

Laboratorio di Informatica per chimica

Rappresentazione delle informazioni: i numeri

Codice

- La relazione che associa ad ogni successione ben formata di simboli di un **alfabeto** il dato corrispondente è detta **codice**.
- Un codice mette quindi in relazione le successioni di simboli con il significato loro attribuito.

Calcolatori digitali

- I calcolatori moderni sono detti **calcolatori digitali**
- **digit** in inglese significa “cifra”
- tutte le informazioni vengono rappresentate in forma numerica:
 - numeri
 - caratteri
 - immagini statiche e in movimento
 - suoni

Il bit

- noi rappresentiamo i numeri utilizzando un alfabeto di dieci simboli (0, 1, ..., 9)
- per i calcolatori si usa un alfabeto binario
- alfabeto binario: costituito da due simboli
 - convenzionalmente “0” e “1”
- binit o **bit** (binary digit, cifra binaria): elemento che assume un valore binario
- il **bit** è anche l’unità elementare di informazione

La codifica binaria

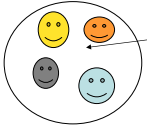
- Avendo a disposizione un solo bit si possono rappresentare
 - **due** elementi diversi:
 - si assegna al primo la **codifica 0**
 - al secondo la **codifica 1**

La codifica binaria

- con **2** bit si possono rappresentare $4 = 2^2$ elementi (oggetti, individui, ...) diversi, assegnando a ciascuno una codifica diversa:
 - paperino codifica 00
 - qui codifica 01
 - quo codifica 10
 - qua codifica 11
- con **3** bit si possono rappresentare $8 = 2^3$ elementi diversi
-
- con **n** bit si possono rappresentare 2^n elementi

Rappresentazione dei numeri

I numeri sono entità matematiche astratte; vanno distinti dalla loro rappresentazione



Il numero cardinale quattro: **cardinalità** degli insiemi contenenti quattro elementi

Rappresentazione unitaria: IIII
Rappresentazione additiva/sottrattiva: IV
Rappresentazione posizionale (base dieci): 4 ecc.

Rappresentazione additivo/sottrattivo:

- Il significato dei simboli che compongono un numero è indipendente dalla posizione in cui compaiono
- Es. sistema con un unico simbolo, per l'unità
- Es. sistema di numerazione romano, con i simboli I, V, X, L, C, D, M

Rappresentazione posizionale

- **Sistema di numerazione arabo:**
 - introdotto in Europa nel Medio Evo
 - in **base dieci**: utilizza le dieci cifre 0, 1, ..., 9
 - è una **notazione posizionale**: il valore di ogni cifra dipende dalla sua posizione nella successione di simboli che rappresenta il numero.
 - Es. $1234 = 1 \times 10^3 + 2 \times 10^2 + 3 \times 10^1 + 4 \times 10^0$
 - Es. $4321 =$
 $= 4 \times 10^3 + 3 \times 10^2 + 2 \times 10^1 + 1 \times 10^0$

Rappresentazione posizionale

- **Sistema di numerazione arabo:**
 - In generale, per un numero composto di n cifre si ha che:
 $c_{n-1}c_{n-2}\dots c_1c_0 = c_{n-1}10^{n-1} + c_{n-2}10^{n-2} + \dots + c_110^1 + c_010^0$
 - Si chiamano **cifre più significative** quelle associate ai pesi maggiori, **cifre meno significative** quelle associate ai pesi minori
 - La cifra c_{n-1} è la più significativa e c_0 è la cifra meno significativa.
 - Es. $1234 = 1 \times 10^3 + 2 \times 10^2 + 3 \times 10^1 + 4 \times 10^0$
 - Es. $4321 =$
 $= 4 \times 10^3 + 3 \times 10^2 + 2 \times 10^1 + 1 \times 10^0$

Rappresentazione posizionale

- **Base 2**: quella in cui lavora il calcolatore
 - cifre 0,1
- **Base 10**: quella dell'utente umano
 - cifre 0,1,2,3,4,5,6,7,8,9
- **Base 8**: per abbreviare i numeri binari
 - cifre 0,1,2,3,4,5,6,7
- **Base 16**: per abbreviare i numeri binari
 - cifre 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

Rappresentazione posizionale

- Se la base della numerazione è B , si hanno a disposizione B cifre, comprese tra 0 e $B-1$.
- Tramite n cifre in base B è possibile rappresentare B^n numeri naturali, da 0 a B^n-1 .

