MINISTERO DELLA DIFESA

Accademia Aeronautica di Pozzuoli

Elementi di Informatica
e
Fondamenti di Informatica

La rappresentazione delle informazioni

Parte prima: Informazione, dato, codifica binaria; codifica dei caratteri

Prof. Stefano RUSSO
Università degli Studi di Napoli Federico II
DIETI – Dipartimento di Ingegneria Elettrica e Tecnologie dell'Informazione
Via Claudio 21, 80125 NAPOLI
Tel.: 081 7683832, Email: stefano.russo@unina.it

Informazione

- **L'informazione** è legata al concetto di **scelta**: esse sono caratterizzate dalla tripla {tipo, valore, attributo}:
  - **Tipo**: definisce **l'insieme degli elementi** fra i quali si compie la scelta.
  - **Valore**: è il particolare **elemento scelto**.
  - **Attributo**: è un **identificatore** che **associa un significato al valore,** dando senso all'informazione.

> **Es.:** "L'ora di New York è 12:00pm", "La capitale della Francia è Parigi", "Il semaforo di ingresso in galleria è rosso", "Mario ha 18 anni".

- Affinché una informazione possa essere utilizzata è necessario rappresentarla!
  - …. se non fosse esistita la scrittura …
  - scrivere, leggere ed elaborare informazioni richiede di aver preliminarmente definito una rappresentazione (**codifica**), ossia una serie di regole e convenzioni da seguire

Codifica

- La **codifica** è l'insieme di convenzioni e di regole per trasformare un'informazione in una sua rappresentazione e viceversa.
- Una informazione può essere codificata in modi diversi (rappresentazioni diverse)
  - Es.: le rappresentazioni araba o romana dei numeri (i simboli "1" e "`I`" costituiscono due codifiche diverse della stessa informazione numerica)
- Per essere correttamente elaborata, l'informazione deve essere codificata in una rappresentazione manipolabile dall'elaboratore.

Codici

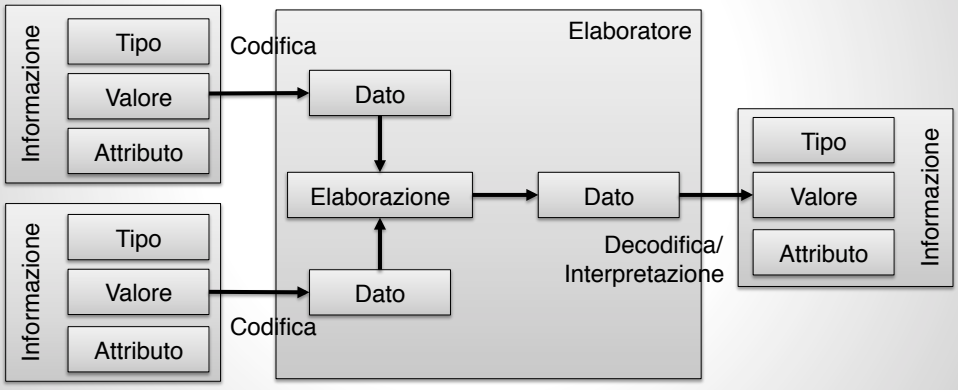
- Un codice è un sistema di simboli che permette la rappresentazione dell'informazione ed è definito da:
  - i simboli da rappresentare (alfabeto origine o sorgente);
  - l'alfabeto in codice dei simboli elementari possibili;
  - le parole codice o stringhe che rappresentano sequenze possibili (ammissibili) di simboli elementari: la *lunghezza* di una parola codice è il numero di simboli dell'alfabeto da cui è composta;
  - le regole per costruire parole codice.
- Es.: linguaggio MORSE
  - alfabeto sorgente: a, b, c, …
  - alfabeto: • (punto), **–** (linea)
  - codifica della a: parola codice **•–**, lunghezza: 2
  - codifica della s: **•••**, lunghezza: 3
  - codifica della o: **–––**, lunghezza: 3

Codifica a lunghezza fissa/variabile

- Codifica a lunghezza fissa:
  - tutte le parole codice hanno sempre la stessa lunghezza fissata da particolari esigenze applicative.
- Codifica a lunghezza variabile:
  - Non tutte le parole codice hanno la stessa lunghezza
- La scrittura è un evidente caso di codifica a lunghezza variabile.
- I calcolatori adottano codifiche a lunghezza fissata e definita

La trasformazione delle informazioni

I calcolatori elaborano **dati** ovvero **rappresentazioni dei valori delle informazioni!**
L'elaboratore opera sulle rappresentazioni dei valori delle informazioni e le trasforma in rappresentazione dei risultati!



