 01100001 01101110 01100101 -> **C a n e**

viene decodificata nella stringa, '*Cane*'.

Per un po' di anni, tutto questo è bastato, poi con la difffusione dell'informatica in tutti i paesi e quindi non solo quelli iniziali (principalmente Stati Uniti e altri paesi dell'Europa Occidentale) ma in tutti i continenti, l'alfabeto dello **standard ASCII** (che sta infatti per *American Standard Code for Information Interchange*) non basta più: non sono codificate le lettere ad esempio dell'alfabeto arabo o quelle del cinese.

Con un byte come abbiamo già detto codificare solo, 2^8, 256 possibili carateri, oramai troppo pochi.

Un ente di standarizzazione internazionale si occupa attualmente di stardardizzare tutti i caratteri dei vari alfabeti e si chiama UNICODE e utilizza 2 byte (cioè 16 bit e quindi 2^{16} possibili caratteri 65.536 (2^16).

E’ importante capire che i codici UNICODE sono un’estensione della tabella ASCII, i primi 256 caratteri della tabella ASCII sono uguali a quelli della tabella UNICODE.

Codice ASCII *STANDARD*

Byte	Cod.	Char
00000000	0	Null
00000001	1	Start of heading
00000010	2	Start of text
00000011	3	End of text
00000100	4	End of transmit
00000101	5	Enquiry
00000110	6	Acknowledge
00000111	7	Audible bell
00001000	8	Backspace
00001001	9	Horizontal tab
00001010	10	Line feed
00001011	11	Vertical tab
00001100	12	Form Feed
00001101	13	Carriage return
00001110	14	Shift out
00001111	15	Shift in
00010000	16	Data link escape
00010001	17	Device control 1
00010010	18	Device control 2
00010011	19	Device control 3
00010100	20	Device control 4
00010101	21	Neg. acknowledge
00010110	22	Synchronous idle
00010111	23	End trans. block
00011000	24	Cancel
00011001	25	End of medium
00011010	26	Substitution
00011011	27	Escape
00011100	28	File separator
00011101	29	Group separator
00011110	30	Record Separator
00011111	31	Unit separator

Byte	Cod.	Char
00100000	32	Spc
00100001	33	!
00100010	34	"
00100011	35	#
00100100	36	$
00100101	37	%
00100110	38	&
00100111	39	'
00101000	40	(
00101001	41	)
00101010	42	*
00101011	43	+
00101100	44	,
00101101	45	-
00101110	46	.
00101111	47	/
00110000	48	0
00110001	49	1
00110010	50	2
00110011	51	3
00110100	52	4
00110101	53	5
00110110	54	6
00110111	55	7
00111000	56	8
00111001	57	9
00111010	58	:
00111011	59	;
00111100	60	<
00111101	61	=
00111110	62	>
00111111	63	?

Byte	Cod.	Char
01000000	64	@
01000001	65	A
01000010	66	B
01000011	67	C
01000100	68	D
01000101	69	E
01000110	70	F
01000111	71	G
01001000	72	H
01001001	73	I
01001010	74	J
01001011	75	K
01001100	76	L
01001101	77	M
01001110	78	N
01001111	79	O
01010000	80	P
01010001	81	Q
01010010	82	R
01010011	83	S
01010100	84	T
01010101	85	U
01010110	86	V
01010111	87	W
01011000	88	X
01011001	89	Y
01011010	90	Z
01011011	91	[
01011100	92	\
01011101	93	]
01011110	94	^
01011111	95	_

Byte	Cod.	Char
01100000	96	`
01100001	97	a
01100010	98	b
01100011	99	c
01100100	100	d
01100101	101	e
01100110	102	f
01100111	103	g
01101000	104	h
01101001	105	i
01101010	106	j
01101011	107	k
01101100	108	l
01101101	109	m
01101110	110	n
01101111	111	o
01110000	112	p
01110001	113	q
01110010	114	r
01110011	115	s
01110100	116	t
01110101	117	u
01110110	118	v
01110111	119	w
01111000	120	x
01111001	121	y
01111010	122	z
01111011	123	{
01111100	124	|
01111101	125	}
01111110	126	~
01111111	127	Del

Codice ASCII ***ESTESO*** (per Windows in Italiano)

Byte	Cod.	Char
10000000	128	Ç
10000001	129	ü
10000010	130	é
10000011	131	â
10000100	132	ä
10000101	133	à
10000110	134	å
10000111	135	ç
10001000	136	ê
10001001	137	ë
10001010	138	è
10001011	139	ï
10001100	140	î
10001101	141	ì
10001110	142	Ä
10001111	143	Å
10010000	144	É
10010001	145	æ
10010010	146	Æ
10010011	147	ô
10010100	148	ö
10010101	149	ò
10010110	150	û
10010111	151	ù
10011000	152	ÿ
10011001	153	Ö
10011010	154	Ü
10011011	155	ø
10011100	156	£
10011101	157	Ø
10011110	158	×
10011111	159	ƒ

Byte	Cod.	Char
10100000	160	á
10100001	161	í
10100010	162	ó
10100011	163	ú
10100100	164	ñ
10100101	165	Ñ
10100110	166	ª
10100111	167	º
10101000	168	¿
10101001	169	®
10101010	170	¬
10101011	171	½
10101100	172	¼
10101101	173	¡
10101110	174	«
10101111	175	»
10110000	176	▒
10110001	177	▒
10110010	178	▒
10110011	179	┤
10110100	180	┤
10110101	181	Á
10110110	182	Â
10110111	183	À
10111000	184	©
10111001	185	╣
10111010	186	╣
10111011	187	+
10111100	188	+
10111101	189	¢
10111110	190	¥
10111111	191	+

Byte	Cod.	Char
11000000	192	+
11000001	193	-
11000010	194	-
11000011	195	+
11000100	196	-
11000101	197	+
11000110	198	ã
11000111	199	Ã
11001000	200	+
11001001	201	+
11001010	202	-
11001011	203	-
11001100	204	¦
11001101	205	-
11001110	206	+
11001111	207	¤
11010000	208	ð
11010001	209	Ð
11010010	210	Ê
11010011	211	Ë
11010100	212	È
11010101	213	ı
11010110	214	Í
11010111	215	Î
11011000	216	Ï
11011001	217	+
11011010	218	+
11011011	219	▒
11011100	220	▒
11011101	221	¦
11011110	222	Ì
11011111	223	▒

Byte	Cod.	Char
11100000	224	Ó
11100001	225	ß
11100010	226	Ô
11100011	227	Ò
11100100	228	õ
11100101	229	Õ
11100110	230	µ
11100111	231	þ
11101000	232	Þ
11101001	233	Ú
11101010	234	Û
11101011	235	Ù
11101100	236	ý
11101101	237	Ý
11101110	238	¯
11101111	239	´
11110000	240	-
11110001	241	±
11110010	242	‗
11110011	243	¾
11110100	244	¶
11110101	245	§
11110110	246	÷
11110111	247	¸
11111000	248	º
11111001	249	¨
11111010	250	·
11111011	251	¹
11111100	252	³
11111101	253	²
11111110	254	■
11111111	255	

SOMMARIO

Ricapitolando i concetti visti:

- il bit, che cos'è e che valori può assumere;
- il problema di codificare le informazioni dato un alfabeto di soli 0 e 1;
- che cos'è il byte, e come ci si è arrivati;
- come codificare una stringa di caratteri alfanumerici in 0 e 1;
- come decodificare da una sequenza di 0 e 1 in una stringa alfanumerica;

Vedete lezione 4 del testo con esercizi su codifiche.