Representación de la información

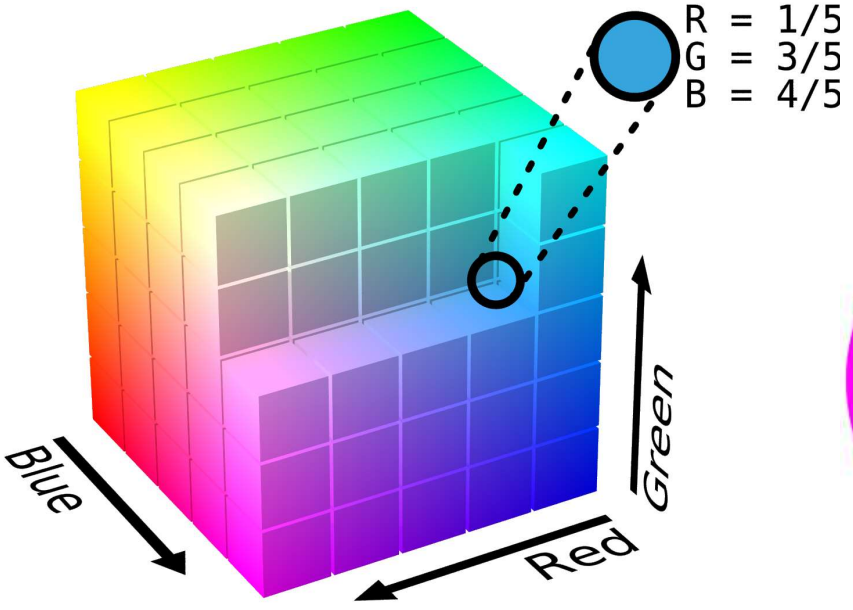
Fernando Bobillo

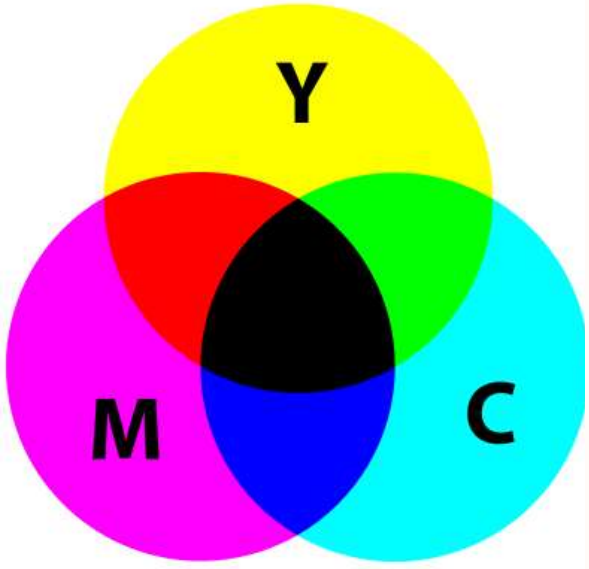
Resumen de contenidos

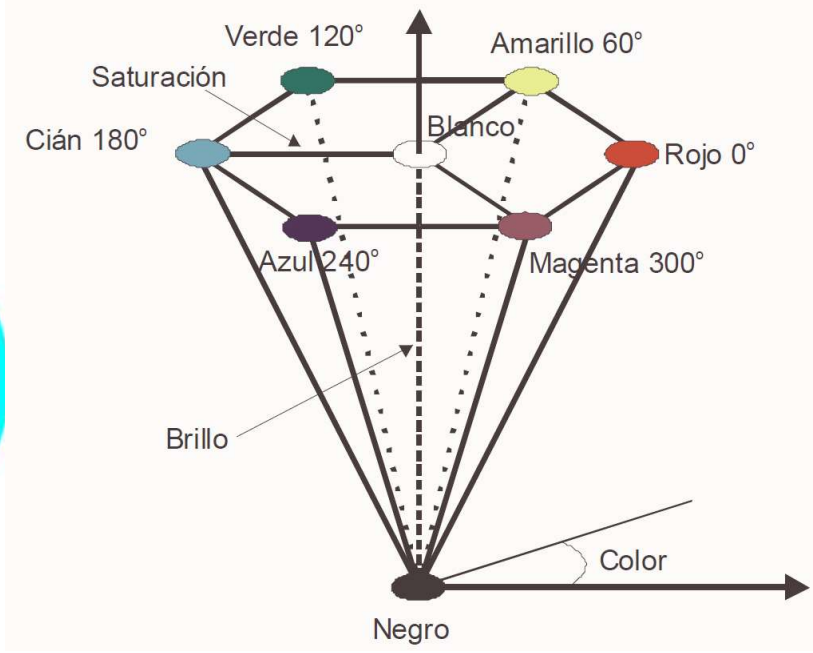
- Representación de la información en ordenadores
  - Colores
  - Caracteres
  - Enteros
  - Reales

Colores

Representación de color: RGB, CMY y HSV







Representación de caracteres: ASCII

- ASCII: entero en [0, 127] (7 bits)