Dati

Le rappresentazioni dei valori (dati) devono essere scelte in un formato conveniente per l'elaborazione/elaboratore.
Es.: per l'uomo moderno è più pratico sommare numeri arabi che romani, i computer (elettronici e digitali) sono più efficienti con rappresentazioni in binario.

Informazione: Ho cento Euro nel portafoglio.
Tipo: Valùta EUR
Attributo: "Soldi nel portafoglio"
Valore:
- Cento (rappresentazione in italiano).
- 100 (rappresentazione in num. arabi)
- C (rappresentazione in num. romani)
- 1100100 (rappresentazione in binario)

Informazione: Ho mille Euro in banca.
Tipo: Valùta EUR
Attributo: "Soldi in banca"
Valore:
- Mille (rappresentazione in italiano).
- 1000 (rappresentazione in num. arabi)
- M (rappresentazione in num. romani)
- 1111101000 (rappresentazione in binario)

Elaborazione dei dati
Y=F(X)
X dati in input
Y dati in output

+

1100

Decodifica

Il dato in output è una rappresentazione di un valore. Interpretato (o decodificato), diventa informazione.

Informazione: Ho millecento Euro.
Tipo: Valùta EUR
Attributo: "Saldo"
Valore:
- Millecento
- 1100
- MC
- 10001001100

Rappresentazione Analogica e Discreta

- La **rappresentazione del valore** di una informazione (il dato) può essere **analogica** o **discreta (o numerica, o digitale: "a cifre")**.
- In una rappresentazione analogica **le proprietà del fenomeno rappresentato sono *omomorfe*** alla rappresentazione.
  - **la rappresentazione varia in analogia con la grandezza reale**:
    - una grandezza è rappresentata in modo continuo e la gran parte delle **grandezze fisiche della realtà** sono di tipo continuo.
    - La codifica del valore è **intensionale** (implicita) data dal legame con la variazione della grandezza analoga.
- Una **rappresentazione digitale** usa **un insieme finito di rappresentazioni distinte che vengono messe in relazione con alcuni elementi dell'universo rappresentato**.
  - La codifica è **estensionale** (esplicita) definita tramite un **alfabeto** e **regole di codifica**.
  - Se il tipo dell'informazione non è finito, una rappresentazione digitale introdurrà necessariamente un errore di rappresentazione dei valori.

Esempio: la bilancia

Informazione: "Ieri il mio peso era di 82.340 Kg"; **Tipo**: $\mathbb{R}^+$; **Attributo**: "Peso di ieri";
Valore rappresentato analogicamente: **Valore digitale rappresentato su 3 cifre:**

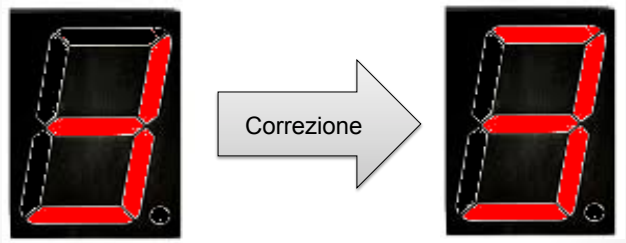
Informazione: "Oggi il mio peso è 82.310 Kg"; **Tipo:** $\mathbb{R}^+$**; Attributo:** "Peso di oggi";
Valore rappresentato analogicamente: **Valore digitale rappresentato su 3 cifre:**

Informazione derivata da elaborazione: "Oggi sono dimagrito ?? Kg".
Come la rappresentazione del valore ha effetti sull'elaborazione dell'informazione?

Rappresentazioni digitali di informazioni di tipo non finito introducono errori che devono essere opportunamente gestiti nelle operazioni di calcolo!
Es.: sono dimagrito 0.03 Kg!

Perché una rappresentazione digitale?

- La **rappresentazione digitale** permette di conservare e trasmettere le informazioni **in maniera più robusta** di una rappresentazione analogica, perché i sistemi digitali possono facilmente **correggere interferenze** che modificherebbero irreversibilmente una rappresentazione analogica.



- I calcolatori digitali non operano su informazioni continue! Affinché un elaboratore possa usare queste informazioni è necessario prima trasformarle in discrete per permettere l'uso di rappresentazioni digitali.

