

La Codifica del Testo: ASCII e Unicode

Come fa il computer a capire la differenza tra la lettera "A" e il numero 65? Semplice: usa una **codifica del testo**, cioè una tabella che associa ogni simbolo (lettera, numero, emoji) a un valore numerico.

Senza un sistema di codifica, il computer vedrebbe solo una sequenza di bit tipo `01000001` e non saprebbe se è una lettera, un numero o che altro. Con la codifica invece sa che `01000001` = "A"!

ASCII: American Standard Code for Information Interchange

ASCII è il nonno di tutti i sistemi di codifica del testo, sviluppato negli **anni '60** (quando i computer erano grandi come armadi).

Usa **7 bit** per rappresentare **128 simboli** diversi:

- **Lettere maiuscole e minuscole:** A-Z, a-z (sì, maiuscole e minuscole hanno codici diversi!)
- **Numeri:** 0-9 (attenzione: il carattere "0" è diverso dal numero 0!)
- **Simboli speciali:** `!`, `@`, `#`, `$`, `%` ...
- **Caratteri di controllo:** tipo `Enter` (a capo), `Tab` (tabulazione), `Backspace` ...

Tabella dei Caratteri ASCII

Simbolo	Codice Decimale	Codice Binario
A	65	01000001
B	66	01000010
a	97	01100001
b	98	01100010
0	48	00110000
1	49	00110001
@	64	01000000
!	33	00100001

ASCII TABLE

Decimal	Hexadecimal	Binary	Octal	Char	Decimal	Hexadecimal	Binary	Octal	Char	Decimal	Hexadecimal	Binary	Octal
0	0	0	0	[NULL]	48	30	110000	60	0	96	60	1100000	140
1	1	1	1	[START OF HEADING]	49	31	110001	61	1	97	61	1100001	141
2	2	10	2	[START OF TEXT]	50	32	110010	62	2	98	62	1100010	142
3	3	11	3	[END OF TEXT]	51	33	110011	63	3	99	63	1100011	143
4	4	100	4	[END OF TRANSMISSION]	52	34	110100	64	4	100	64	1100100	144
5	5	101	5	[ENQUIRY]	53	35	110101	65	5	101	65	1100101	145
6	6	110	6	[ACKNOWLEDGE]	54	36	110110	66	6	102	66	1100110	146
7	7	111	7	[BELL]	55	37	110111	67	7	103	67	1100111	147
8	8	1000	10	[BACKSPACE]	56	38	111000	70	8	104	68	1101000	150
9	9	1001	11	[HORIZONTAL TAB]	57	39	111001	71	9	105	69	1101001	151
10	A	1010	12	[LINE FEED]	58	3A	111010	72	:	106	6A	1101010	152
11	B	1011	13	[VERTICAL TAB]	59	3B	111011	73	;	107	6B	1101011	153
12	C	1100	14	[FORM FEED]	60	3C	111100	74	<	108	6C	1101100	154
13	D	1101	15	[CARRIAGE RETURN]	61	3D	111101	75	=	109	6D	1101101	155
14	E	1110	16	[SHIFT OUT]	62	3E	111110	76	>	110	6E	1101110	156
15	F	1111	17	[SHIFT IN]	63	3F	111111	77	?	111	6F	1101111	157
16	10	10000	20	[DATA LINK ESCAPE]	64	40	1000000	100	@	112	70	1110000	160
17	11	10001	21	[DEVICE CONTROL 1]	65	41	1000001	101	A	113	71	1110001	161
18	12	10010	22	[DEVICE CONTROL 2]	66	42	1000010	102	B	114	72	1110010	162
19	13	10011	23	[DEVICE CONTROL 3]	67	43	1000011	103	C	115	73	1110011	163
20	14	10100	24	[DEVICE CONTROL 4]	68	44	1000100	104	D	116	74	1110100	164
21	15	10101	25	[NEGATIVE ACKNOWLEDGE]	69	45	1000101	105	E	117	75	1110101	165
22	16	10110	26	[SYNCHRONOUS IDLE]	70	46	1000110	106	F	118	76	1110110	166
23	17	10111	27	[END OF TRANS. BLOCK]	71	47	1000111	107	G	119	77	1110111	167
24	18	11000	30	[CANCEL]	72	48	1001000	110	H	120	78	1111000	170
25	19	11001	31	[END OF MEDIUM]	73	49	1001001	111	I	121	79	1111001	171
26	1A	11010	32	[SUBSTITUTE]	74	4A	1001010	112	J	122	7A	1111010	172
27	1B	11011	33	[ESCAPE]	75	4B	1001011	113	K	123	7B	1111011	173
28	1C	11100	34	[FILE SEPARATOR]	76	4C	1001100	114	L	124	7C	1111100	174
29	1D	11101	35	[GROUP SEPARATOR]	77	4D	1001101	115	M	125	7D	1111101	175
30	1E	11110	36	[RECORD SEPARATOR]	78	4E	1001110	116	N	126	7E	1111110	176
31	1F	11111	37	[UNIT SEPARATOR]	79	4F	1001111	117	O	127	7F	1111111	177
32	20	100000	40	[SPACE]	80	50	1010000	120	P				
33	21	100001	41	!	81	51	1010001	121	Q				
34	22	100010	42	"	82	52	1010010	122	R				
35	23	100011	43	#	83	53	1010011	123	S				
36	24	100100	44	\$	84	54	1010100	124	T				
37	25	100101	45	%	85	55	1010101	125	U				
38	26	100110	46	&	86	56	1010110	126	V				
39	27	100111	47	'	87	57	1010111	127	W				
40	28	101000	50	(88	58	1011000	130	X				
41	29	101001	51)	89	59	1011001	131	Y				
42	2A	101010	52	*	90	5A	1011010	132	Z				
43	2B	101011	53	+	91	5B	1011011	133	[
44	2C	101100	54	,	92	5C	1011100	134	\				
45	2D	101101	55	-	93	5D	1011101	135]				
46	2E	101110	56	.	94	5E	1011110	136	^				
47	2F	101111	57	/	95	5F	1011111	137	_				