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

Representación de caracteres: EASCII

- ASCII extendido (EASCII): entero en [0, 255] (8 bits)

128	Ç	144	É	160	á	176	▒	192	└	208	╨	224	α	240	≡
129	ü	145	æ	161	í	177	▒	193	┴	209	╤	225	ß	241	±
130	é	146	Æ	162	ó	178	▓	194	┬	210	╥	226	Γ	242	≥
131	â	147	ô	163	ú	179	│	195	├	211	╙	227	π	243	≤
132	ä	148	ö	164	ñ	180	┤	196	─	212	╘	228	Σ	244	⌠
133	à	149	ò	165	Ñ	181	╡	197	┼	213	╒	229	σ	245	⌡
134	å	150	û	166	ª	182	╢	198	╞	214	╓	230	µ	246	÷
135	ç	151	ù	167	º	183	╖	199	╟	215	╫	231	τ	247	≈
136	ê	152	ÿ	168	¿	184	╕	200	╚	216	╪	232	Φ	248	°
137	ë	153	Ö	169	⌐	185	╣	201	╔	217	┘	233	Θ	249	∙
138	è	154	Ü	170	¬	186	║	202	╩	218	┌	234	Ω	250	·
139	ï	155	¢	171	½	187	╗	203	╦	219	█	235	δ	251	√
140	î	156	£	172	¼	188	╝	204	╠	220	▄	236	∞	252	n
141	ì	157	¥	173	¡	189	╜	205	═	221	▌	237	φ	253	²
142	Ä	158	₧	174	«	190	╛	206	╬	222	▐	238	ε	254	■
143	Å	159	ƒ	175	»	191	┐	207	╧	223	▀	239	∩	255	

Representación de caracteres: Unicode

- EASCII solamente añade caracteres de algunas lenguas europeas basadas en alfabeto latino
- Unicode añade caracteres de (casi) cualquier idioma en cualquier alfabeto: cirílico, árabe, hebreo, chino, japonés…
- Inicialmente, 16 bits (65536 caracteres)
- Posteriormente extendido para permitir más de 16 bits
- En Java, el tipo `char` solamente permite representar 16 bits

Sistemas de numeración

Sistemas de numeración

En el juego del mus, la misma piedra vale 1 punto o 5 puntos según la tenga un jugador o su compañero (en este último caso, la piedra se llama amarraco)

Sistemas de numeración

- Un entero K se escribe en base b como $a_n a_{n-1} \ldots a_2 a_1 a_0$
  - $K = a_n\, b^n + a_{n-1}\, b^{n-1} + \ldots + a_2\, b^2 + a_1\, b^1 + a_0\, b^0$
  - $a_i \in \{0, 1, \ldots, b-1\}$
- En base 10 (decimal), $a_i \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$
- En base 2 (binario), $a_i \in \{0, 1\}$
- ¿Y en base hexadecimal?
- Si la base no está clara, se escribe $(a_n a_{n-1} \ldots a_2 a_1 a_0)_b$
- Ejemplo: 12345_{10} =

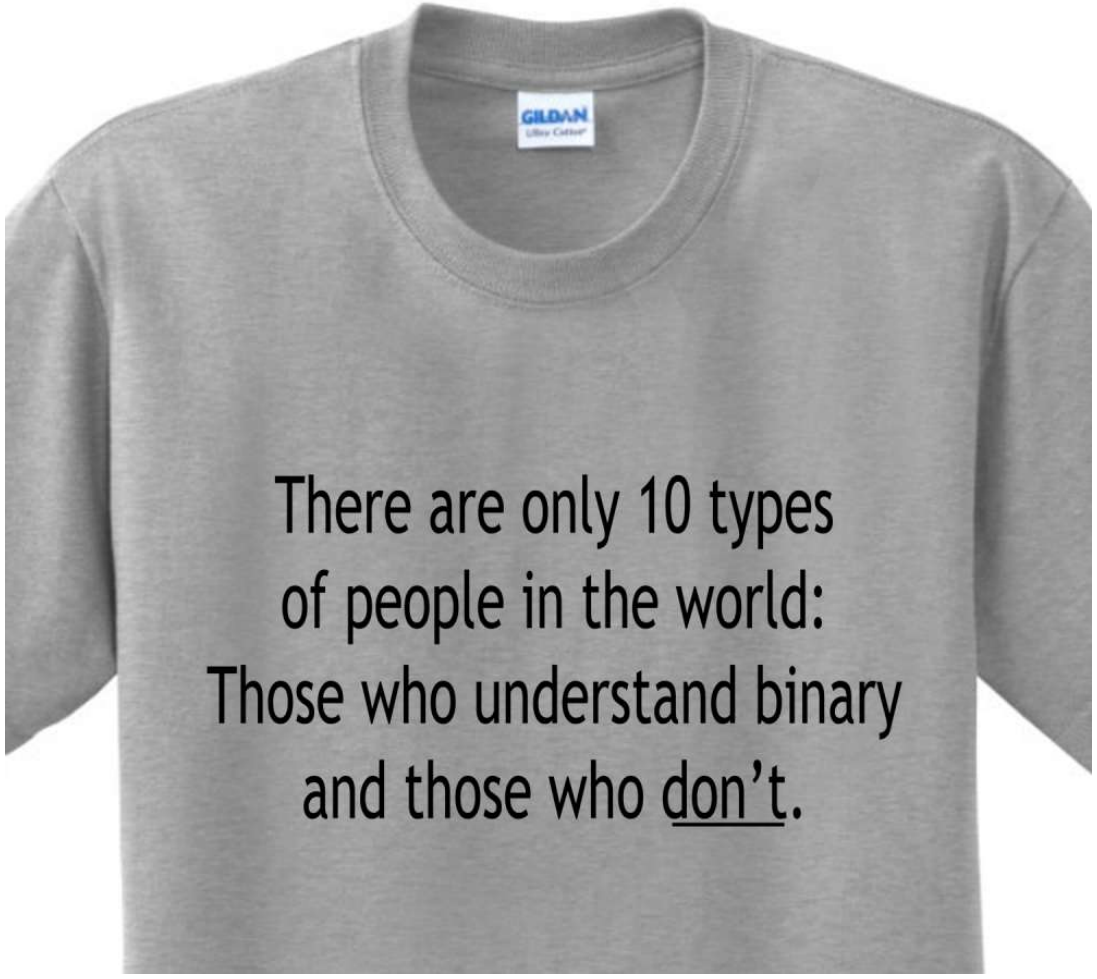
  $1 \cdot 10^4 + 2 \cdot 10^3 + 3 \cdot 10^2 + 4 \cdot 10^1 + 5 \cdot 10^0 =$