Codifica e decodifica

- La **codifica** è la tecnica adoperata per trasformare un'informazione in una sua rappresentazione (dato), la **decodifica** è l'operazione inversa.
- La **codifica** è definita da:
  - Un **alfabeto origine** $T = (x_1, \ldots, x_n)$ dei dati (valori) da codificare.
  - Un **alfabeto in codice** $E = (a_1, \ldots, a_k)$ che identifica un insieme di simboli.
  - **Un insieme** P **di parole-codice, ovvero di simboli o sequenze di due o più simboli (stringhe) dell'alfabeto in codice.**
  - **Una funzione chiamata tabella-codice che associa a ciascun elemento dell'alfabeto origine una parola codice:** $T \rightarrow P$
- Con il termine codice si intende qualche volta una parola-codice, qualche volta l'insieme di tutte le parole codice (o la tabella-codice).

Codifica: esempio

Codifichiamo i numeri interi tra 1 e 10 in numeri romani.

- **Alfabeto origine:** $T = (1, 2, 3, 4, \ldots, 10)$.
- **Alfabeto in codice:** $E = (I, V, X, L, D, C, M)$
- **Parole-codice:** $P = (I, II, III, IV, V, VI, \ldots IX, X)$.
- **Tabella-codice:** rappresentiamo la funzione in maniera tabellare.

	Parole-Codice (P)										
Alfabeto origine (T)		I	II	III	IV	V	VI	VII	VIII	IX	X
	1	•									
	2		•								
	3			•							
	4				•						
	5					•					
	6						•				
	7							•			
	8								•		
	9									•	
	10										•

Parole Codice (1/2)

- Con **cardinalità del codice** (n) si intende la **cardinalità dell'alfabeto in codice**, ovvero il numero di simboli distinti che possono essere adoperati per creare parole-codice.
- Un codice è detto a **lunghezza fissa** se le sue parole-codice hanno tutte la stessa lunghezza (l), altrimenti è detto a **lunghezza variabile**.
- **Per ogni coppia** (n, l) **è possibile generare al più un numero di parole-codice pari a** n^l, ottenute dall'applicare l volte il prodotto cartesiano ($\times$) sui simboli dell'alfabeto in codice.
  - Es.: Alfabeto in codice: $E=(a, b, c)$, $n = |E| = 3$
    - Con $l = 1$ posso al più avere tante parole-codice quanti sono i simboli dell'alfabeto in codice, ovvero n.
    - Con $l = 2$ posso al più generare tante parole codice quante sono le combinazioni con ripetizione degli n simboli nelle due posizioni, ovvero n^2: $E \times E = \{ (a, a), (a, b), (a, c), (b, a), (b, b), \ldots, (c, c) \}$.
    - Con $l = 3$ al più ho n^3 parole codice (prod. cartesiano $A \times A \times A$).

Parole Codice (2/2)

- Dall'analisi, risulta che se l'alfabeto origine ha m elementi, per definire una tabella-codice necessariamente: $n^l \geq m$ (affinché le colonne della tabella-codice siano almeno quanto le righe, cioé l'alfabeto origine).
- Ne risulta che:

> Per codificare un dato di cardinalità m mediante un alfabeto di n simboli è necessaria una stringa di lunghezza minima l_{min}, con:
> $l_{min} = \lceil \log_n m \rceil$

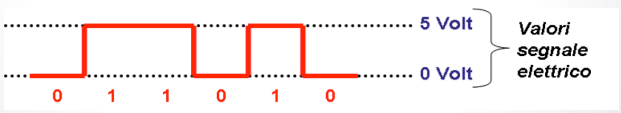
- Se $n^l > m$ è possibile generare parole-codice che non sono messe in relazione con l'alfabeto origine (più colonne che righe).

Codice binario

- Ai fini informatici assume particolare interesse la rappresentazione binaria digitale
  - basata su un alfabeto costituito da due soli simboli distinti, che assumono convenzionalmente la forma di “0” e “1”.
  - Tali due simboli rappresentano le unità minime di rappresentazione e memorizzazione digitale e vengono denominate bit da “binary digit”.
- È un **codice binario** ogni codifica che utilizza un alfabeto in codice composto da solamente due simboli (rappresentati, tipicamente, come (0, 1), o anche: (OFF, ON), (FALSE, TRUE), (SPENTO, ACCESO), etc.)
- Le parole-codice di un codice binario sono rappresentate da stringhe composte esclusivamente dai due simboli
  - Es.: 0011, 1100, 1010, 1111.

