

Rappresentazione di caratteri, stringhe e altri dati

Fondamenti di Informatica I
Corso di laurea in Ingegneria Informatica e Automatica
Sapienza Università di Roma

Domenico Lembo, Paolo Liberatore, Giuseppe Santucci,
Alberto Marchetti Spaccamela, Marco Schaerf

Rappresentazione di caratteri

- Il calcolatore è in grado di manipolare solo due simboli elementari che noi codifichiamo normalmente in “0” e “1”
- Tutte le informazioni possono essere rappresentate con sequenze di “0” e “1”
- La dimensione minima di una cella di memoria è 8 bit, $2^8 = 256$ possibili configurazioni di “0” e “1”
- Siamo quindi interessati a capire come possiamo realizzare tali rappresentazioni.
- Partiamo dalla rappresentazione dei caratteri:
 - ogni carattere si rappresenta con un numero (codificato in binario)
 - versioni più semplici: un numero a otto bit, ovvero un byte
 - versione moderna più utilizzata (anche da Python): da 8 a 32 bit, ovvero da 1 a 4 byte

Codifica ASCII

- ASCII: American Standard Code for Information Interchange
- pubblicato dall'American National Standards Institute (ANSI) nel 1968.
- È un sistema di codifica a 7 bit. Questo consente di utilizzare $2^7=128$ numeri (da 0 a 127) per codificare altrettanti caratteri.
- **Esempi:**
 - 'z' → 1111010 (numero 122 - espresso in binario con 7 bit)
 - ',' → 0111011 (numero 59)
 - 'E' → 1000101 (numero 69)
 - '0' → 0110000 (numero 48)
- Invece di 7 bit è possibile (e necessario) usarne 8, ponendo il primo bit pari a 0 (quindi, ad esempio, la codifica di 'z' diventa 01111010, che rappresenta sempre il numero 122 espresso in binario con 8 bit – (la rappresentazione binaria dei numeri naturali sarà oggetto di una lezione successiva)

Caratteri speciali

- alcuni numeri (da 0 a 31, e il 127) non rappresentano veri caratteri, ma simboli speciali (non stampabili), comunemente detti **caratteri di controllo**
- Ad esempio
 - 0 (chiamato NUL) indica la fine di una stringa
 - 7 (chiamato BEL) indica un beep, inteso come suono (bell, campanello)!
 - 9 (chiamato TAB) indica una tabulazione orizzontale
 - 10 (chiamato LF, e cioè Line Feed) indica l'andata a capo (sulle vecchie stampanti a rullo faceva avanzare la carta di una riga) indicato con `\n` (si usa esplicitamente in Python)
 - 13 (chiamato CR, e cioè Carriage Return) indica lo spostamento del cursore all'inizio della linea
 - 19 (chiamato DC3, dove DC sta per Device Control) indica la richiesta sospensione trasmissione
 - 17 (chiamato DC1) indica richiesta ripresa trasmissione

Binary	Dec	Hex	Abbr	C	Description
000 0000	0	00	NUL	\0	Null character
000 0001	1	01	SOH		Start of Header
000 0010	2	02	STX		Start of Text
000 0011	3	03	ETX		End of Text
000 0100	4	04	EOT		End of Transmission
000 0101	5	05	ENQ		Enquiry
000 0110	6	06	ACK		Acknowledgment
000 0111	7	07	BEL	\a	Bell
000 1000	8	08	BS	\b	Backspace
000 1001	9	09	HT	\t	Horizontal Tab
000 1010	10	0A	LF	\n	Line feed
000 1011	11	0B	VT	\v	Vertical Tab
000 1100	12	0C	FF	\f	Form feed
000 1101	13	0D	CR	\r	Carriage return
000 1110	14	0E	SO		Shift Out
000 1111	15	0F	SI		Shift In
001 0000	16	10	DLE		Data Link Escape
001 0001	17	11	DC1		Device Control 1 (oft. XON)
001 0010	18	12	DC2		Device Control 2
001 0011	19	13	DC3		Device Control 3 (oft. XOFF)
001 0100	20	14	DC4		Device Control 4
001 0101	21	15	NAK		Negative Acknowledgement
001 0110	22	16	SYN		Synchronous Idle
001 0111	23	17	ETB		End of Trans. Block
001 1000	24	18	CAN		Cancel
001 1001	25	19	EM		End of Medium
001 1010	26	1A	SUB		Substitute
001 1011	27	1B	ESC	\e	Escape
001 1100	28	1C	FS		File Separator
001 1101	29	1D	GS		Group Separator
001 1110	30	1E	RS		Record Separator
001 1111	31	1F	US		Unit Separator
111 1111	127	7F	DEL		Delete