$$c_n c_{n-1} \dots c_1 c_0 =$$

$$= c_n \cdot B^n + c_{n-2} \cdot B^{n-1} + \dots + c_1 \cdot B^1 + c_0 \cdot B^0$$

$$= \sum_{i=0 \dots n} c_i \cdot B^i$$

- **Esempio:** $705_8 = 7 \times 8^2 + 0 \times 8^1 + 5 \times 8^0 = 453_{10}$

Cambio di rappresentazione

- Si applica la definizione
 - $c_n c_{n-1} \dots c_1 c_0 = c_n \cdot B^n + c_{n-1} \cdot B^{n-1} + \dots + c_1 \cdot B^1 + c_0 \cdot B^0$
- Esempi
 - base 2: $1011_{\text{due}} = (1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1)_{\text{dieci}}$
 - base 8: $2705_{\text{otto}} = (2 \cdot 8^3 + 7 \cdot 8^2 + 0 \cdot 8^1 + 5)_{\text{dieci}}$
 - base 16: $3F01_{16} = (3 \cdot 16^3 + 15 \cdot 16^2 + 0 \cdot 16^1 + 1)_{\text{dieci}}$

Rappresentazione binaria

- Da ora in poi consideriamo la rappresentazione binaria e in particolare:
 - Numeri naturali
 - Rappresentazione in modulo e segno
 - Rappresentazione in complemento
 - Rappresentazione in eccesso
 - Numeri interi relativi
 - Numeri reali

N: numeri naturali

- Tramite n cifre in base 2 è possibile rappresentare 2^n numeri naturali, da 0 a $2^n - 1$.
 - Per i numeri naturali si usano di solito 32 bit
 - $2^{32} - 1 = 4.294.967.295 \approx 4 \times 10^9$
 - Raddoppiando la lunghezza, il massimo numero rappresentabile aumenta esponenzialmente. Se si utilizzano 64 bit $2^{64} - 1 \approx 1,6 \times 10^{19}$

I primi 16 numeri binari

0	= 0	1000	= 8
1	= 1	1001	= 9
10	= 2	1010	= 10
11	= 3	1011	= 11
100	= 4	1100	= 12
101	= 5	1101	= 13
110	= 6	1110	= 14
111	= 7	1111	= 15

Numeri naturali: cambio di rappresentazione

- Esempio:

$$101100_{\text{due}} = (1x2^5 + 0x2^4 + 1x2^3 + 1x2^2 + 0x2^1 + 0x2^0)_{\text{dieci}}$$

$$= (32 + 8 + 4)_{\text{dieci}} = 44_{\text{dieci}}$$
- Esercizio:

$$100011_{\text{due}} = (1x2^5 + 0x2^4 + 0x2^3 + 0x2^2 + 1x2^1 + 1x2^0)_{\text{dieci}}$$

$$= (32 + 2 + 1)_{\text{dieci}} = 35_{\text{dieci}}$$

Numeri naturali: cambio di rappresentazione

Ad esempio, se la base è 2, si ripete la divisione intera per 2 e si raccolgono i quozienti interi ed i resti come cifre

numero	div base	quoz.	resto	
6	div 2 =	3	0	Cifra posto 0
3	div 2 =	1	1	Cifra posto 1
1	div 2 =	0	1	Cifra posto 2

$6_{10} = 110_2$ Le cifre ottenute corrispondono alla rappresentazione binaria

Numeri naturali: esempio

Si considerino celle di memoria di 6 bit:

numero	divisore	quoziente	resto	
41	2	20	1	cifra bin. meno significativa
20	2	10	0	
10	2	5	0	
5	2	2	1	
2	2	1	0	
1	2	0	1	cifra bin. più significativa

$41_{\text{dieci}} = 101001_{\text{due}}$ riprova: $2^5+2^3+2^0=32+8+1=41$

Numeri naturali: esempio

Si considerino celle di memoria di 6 bit:

numero	divisore	quoziente	resto	
10	2	5	0	cifra bin. meno significativa
5	2	2	1	
2	2	1	0	
1	2	0	1	
0	2	0	0	
0	2	0	0	cifra bin. più significativa

$10_{\text{dieci}} = 001011_{\text{due}}$ riprova: $2^3+2^1=8+2=10$

Numeri naturali: esercizio

- Si consideri un calcolatore le cui memorie centrali hanno lunghezza 6 bit
 - Quanti e quali numeri naturali posso rappresentare?
Si possono rappresentare $2^6 = 64$ numeri naturali da 0 a 63
 - Come viene rappresentato il numero in base decimale 11?
001011
 - In una cella è memorizzato il numero: 111111.
A quale numero in base decimale corrisponde?
63 perché $2^5+2^4+2^3+2^2+2^1+2^0=32+16+8+4+2+1$ o per la proprietà $np!$
 - Posso rappresentare l'operazione $30+40$?
 - *No, perché 70 non appartiene al range di numeri naturali rappresentabili con celle di 6 bit (errore di overflow)*