Perché binaria

- La rappresentazione digitale, semplifica la memorizzazione delle informazioni e rende i sistemi digitali meno soggetti ai disturbi elettrici rispetto ai sistemi analogici.
  - La rappresentazione delle informazioni all’interno dell’elaboratore si basa sull’alfabeto binario {0,1} in quanto i supporti di memorizzazione delle informazioni (i registri di memoria), sono realizzati con componenti elementari detti **flip-flop**, che operano in due soli stati possibili.



Esercizio sui codici binari (1/3)

- Creare un codice binario a lunghezza fissa l che permetta di rappresentare la seguente informazione, scegliendo il minimo valore possibile per l:
- **Informazione: "**La prima nota della canzone è Sol".
  (Si trascurino le alterazioni e le durate delle note).
  - **Tipo**: note musicali, senza dettagli su alterazioni e durata.
  - **Valore: "**Sol".
  - **Attributo:** "Prima nota della canzone".

Esercizio sui codici binari (2/3)

Alfabeto origine: T = (Do, Re, Mi, Fa, Sol, La, Si),
Alfabeto in codice: E = (0, 1)
Lunghezza minima del codice: $l_{min} = \lceil \log_n m \rceil = \lceil \log_2 7 \rceil = \lceil 2.8\ldots \rceil = 3$
Parole-codice generabili per (n=2, l=3)**:** P = (000, 001, 010, 011, 100, 101, 110, 111); $|P| = n^l = 8$
Una possibile tabella-codice, rappresentata tabularmente:

		Parole-Codice (P)							
		000	001	010	011	100	101	110	111
Alfabeto origine (T)	Do	•							
	Re		•						
	Mi			•					
	Fa				•				
	Sol					•			
	La						•		
	Si							•	

Siccome $|P| > |T|$ esistono parole-codice generabili che non sono messe in relazione con alcun simbolo dell'alfabeto origine, ovvero non rappresentano alcun valore!

Esercizio sui codici binari (3/3)

Definito un codice, è possibile applicarlo in sequenza per rappresentare stringhe di valori dello stesso tipo!

Informazione: Le prime tre note della canzone sono “Sol, Fa, Mi”.

Tipo: Una sequenza di tre note musicali (trascurando le alterazioni e le durate).
Valore: (Sol, Fa, Mi).
Attributo: “prime tre note della canzone”.

Essendo il valore una stringa di valori elementari, possiamo applicare in sequenza la codifica che abbiamo precedentemente definito:

Convenendo un codice a lunghezza fissa sappiamo immediatamente separare le singole parole-chiave che compongono il nostro messaggio: (100, 011, 010).

Rappresentazione in macchina dei dati

- In un calcolatore i dati sono memorizzati in apparati chiamati **registri.**
- Un registro può assumere ***k*-stati** (ovvero, configurazioni) che sono messi in relazione a ***k* valori** che un dato può assumere.
- Componendo più registri, si compongono i loro stati. Ciò permette di memorizzare dati con cardinalità maggiore.
- In un calcolatore (per motivi tecnologici) si impiegano **registri elementari bistabili** (chiamati **flip-flop**) che memorizzano informazioni a due soli valori (rappresentati da 0 e 1), i **bit (binary digit)**.

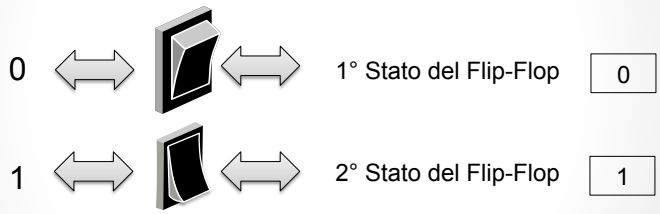
Registro a due stati
{ON, OFF}

Registro a quattro stati
{{ON, ON}, {ON, OFF}, {OFF, ON}, {OFF, OFF}}

Registro a otto stati
{{ON, ON, ON}, … }

La codifica binaria e i registri

- Dal momento che il registro elementare dei calcolatori attuali è composto da elementi bistabili, i calcolatori necessariamente impiegano una codifica binaria per rappresentare le informazioni.
- I due simboli di cui sono composti i codici binari sono infatti messi in relazione con la coppia di stati che possono assumere i flip-flop.