La tabella ASCII

Binary	Oct	Dec	Hex	Glyph
010 0000	040	32	20	
010 0001	041	33	21	!
010 0010	042	34	22	"
010 0011	043	35	23	#
010 0100	044	36	24	\$
010 0101	045	37	25	%
010 0110	046	38	26	&
010 0111	047	39	27	'
010 1000	050	40	28	(
010 1001	051	41	29)
010 1010	052	42	2A	*
010 1011	053	43	2B	+
010 1100	054	44	2C	,
010 1101	055	45	2D	-
010 1110	056	46	2E	.
010 1111	057	47	2F	/
011 0000	060	48	30	0
011 0001	061	49	31	1
011 0010	062	50	32	2
011 0011	063	51	33	3
011 0100	064	52	34	4
011 0101	065	53	35	5
011 0110	066	54	36	6
011 0111	067	55	37	7
011 1000	070	56	38	8
011 1001	071	57	39	9
011 1010	072	58	3A	:
011 1011	073	59	3B	;
011 1100	074	60	3C	<
011 1101	075	61	3D	=
011 1110	076	62	3E	>
011 1111	077	63	3F	?

Binary	Oct	Dec	Hex	Glyph
100 0000	100	64	40	@
100 0001	101	65	41	A
100 0010	102	66	42	B
100 0011	103	67	43	C
100 0100	104	68	44	D
100 0101	105	69	45	E
100 0110	106	70	46	F
100 0111	107	71	47	G
100 1000	110	72	48	H
100 1001	111	73	49	I
100 1010	112	74	4A	J
100 1011	113	75	4B	K
100 1100	114	76	4C	L
100 1101	115	77	4D	M
100 1110	116	78	4E	N
100 1111	117	79	4F	O
101 0000	120	80	50	P
101 0001	121	81	51	Q
101 0010	122	82	52	R
101 0011	123	83	53	S
101 0100	124	84	54	T
101 0101	125	85	55	U
101 0110	126	86	56	V
101 0111	127	87	57	W
101 1000	130	88	58	X
101 1001	131	89	59	Y
101 1010	132	90	5A	Z
101 1011	133	91	5B	[
101 1100	134	92	5C	\
101 1101	135	93	5D]
101 1110	136	94	5E	^
101 1111	137	95	5F	_

Binary	Oct	Dec	Hex	Glyph
110 0000	140	96	60	`
110 0001	141	97	61	a
110 0010	142	98	62	b
110 0011	143	99	63	c
110 0100	144	100	64	d
110 0101	145	101	65	e
110 0110	146	102	66	f
110 0111	147	103	67	g
110 1000	150	104	68	h
110 1001	151	105	69	i
110 1010	152	106	6A	j
110 1011	153	107	6B	k
110 1100	154	108	6C	l
110 1101	155	109	6D	m
110 1110	156	110	6E	n
110 1111	157	111	6F	o
111 0000	160	112	70	p
111 0001	161	113	71	q
111 0010	162	114	72	r
111 0011	163	115	73	s
111 0100	164	116	74	t
111 0101	165	117	75	u
111 0110	166	118	76	v
111 0111	167	119	77	w
111 1000	170	120	78	x
111 1001	171	121	79	y
111 1010	172	122	7A	z
111 1011	173	123	7B	{
111 1100	174	124	7C	
111 1101	175	125	7D	}
111 1110	176	126	7E	~

Caratteri speciali

Caratteri stampabili

Limiti della codifica ASCII

- La codifica ASCII **cattura solo un limitato di caratteri**. Ad esempio, mancano le lettere accentate italiane (à, è, é, ì, ò, ù), o la ñ spagnola, i caratteri tedeschi ä, ö, ü, ß
- Ovviamente mancano molti altri simboli, come gli ideogrammi o i simboli matematici e chimici, o anche un simbolo di uso molto comune come il simbolo dell'euro
- Sono quindi state proposte altre codifiche per ampliare l'uso dei caratteri rappresentabili
- Nel seguito ci concentriamo in particolare sulle codifiche **ISO-8859-1** e **Unicode UTF-8**

ISO-IEC 8859

- Estende i codici ASCII a 8 bit ponendo il primo bit a 1
- Estensione **non univoca**: le 128 codifiche dipendono del *code page* selezionato e impostato nel sistema operativo
- **15** diversi codici nazionali (*code page*) **ISO-IEC 8859-x**
 - Erano 16, il 12 è morto.... (Celtico, ora 14)
- A noi interessa **ISO-IEC 8859-1 Latin-1 West European**
che codifica i caratteri usati nella maggior parte delle lingue europee occidentali, incluse l'italiano, lo spagnolo e il tedesco. È probabilmente la parte di ISO-8859 più usata.
- Impossibile usarli simultaneamente: la confusione nasce quando un carattere è interpretato in un codice nazionale diverso da quello nel quale era stato scritto (l'informazione non è nel byte!)