Numeri naturali: esercizio

Si considerino celle di memoria di 6 (7) bit:

numero	divisore	quoziente	resto	
11	2	5	1	cifra bin. meno significativa
5	2	2	1	
2	2	1	0	
1	2	0	1	
0	2	0	0	
0	2	0	0	
(0)	2	0	(0)	

$11_{\text{dieci}} = (0)001011_{\text{due}}$ riprova: $2^3+2^1+2^0=8+2+1=11$

Z: numeri interi relativi

- **Codifica con modulo e segno:** consiste nell'indicare il segno seguito dal valore assoluto, come succede normalmente nella codifica decimale.
- Il primo bit indica il segno
 - 0 per positivo
 - 1 per negativo
- Gli altri $n-1$ bit rappresentano il valore assoluto

Aritmetica finita

- Si usa dunque un'aritmetica finita, cioè con un numero massimo di cifre binarie disponibili;
- Nell'aritmetica finita dei calcolatori
 - i numeri relativi sono rappresentati in complemento, come vedremo
 - i numeri "reali" sono rappresentati in virgola mobile, come vedremo
- Siccome il numero di cifre massimo è limitato, la precisione raggiungibile nella rappresentazione dei numeri reali è limitata;

Z: Rappresentazione in complemento a due

- Range di rappresentabilità con n bit
 - Con n bit possono essere rappresentati gli interi compresi tra -2^{n-1} e $+(2^{n-1} - 1)$.
 - Esempio: con 4 bit possono essere rappresentati gli interi compresi tra -2^{4-1} e $+(2^{4-1} - 1)$ cioè tra -8 e +7

Esercizio:

Interi assoluti e interi relativi

Esercizio: date celle di memoria di un byte

- quanti e quali numeri interi assoluti posso rappresentare?
- *posso rappresentare $2^8 = 256$ numeri interi assoluti: da 0 a 255*
- quanti e quali numeri interi relativi posso rappresentare?
- *posso rappresentare $2^8 = 256$ numeri interi relativi: da -128 a +127*

Gli errori di overflow

- L'intervallo dei numeri interi (assoluto o relativi) rappresentabili con un fissato numero di bit è limitato
- Eseguendo un'operazione su numeri rappresentabili, può accadere che il risultato esca dall'intervallo rappresentabile; in tal caso si dice che
 - si ha un errore di overflow
- ESEMPIO con 8 bit: $2^8=256$
 - Caso di valori assoluti: massimo rappresentabile = 255
 - Overflow $150 + 150 = 300$ perché > 255
 - Caso complemento a 2: intervallo rapp.: -128 ... 127
 - Overflow $-100 - 100 = -200$ perché < -128

R: numeri reali

- Rappresentazione con la virgola in base B:
 $c_n \cdot c_0 \cdot c_{-1} \cdot c_{-k} = c_n B^n + \dots + c_0 B^0 + c_{-1} B^{-1} + \dots + c_{-k} B^{-k}$
- Esempio:
 $123,45 = 1 \times 10^2 + 2 \times 10^1 + 3 \times 10^0 + 4 \times 10^{-1} + 5 \times 10^{-2}$
- Esercizio: 543,21
 $543,21 = 5 \times 10^2 + 4 \times 10^1 + 3 \times 10^0 + 2 \times 10^{-1} + 1 \times 10^{-2}$

R: codifica

- **Notazione scientifica:** un numero viene rappresentato come
 $\pm m \times 10^p$ Es. $123.000.000 = 1,23 \times 10^8$
- Se, più in generale, la base è B,
 $\pm m \times B^p$
- Il coefficiente m è detto **mantissa** (la convenzione è di inserire implicitamente la virgola decimale subito dopo la prima cifra)
- p , detto **caratteristica**, è l'esponente a cui elevare la base B

R: codifica

- Esempio: $1,23 \times 10^8$
base 10, segno +, mantissa 123, caratteristica 8
- Esercizio: Scrivere in notazione scientifica il numero 40000002 e indicarne mantissa e caratteristica
 $40000002 = 4,0000002 \times 10^7$
base 10, segno +, mantissa 40000002, caratteristica 7

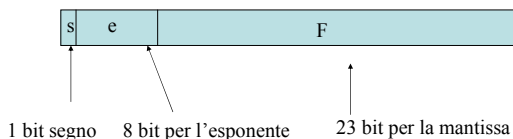
R: rappresentazione in virgola mobile

- La rappresentazione binaria dei numeri reali che usa la notazione scientifica è detta **rappresentazione in virgola mobile** (floating point).
- $101010000_{\text{due}} = 1,0101 \times 10^{1000}_{\text{due}}$
- Cambiando il numero di cifre dedicato alla rappresentazione di mantissa ed esponente cambia la precisione dei risultati che si ottengono.
- L'interesse a uniformare la precisione di calcolo ha condotto alla definizione di uno standard internazionale proposto dall'Institute of Electrical and Electronic Engineers (IEEE).