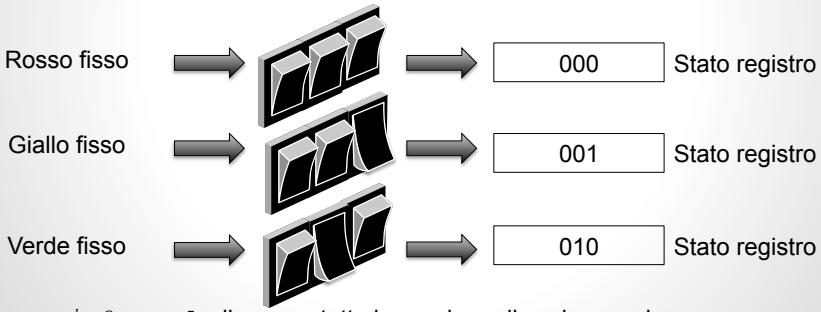


Usando una codifica binaria siamo capaci di rappresentare qualsiasi informazione in un calcolatore, associando le stringhe binarie allo stato interno dei registri elementari!

Esempio codifica binaria: il tipo Semaforo

- **Tipo Semaforo:** {"Rosso fisso", "Giallo fisso", "Verde fisso", "Giallo lampeggiante", "Spento"}.
- Considerando un alfabeto in codice di due simboli (che possiamo rappresentare come {0, 1} o { , } per codificare i valori di un semaforo abbiamo bisogno di un codice di lunghezza l:

$$l > l_{min} = \lceil \log_n m \rceil = \lceil \log_2 5 \rceil = \lceil \log_2 5 \rceil = \lceil 2.321\ldots \rceil = 3$$



- Siccome $n^l = 8 > m = 5$, allora non tutte le parole-codice che possiamo generare sono messe in relazione con l'alfabeto origine.

Misura dell'informazione

- Grazie alla codifica binaria, ogni flip-flop può rappresentare una informazione che varia tra due valori (appunto, 0 o 1): il **bit** è pertanto definibile come **un tipo di dato di cardinalità 2**.
- Dal momento che è possibile comporre i flip-flop in registri, possiamo anche definire il bit come **una unità di misura dell'informazione.**
  - **Maggiore è la quantità di informazione** (ovvero il numero di scelte tra le quali una informazione può assumere valore - cioé la cardinalità del tipo), **maggiore sarà il numero di bit che dovrò usare per codificare l'informazione stessa**.
  - Es.: Poiché l'informazione di tipo semaforo è codificabile al minimo con 3 bit, la sua quantità d'informazione è 3 bit perché in un calcolatore devo usare almeno 3 flip-flop per rappresentarla.

Byte, Word e Memorie

- Per ragioni legate alla costruzione dei moderni calcolatori, è d'uso fare riferimento a (stringhe di) 8 bit con il termine **byte**.
- I registri sono, in generale, multipli di una **word**, che dipende dalle caratteristiche del sistema, ma è un multiplo del byte: 8, 16, 32, 64 bit
- Organizzazioni di registri prendono il nome di **memorie**.

Misuriamo la quantità di informazione memorizzabile in un organo di memoria usando come unità di misura il bit, i byte o i loro multipli.

Sigla	Nome	Numero byte	Numero bit
B	Byte	1	8
KB	KiloByte	2^{10} = 1˙024 (oppure 10^3)	8˙192
MB	MegaByte	2^{20} = 1˙048˙576 (oppure 10^6)	8˙388˙608
GB	GigaByte	2^{30} = 1˙073˙741˙824 (oppure 10^9)	8˙589˙934˙592
TB	TeraByte	2^{40} = 1˙099˙511˙627˙776 (oppure 10^{12})	8˙796˙093˙022˙208

Le definizioni con base 2 sono storiche,
quelle in base 10 sono conformi al Sistema Internazionale (SI)

Come funziona nei moderni calcolatori

- I processori dei calcolatori moderni lavorano con unità di dati maggiori o uguali ad un byte (8 bit).
  - Ad esempio, nonostante possa codificare l'informazione semaforo con 3 bit, essa sarà rappresentata in un calcolatore con 1 byte (8 bit).
- Per elaborare informazioni di dimensione maggiore si impiegano multipli del byte o delle word.
- In maniera più dettagliata, i moderni calcolatori usano una codifica binaria a lunghezza fissa che adotta parole codice con lunghezza a multipli di 8.

Numero di byte ***b***	Numero di bit (***b****8)	$2^{(b*8)}$	Configurazioni
1	8	2^{8}	256
2	16	2^{16}	65.536
3	24	2^{24}	16.777.216
4	32	2^{32}	4.294.967.296

Codifiche in uso negli elaboratori

- Per elaborare informazioni numeriche, i calcolatori adottano codifiche binarie generalmente a lunghezza fissa, tra cui le seguenti:
  - Codifica di numeri interi relativi $\mathbb{Z}$ in segno e modulo
  - Codifica di numeri interi relativi $\mathbb{Z}$ in complementi alla base
  - Codifica di numeri reali relativi $\mathbb{R}$ secondo lo standard IEEE 754
- Per elaborare informazioni testuali sono comuni le seguenti codifiche:
  - Codifica di caratteri secondo lo standard ASCII (7-8 bit)
  - Codifica di caratteri secondo lo standard Unicode (16 bit)
- Esistono poi numerose codifiche specifiche per elaborare immagini, video e suoni.