Problema tipico: caratteri strani nelle e-mail e nelle pagine web

ISO-IEC 8859: la torre di Babele

Comparison of the various parts (1–16) of ISO/IEC 8859

Binary	Oct	Dec	Hex	1	2	3	4	5	6	7	8	9	10	11	13	14	15	16
1010 0000	240	160	A0	Non-breaking space (NBSP)														
1010 0001	241	161	A1	ı	À	Ħ	Ā	Ě		‘		ı	Ā	ŋ	”	Ď	ı	Ā
1010 0010	242	162	A2	ø	˘	κ	Ṭ			’	ø	ø	Ě	ɯ	ø	ḃ	ø	ą
1010 0011	243	163	A3	£	Ł	£	Ṛ	Ġ			£		Ḡ	ɰ		£		Ł
1010 0100	244	164	A4		¤			€	¤	€	¤	Ī	ṇ	¤	Ć		€	
1010 0101	245	165	A5	¥	Ł		Ī	Š		Đ	¥	Ī	ṇ	„	ć	¥	„	
1010 0110	246	166	A6	ı	Š	Ĥ	Ł	ı			ı	Ḳ	ɰ	ı	Đ		Š	
1010 0111	247	167	A7		§			İ			§			ɟ		§		
1010 1000	250	168	A8		ˆ			Ĵ		ˆ		Ł	ɔ	Ø	Ŵ		š	
1010 1001	251	169	A9	©	Š	ı	Š	Љ			©	Đ	ɔ		©			
1010 1010	252	170	AA	ª	Ş	Ě	Ѓ			×	ª	Š	ɰ	Ṛ	Ŵ	ª	Ş	
1010 1011	253	171	AB	«	Ť	Ğ	Ḡ	Ṭ			«	Ṭ	ɰ	«	đ		«	
1010 1100	254	172	AC	¬	Ž	Ĵ	Ṭ	Ķ	‘	¬		Ž	ɰ	¬	Ÿ	¬	Ž	

12 caratteri nelle 15 diverse codifiche di ISO/IEC 8859

Limiti della codifica ISO-8859-1

- Questo sistema va bene per lingue come l'italiano, l'inglese, lo spagnolo, il norvegese, ma **copre solo in parte i simboli usati in altre lingue** (ad es. lingue dell'est europa, il russo, l'arabo, il turco, ecc.)
- Per queste bisogna ricorrere ad altri standard code page
- Per esempio, il code page ISO-8859-9 consente di codificare tutti i caratteri usati nella lingua turca, e usa il numero 253 per la i senza punto i invece che per la y con accento acuto ý come in ISO-8859-1
- Quindi, le varie ISO-8859-x sono incompatibili fra loro e **bisogna sempre specificare quale ISO-8859-x si sta usando**
- Non si possono scrivere testi in **più lingue insieme**
- **alcune lingue**, come il cinese, **hanno più di 256 caratteri**

Unicode

- **Unicode** (anche detto Universal Coded Character Set, o UCS) è un sistema di codifica in grado di rappresentare i caratteri usati in quasi tutte le lingue vive e in alcune lingue morte, simboli matematici e chimici, cartografici, l'alfabeto Braille, ideogrammi ecc. E' in realtà un enorme dizionario di caratteri
- Nella formulazione iniziale usava due byte (non si impara mai dal passato ...) codificando 65.536 caratteri; ora prevede 17 (00..10 in esadecimale) insiemi di codifiche da 65.536 , ovvero $17 \times 65.536 = 1.114.112$ possibili codifiche, rappresentabili con 21 bit (solo 393216 assegnate)
- Per rappresentare in modo efficiente i caratteri, **Unicode prevede tre possibili codifiche**
 - **UTF-8**, sequenza *fino a* 4 unità da 8 bit
 - **UTF-16**, sequenza *fino a* 2 unità da 16 bit (è una evoluzione del precedente UCS-2, codifica di lunghezza fissa a 2 byte)
 - **UTF-32** (nota anche come UCS-4), sequenza di *esattamente* 32 bit per carattere
- Delle tre, UTF-8 è la più efficiente nel gestire lo spazio, consentendo anche di usare una sola unità da 8 bit (cioè un byte) per rappresentare un carattere (i 127 caratteri ASCII)