R: rappresentazione in virgola mobile

- Virgola mobile: $m E e$ con
 - mantissa m per numeri diversi da 0: $1 \leq m < B$
 - e esponente
 - significato $m E e = m \cdot B^e$
- Esempio decimale: 344,013 in virgola mobile si scrive:
 - $3,44013 E 3 = 3,44013 \cdot 10^3$
- Esempio binario: 101,001 in virgola mobile si scrive:
 - $1,01001 E 10 = (1+2^{-2}+2^{-5}) \cdot 2^2$

R: rappresentazione in virgola mobile precisione singola, 32 bit



R: errori di arrotondamento

- La rappresentazione esatta di alcuni numeri richiederebbe un numero infinito di cifre
 - Es. $1/3 = 0,3333333\dots$ $\pi = 3,14159\dots$
- Un problema analogo sorge per numeri con valore assoluto molto grande o molto piccolo, in cui il numero di cifre richiesto non sarebbe infinito, ma molto elevato
- In questi casi possiamo considerare solo le cifre più significative.

R: errori di arrotondamento

- Qualunque sia la codifica scelta, la rappresentazione dei numeri nel calcolatore è soggetta ad **approssimazioni**.
- Il limitato numero di cifre disponibili nella mantissa porta ad errori di arrotondamento quando si debbano rappresentare numeri con una mantissa più lunga.
- Tali approssimazioni si propagano nel corso della esecuzione delle operazioni causando **errori numerici** anche importanti.

I CARATTERI...

La rappresentazione dei caratteri

Caratteri Alfanumerici: "a, ..., z", "A, ..., Z" e "0, ..., 9"

Simboli: "- + * @ [] () { } § ° ç ^ ì à ò é è \$..."

"A" **CODIFICA** → $(65)_{10} = 01000001$

- Codifica:
 - Definisci un ordine sui caratteri (es. $a < b$)
 - Associa ad ogni carattere il numero risultante dall'ordinamento
- I più noti sono i codici **ASCII** e **Unicode**

Codice ASCII

(American Standard Code for Information Interchange)

- Il codice ASCII prevede la codifica di ogni carattere in un byte.
- Delle 256 possibili configurazioni...
 - 128 sono state definite in modo univoco in tutto il mondo
 - Le restanti corrispondono a caratteri diversi a seconda delle esigenze dei diversi Paesi. Nell'Europa occidentale si usa l'ISO-8859-1

Codice ASCII

- Nel codice ASCII, ad esempio:
 - A = 0100 0001
 - c = 0110 0011
 - (= 0011 0001
 - Spazio = 0010 0000
- Nel codice ASCII non ci sono le lettere accentate, che sono state aggiunte nell'ISO-8859-1

Codice ASCII

32	,	44	8	56	D	68	P	80	\	92	h	104	t	116	
!	33	-	45	9	57	E	69	Q	81]	93	i	105	u	117
"	34	.	46	:	58	F	70	R	82	^	94	j	106	v	118
#	35	/	47	;	59	G	71	S	83	_	95	k	107	w	119
\$	36	0	48	<	60	H	72	T	84	`	96	l	108	x	120
§	37	1	49	=	61	I	73	U	85	a	97	m	109	y	121
&	38	2	50	>	62	J	74	V	86	b	98	n	110	z	122
'	39	3	51	?	63	K	75	W	87	c	99	o	111	{	123
(40	4	52	@	64	L	76	X	88	d	100	p	112		124
)	41	5	53	A	65	M	77	Y	89	e	101	q	113	}	125
*	42	6	54	B	66	N	78	Z	90	f	102	r	114	~	126
+	43	7	55	C	67	O	79	[91	g	103	s	115		127

Il codice ASCII

- È uno dei formati più interoperabili in assoluto
- È standardizzato
- È leggibile da tutti i programmi che manipolano testi (strutturati e non)

Cosa succede, però...

- ...se salvo un file di testo qui e tento di leggerlo in Polonia?
- I caratteri ASCII (i primi 128) sono gli stessi, ma gli altri sono diversi (in Polonia usano l'ISO-8859-2)
- Ad es., può darsi che la "à" diventi ad esempio una Ł

Internazionalizzazione

- Internet permette di scambiare file – occorre che i formati siano altamente interoperabili
- Occorre estendere l'ASCII in modo da comprendere tutti i caratteri possibili
- Per questo è nato Unicode, che usa 4 byte per rappresentare un carattere

Unicode

- 4 byte = 4×8 bit = 32 bit
- Con 32 bit si hanno a disposizione 2^{32} = circa 4.300.000.000 di caratteri!
- Ovviamente si spreca spazio (un testo scritto con caratteri “normali” occupa il quadruplo)