Errori di rappresentazione

- Adottando codifiche numeriche finite, i calcolatori possono rappresentare solamente sottoinsiemi degli insiemi numerici infiniti:

$$\mathbb{Z}^* \subseteq \mathbb{Z};\ \mathbb{R}^* \subseteq \mathbb{R}$$

- L'errore di **round-off** è l'errore che si commette nell'approssimare (arrotondare) un elemento di un insieme numerico (es.: $r \in \mathbb{R}$) al valore più vicino rappresentabile (es.: $r^* \in \mathbb{R}^*$).
  - Gli algoritmi numerici devono opportunamente stimare e ridurre gli errori di round-off per garantire risultati accurati.
- Inoltre, in un insieme numerico finito, una operazione di calcolo può generare:
  - Una condizione di **overflow,** quando il risultato dell'operazione supera i limiti dell'intervallo rappresentabile (minore del minimo, o maggiore del massimo).
  - Una condizione di **underflow,** quando il risultato dell'operazione, a causa dell'approssimazione, viene rappresentato con lo zero pur non essendo nullo.

Esempio

- Calcolatrice decimale dotata di sole tre cifre, con intervallo di definizione formato da numeri **interi** compresi nell'intervallo [-999,+999]

Operazione	Condizione
200 +100	risultato rappresentabile
730 + 510	Overflow
-500-720	Underflow
2 : 3	risultato non rappresentabile

Algebra con precisione finita

- L'algebra dei numeri a precisione finita è diversa da quella convenzionale poiché alcune delle proprietà:
  - proprietà associativa: ***a + (b - c) = (a + b) – c***
  - proprietà distributiva: ***a × (b - c) = a × b – a × c***
- non sempre vengono rispettate in base all'ordine con cui le operazioni vengono eseguite

a	***b***	***c***	***a + (b - c)***	condizione	***(a + b) – c***	Condizione
100	900	600	100 + (900 - 600)	ok	(100 + 900) - 600	Overflow
a	***b***	***c***	***a × (b - c)***	condizione	***a × b – a × c***	Condizione
200	90	88	200 × (90 – 88)	ok	200 × 90 – 200 × 88	Overflow

Esempio: Errori di rappresentazione

- Consideriamo una codifica che permette di rappresentare solamente i seguenti elementi di $\mathbb{R}$:

$$\mathbb{R}^* = \{-2, -1.5, -1, -0.5, 0, 0.5, 1, 1.5, 2\}.$$

- L'errore di **round-off** è quello commesso quando approssimo un numero al fine di trasformarlo in uno dei valori rappresentabili:
  - $1.55 \rightarrow 1.5$; Errore round-off = $| r - r^* | = 0.05$
  - $-0.666\ldots \rightarrow -0.5$; Errore round-off = $| r - r^* | = | -2/3 + 1/2 | = 1/6$
  - $1.5 \rightarrow 1.5$; Errore round-off = $| r - r^* | = 0$
- L'**overflow** si verifica quando il risultato di una operazione supera i limiti dell'intervallo rappresentabile:
  - $1.5 + 1.5 \rightarrow$ **overflow**
  - $-2 + -0.5 \rightarrow$ **overflow**
- L'**underflow** si verifica quando il risultato di una operazione rappresenta con lo zero un numero diverso dallo zero:
  - $-0.5 / 2 \rightarrow 0$ **underflow**

La matematica per gli insiemi finiti

- Dal momento che i calcoli avvengono su insiemi finiti, l'informatica fa ampio uso di due branche della matematica:
  - La **matematica discreta**, che studia a "livello teorico" strutture matematiche discrete, piuttosto che continue.
  - Il **calcolo numerico**, che studia a "livello applicativo" come approssimare modelli matematici continui per risolverli con insiemi numerici finiti.