Limiti di ASCII (spoiler: non basta!)

ASCII funziona bene per l'**inglese**, ma ha un problemonone: non ha spazio per lettere accentate (àèìòù), simboli di altre lingue (ñ, ü, ç), caratteri cinesi, giapponesi, arabi, emoji... niente di tutto questo!

Provate a scrivere "caffè" in ASCII puro: diventa "caff?" perché la "è" non esiste nella tabella ASCII. Disastro!

Per questo è nato **Unicode**...

Unicode: Il Codice Universale (finalmente!)

Unicode è lo standard moderno che include simboli da **quasi tutte le lingue del mondo**: latino, greco, cirillico, arabo, cinese, giapponese, coreano, hindi... e anche le **emoji**! 🍕

Il formato più usato è **UTF-8** (e il suo fratello **UTF-16**):

- **UTF-8**: codifica variabile che usa da 1 a 4 byte per simbolo, a seconda di quanto è "esotico". Lettere inglesi = 1 byte, caratteri cinesi = 3-4 byte.
- **È retrocompatibile con ASCII**: tutti i simboli ASCII occupano esattamente 1 byte in UTF-8 con lo stesso valore. Geniale!

Questa compatibilità rende UTF-8 perfetto per il web: può mostrare testi in qualsiasi lingua, emoji incluse, senza problemi. 🌐

Fun fact: Esistono oltre 143.000 caratteri Unicode, incluse emoji, simboli matematici, note musicali, e perfino geroglifici egizi!

Differenze tra ASCII e Unicode

Caratteristica	ASCII	Unicode
Bit utilizzati	7 bit (128 simboli)	Variabile (UTF-8, UTF-16, ecc.)
Supporto per lingue	Solo inglese e simboli base	Tutte le lingue e simboli
Compatibilità UTF-8	Compatibile	UTF-8 è una codifica di Unicode
Utilizzo principale	Vecchi sistemi e dati semplici	Pagine web, applicazioni moderne

Perché è importante la codifica del testo?

Senza codifiche standardizzate, Internet non funzionerebbe! Immaginate:

- Email con caratteri strani tipo "cafÃ" invece di "caffè"
- Siti web incomprensibili se usano lingue diverse dall'inglese
- Emoji che appaiono come quadratini vuoti

Grazie a **Unicode** e **UTF-8** oggi possiamo scrivere in qualsiasi lingua, usare emoji 🤔, e tutto funziona ovunque. Internet è veramente globale!

Esercizi (per chi vuole smanettare)

1. Usando la tabella ASCII, scrivi il codice binario per i caratteri `c` e `i`.
2. Converti la frase "Hello!" in binario usando ASCII (suggerimento: guardate i codici di H, e, l, l, o, !).
3. Perché il codice ASCII di "A" (65) è diverso da quello di "a" (97)? Cosa possiamo dedurre?

Mappa concettuale