UTF-8: schema di codifica

		Bits	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
0	127	7	0ddddddd					
128	2047	11	110dddd	10dddddd				
2048	65535	16	1110dddd	10dddddd	10dddddd			
65536	1114111	21	11110ddd	10ddd	10dddddd	10dddddd		
		26	111110dd	10dddddd	10dddddd	10dddddd	10dddddd	
		31	1111110x	10dddddd	10dddddd	10dddddd	10dddddd	10dddddd

5 bit

16 bit

- i primi 128 caratteri, che sono gli stessi di ASCII, otto bit, il cui primo bit è 0
- altri caratteri: sequenza di numerali a otto bit che hanno sempre 1 come primo bit; il primo di questi numeri è nella forma 1...10..., e la quantità di 1 indica la lunghezza in byte della sequenza
- I bit effettivamente usati per la codifica sono indicati in figura con **d** (vedi colonna Bits)
- Chi interpreta il codice, guardando i primi bit del primo byte può sempre stabilire il numero di byte usati per quel particolare carattere
- Il primo byte non comincia mai per **10**: questo permette di *rifasarsi* in caso di errori di trasmissione
- Potenzialmente, UTF-8 potrebbe usare sequenze fino a 6 byte con 31 bit (2.147.483.648 code points), ma nel 2003 è stato limitato a 4 byte per coprire solo l'intervallo descritto formalmente nello standard Unicode, per il quale 21 bit sono sufficienti (massimo numero rappresentabile 1.114.111 = 10FFFF in esadecimale).

Python usa UTF-8

- La funzione `chr` (intero) restituisce il carattere corrispondente all'intero

```
chr(97)  
'a'
```

```
chr(255)  
'ÿ'
```

```
chr(2100000)  
Traceback (most recent call last):  
  File "<pyshell#11>", line 1, in <module>  
    chr(2100000)  
ValueError: chr() arg not in range(0x110000)
```

- La funzione `ord` (carattere) restituisce il codice UTF-8 corrispondente al carattere

```
ord('a')  
97
```

```
ord('北')  
20000
```

Stringhe

Le stringhe sono sequenze di caratteri

Ad esempio 'ciao a tutti! ' è una stringa che in Python possiamo stampare con il comando

```
print('ciao a tutti!')
```

sequenza fra virgolette = stringa = sequenza di caratteri

Ogni carattere è un numero. I numeri corrispondenti ai caratteri vengono memorizzati in sequenza. La stringa precedente è così rappresentata

99 105 97 111 32 97 32 116 117 116 116 105 33 **0**

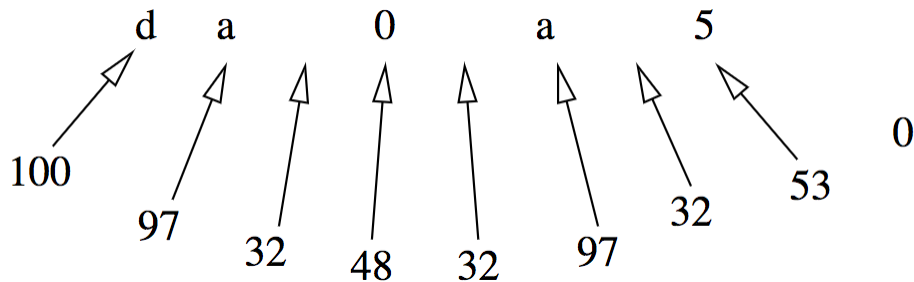
99 = c, 105 = i, 97 = a, ...

32 = spazio

0 = fine stringa

Rappresentazione delle cifre '0' .. '9'

- In ASCII, ISO-8859-1, UTF-8 le cifre '0'..'9' sono rappresentate con i numeri 48..57.
- Quindi nell'esempio precedente lo 0 non corrisponde alla cifra '0', ma è il terminatore di stringa
- Esempio: stringa 'da 0 a 5'



Ritorno a capo

Il ritorno a capo viene indicato con

- Il numero 10 (cioè `\n` Line Feed) (Unix), oppure
- con il numero 13 (cioè `\r` Carriage Return) (Mac Os fino alla versione 9)
- con la sequenza 13 10 (windows, DOS)

C'è un motivo storico per questo, legato all'uso delle prime stampanti:

- 10 = avanzamento carta di una linea
- 13 = ritorno del carrello a inizio linea

Suoni

Sono onde di pressione dell'aria

È possibile fornirne una rappresentazione numerica:

- Pressione misurata a intervalli regolari (es. 48000 volte al secondo)
- Ciascun valore rappresentato in binario (es. a 16 bit)
- Suono = sequenza di questi valori

La sequenza è uguale all'originale (campionando a una frequenza pari al doppio di quella massima)

- variazioni fra una misurazione all'altra non vengono rilevate
- la pressione è un valore continuo

numero prefissato di bit = approssimazione

fedeltà all'originale = alta frequenza di campionamento + alto numero di bit

Colori e Immagini

- La maggior parte dei colori visibili all'occhio umano si possono considerare un miscuglio di **quantità variabili di rosso, verde e blu (RGB)**

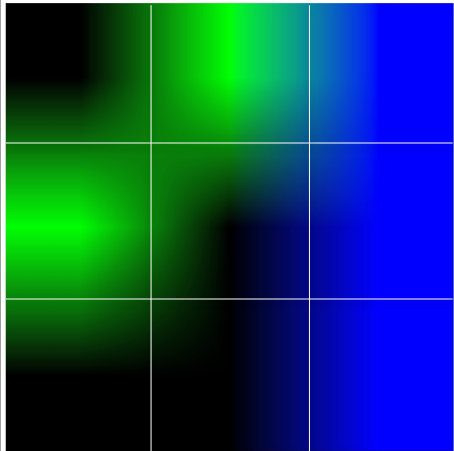
Esempio: massimo di rosso, mezzo verde e niente blu

Immagine raster = griglia di minuscoli quadretti (pixel)

- ogni pixel viene considerata di un colore solo
- per ogni pixel, si rappresentano le quantità di rosso, verde e blu che contiene

Rappresentazione delle immagini

Un semplice formato (ppm)

Immagine (ingrandita)	Rappresentazione
	<pre>P3 3 3 100 0 0 0 0 100 0 0 0 100 0 100 0 0 0 0 0 0 100 0 0 0 0 0 0 0 0 100</pre>

parte iniziale

identificativo del formato (P3)

larghezza e altezza dell'immagine (3×3) – 9 pixel in tutto

massima intensità di colore (100)

matrice

i colori dei pixel, in sequenza (es. 0 100 0 = niente rosso, max verde, niente blu)

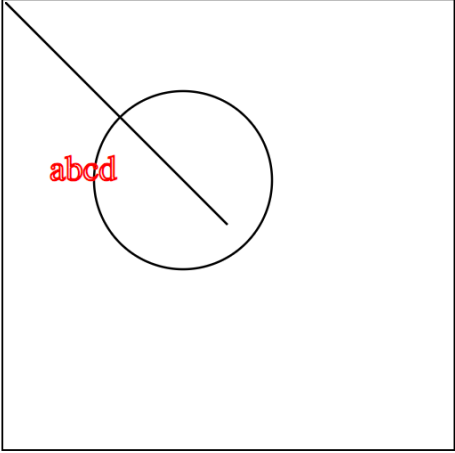
Immagini rappresentate come quadrati di pixel si dicono *raster*.

Approssimazioni e compressioni

- Come per i suoni, si tratta comunque di una rappresentazione approssimata, per i soliti due motivi: ogni pixel viene considerato di un colore unico (quindi variazioni di colore più piccole di un pixel vengono ignorate), e i colori vengono rappresentati con un **numero finito di bit** (per cui esiste una perdita di precisione).
- Esistono meccanismi di compressione, che permettono di ridurre lo spazio di memoria richiesto. Sono basati principalmente su sequenze che si ripetono (es. spesso pixel vicini = colori simili)

Immagini vettoriali

invece dei pixel: figure geometriche elementari

Immagine	Rappresentazione
	<pre><?xml version="1.0" standalone="no"?> <!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN" "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd"> <svg xmlns="http://www.w3.org/2000/svg"> <g style="stroke:#000000;fill:none;"> <polyline points="0,0 100,100" /> <circle cx="80" cy="80" r="40" /> <text x="20" y="80" stroke="#FF0000">abcd</text> </g> </svg></pre>

immagini vettoriali: tag

<polyline points="0,0 100,100" /> segmento da coordinata 0,0 a 100,100

<circle cx="80" cy="80" r="40" /> cerchio

<text x="20" y="80" stroke="#FF0000">abcd</text> scritta abcd in rosso

origine: in alto a sinistra

Immagini vettoriali

immagini vettoriali: meccanismo

sequenze di caratteri (o numeri)

nell'esempio: da `<?xml` fino a `</svg>`

in genere si possono inserire come elementi (oltre a linee, cerchi, ecc.) anche delle immagini raster