Codifica dei caratteri

- I **caratteri** sono **informazioni per loro natura discrete**, pertanto la rappresentazione di tali informazioni si riduce ad **adottare una codifica condivisa.**
- Esistono numerose codifiche per i caratteri. Le più importanti e diffuse sono:
  - La codifica ASCII (7 bit)
  - Le codifiche sotto il nome di ASCII Esteso (8 bit)
  - Le codifiche UTF-8, UTF-16, UTF-32, basati sullo standard UNICODE

Codice ANSI ASCII (1/2)

- **L'ASCII** (**American Standard Code for Information Interchange**) **nasce alla fine degli anni sessanta** dall'ente americano di standardizzazione ANSI (American National Standards Institute) che fissò una codifica **per consentire anche a calcolatori (all'epoca più per telescriventi) di produttori diversi di poter comunicare tra loro**.
- ASCII codifica caratteri alfabetici, numerici, di punteggiatura, simboli, e anche alcuni codici da usare come controllo della comunicazione tra una macchina e l'altra (per esempio, per segnalare l'inizio o la fine di una trasmissione).
- Essendo concepito per l'USA e per macchine con risorse limitate, ASCII è di 7 bit e riesce a codificare con le 128 parole-codice solamente le lettere dell'alfabeto inglese, escludendo, ad esempio, i caratteri accentati in uso nelle lingue europee (è, é, à, ì, ò, …).

Codice ANSI ASCII (2/2)

USASCII code chart

b_4	b_3	b_2	b_1	Row ↓ / Column →	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
					0	1	2	3	4	5	6	7
0	0	0	0	0	NUL	DLE	SP	0	@	P	`	p
0	0	0	1	1	SOH	DC1	!	1	A	Q	a	q
0	0	1	0	2	STX	DC2	"	2	B	R	b	r
0	0	1	1	3	ETX	DC3	#	3	C	S	c	s
0	1	0	0	4	EOT	DC4	$	4	D	T	d	t
0	1	0	1	5	ENQ	NAK	%	5	E	U	e	u
0	1	1	0	6	ACK	SYN	&	6	F	V	f	v
0	1	1	1	7	BEL	ETB	'	7	G	W	g	w
1	0	0	0	8	BS	CAN	(	8	H	X	h	x
1	0	0	1	9	HT	EM	)	9	I	Y	i	y
1	0	1	0	10	LF	SUB	*	:	J	Z	j	z
1	0	1	1	11	VT	ESC	+	;	K	[	k	{
1	1	0	0	12	FF	FS	,	<	L	\	l	|
1	1	0	1	13	CR	GS	-	=	M	]	m	}
1	1	1	0	14	SO	RS	.	>	N	^	n	~
1	1	1	1	15	SI	US	/	?	O	_	o	DEL

Codifica dei caratteri: lo standard ASCII

Binario	Oct	Dec	Hex	Glifo	Binario	Oct	Dec	Hex	Glifo
010 0000	040	32	20	Spazio	100 0000	100	64	40	@
010 0001	041	33	21	!	100 0001	101	65	41	A
010 0010	042	34	22	"	100 0010	102	66	42	B
010 0011	043	35	23	#	100 0011	103	67	43	C
010 0100	044	36	24	$	100 0100	104	68	44	D
010 0101	045	37	25	%	100 0101	105	69	45	E
010 0110	046	38	26	&	100 0110	106	70	46	F
010 0111	047	39	27	'	100 0111	107	71	47	G
010 1000	050	40	28	(	100 1000	110	72	48	H
010 1001	051	41	29	)	100 1001	111	73	49	I
010 1010	052	42	2A	*	100 1010	112	74	4A	J
010 1011	053	43	2B	+	100 1011	113	75	4B	K
010 1100	054	44	2C	,	100 1100	114	76	4C	L
010 1101	055	45	2D	-	100 1101	115	77	4D	M
010 1110	056	46	2E	.	100 1110	116	78	4E	N
010 1111	057	47	2F	/	100 1111	117	79	4F	O
011 0000	060	48	30	0	101 0000	120	80	50	P
011 0001	061	49	31	1	101 0001	121	81	51	Q
011 0010	062	50	32	2	101 0010	122	82	52	R
011 0011	063	51	33	3	101 0011	123	83	53	S
011 0100	064	52	34	4	101 0100	124	84	54	T
011 0101	065	53	35	5	101 0101	125	85	55	U
011 0110	066	54	36	6	101 0110	126	86	56	V

Le estensioni di ASCII: ASCII Esteso

- Con **ASCII Esteso** si intendono quelle codifiche con parole codice di 8 bit che introducono ulteriori caratteri ad ASCII, adottando codifiche compatibili con esso per i primi 7 bit.
  - Es.: ISO 8859-1 (ISO Latin 1) è una codifica ASCII Estesa per le lingue dell'Europa occidentale.

Codepage 819 - Latin 1 - ISO 8859-1

	-0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-A	-B	-C	-D	-E	-F
8-	0080	0081	0082	0083	0084	0085	0086	0087	0088	0089	008A	008B	008C	008D	008E	008F
9-	0090	0091	0092	0093	0094	0095	0096	0097	0098	0099	009A	009B	009C	009D	009E	009F
A-	00A0	¡ 00A1	¢ 00A2	£ 00A3	¤ 00A4	¥ 00A5	¦ 00A6	§ 00A7	¨ 00A8	© 00A9	ª 00AA	« 00AB	¬ 00AC	- 00AD	® 00AE	¯ 00AF
B-	° 00B0	± 00B1	2 00B2	3 00B3	´ 00B4	µ 00B5	¶ 00B6	· 00B7	¸ 00B8	1 00B9	º 00BA	» 00BB	¼ 00BC	½ 00BD	¾ 00BE	¿ 00BF
C-	À 00C0	Á 00C1	Â 00C2	Ã 00C3	Ä 00C4	Å 00C5	Æ 00C6	Ç 00C7	È 00C8	É 00C9	Ê 00CA	Ë 00CB	Ì 00CC	Í 00CD	Î 00CE	Ï 00CF
D-	Ð 00D0	Ñ 00D1	Ò 00D2	Ó 00D3	Ô 00D4	Õ 00D5	Ö 00D6	× 00D7	Ø 00D8	Ù 00D9	Ú 00DA	Û 00DB	Ü 00DC	Ý 00DD	Þ 00DE	ß 00DF
E-	à 00E0	á 00E1	â 00E2	ã 00E3	ä 00E4	å 00E5	æ 00E6	ç 00E7	è 00E8	é 00E9	ê 00EA	ë 00EB	ì 00EC	í 00ED	î 00EE	ï 00EF
F-	ð 00F0	ñ 00F1	ò 00F2	ó 00F3	ô 00F4	õ 00F5	ö 00F6	÷ 00F7	ø 00F8	ù 00F9	ú 00FA	û 00FB	ü 00FC	ý 00FD	þ 00FE	ÿ 00FF

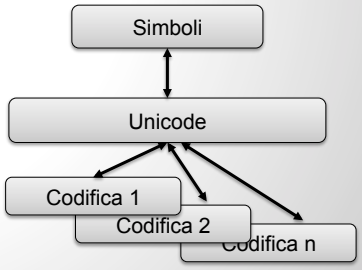
Unicode (1/2)

- **Unicode (Universal Encoding)** è uno standard che si propone di creare una **codifica delle scritture a livello universale**.
  - Cataloga tutti i **simboli usati nei testi** ed assegna a loro **un numero**, in maniera indipendente dalla rappresentazione, alfabeto e lingua.
  - È **compatibile con ASCII** e supporta **tutte le lingue del mondo**, codificando oltre **un milione di simboli**.
- Sullo standard dei numeri assegnati ai simboli Unicode sono poi definite le codifiche concrete, tra cui le **UTF** (**Unicode Transformation Format**).

Faces

1F600	😀	GRINNING FACE
1F601	😁	GRINNING FACE WITH SMILING EYES
1F602	😂	FACE WITH TEARS OF JOY
1F603	😃	SMILING FACE WITH OPEN MOUTH
		→ 263A ☺ white smiling face
1F604	😄	SMILING FACE WITH OPEN MOUTH AND SMILING EYES
1F605	😅	SMILING FACE WITH OPEN MOUTH AND COLD SWEAT

Fonte: http://unicode.org/charts/PDF/U1F600.pdf



Unicode (2/2)

- Creare nuove codifiche basandosi su Unicode permette di **identificare univocamente i simboli a prescindere dalla rappresentazione**, permettendo il dialogo tra tutti i sistemi basati sullo standard.
  - I numeri assegnati da Unicode ai simboli assumono la funzione di una “lingua franca” per tutte le codifiche basate su di esso.
- UTF-32 è un codifica basata Unicode **a lunghezza fissa di 32 bit**.
  - Poiché **la maggior parte dei simboli Unicode sono usati raramente**, è spesso più conveniente adottare una **codifica a lunghezza variabile**.
- UTF-8 e UTF-16 sono codifiche basate su Unicode **a lunghezza variabile**, aventi lunghezza minima di 8 e 16 bit.
  - UTF-16 è la codifica predefinita utilizzata dai moderni sistemi operativi, tra cui Windows e Mac OS X.