  $1 \cdot 10000 + 2 \cdot 1000 + 3 \cdot 100 + 4 \cdot 10 + 5 \cdot 1 =$

  $10000 + 2000 + 300 + 40 + 5$

Representación en binario

- Ejemplo: $10_2 = 1 \cdot 2^1 + 0 \cdot 2^0 = 1 \cdot 2 + 0 \cdot 1 = 2 + 0 = 2_{10}$



Representación en binario

- ¿Qué números se pueden representar con 1, 2, 3 y n bits?

Nº bits	Nº números	Números
1	2	{0, 1}
2	4	{00, 01, 10, 11}
3	8	{000, 001, 010, 011, 100, 101, 110, 111}
n	2^n	$\mathbb{N} \cap [0, 2^n - 1]$

Solamente se representan números positivos, no negativos

Algoritmo para pasar de decimal a base b

```
// Dígito 0: el menos significativo
n := Número a convertir
b := Base a la que convertir n
i := 0
mientras n ≠ 0 repetir
principio
   i-ésimo dígito := n % b
   n := n / b
   i := i + 1
fin
```

- Ejemplo: 20_{10}
  - n = 20
  - Dígito 0: 20 % 2 = 0
  - n = 20 / 2 = 10
  - Dígito 1: 10 % 2 = 0
  - n = 10 / 2 = 5
  - Dígito 2: 5 % 2 = 1
  - n = 5 / 2 = 2
  - Dígito 3: 2 % 2 = 0
  - n = 2 / 2 = 1
  - Dígito 4: 1 % 2 = 1
  - n = 1 / 2 = 0
- Resultado: 10100_2

Algoritmo para pasar de base b a decimal

- Dado un número $(a_n a_{n-1} \ldots a_2 a_1 a_0)_b$, calcular

$$\sum_{i=0}^{n} a_i \, b^i$$

- Ejemplo: 10100_2
- Resultado:

$\sum_{i=1}^{4} a_i \, b^i =$

$0\ 2^0 + 0 \cdot 2^1 + 1 \cdot 2^2 + 0 \cdot 2^3 + 1 \cdot 2^4 =$

$0 + 0 + 4 + 0 + 16 =$

20_{10}

Signo y magnitud

- Extiende la representación anterior permitiendo n^{os} negativos
- Ahora, un número signado de n bits incluye:
  - 1 bit (el más significativo) para signo:
    - 0 si el número es positivo
    - 1 si el número es negativo
  - n - 1 bits para la magnitud (valor absoluto) del número
- Con 3 bits podemos representar 7 números: 000 (+0), 001 (+1), 010 (+2), 011 (+3), 100 (-0), 101 (-1), 110 (-2), 111 (-3)
- Con n bits:
  - $2^n - 1$ números, $[- 2^{n-1} + 1, 2^{n-1} - 1]$

Signo y magnitud

- Desventajas:
  - 2 representaciones distintas para el 0: +0 y -0
  - No permite operar aritméticamente
- Ejemplo: cálculo de $20_{10} + (-97)_{10}$ pasándolos a binario, con 8 bits para representar los números, y haciendo la suma
  - $20_{10} + (-97)_{10} = (-117)_{10} \neq (-77)_{10} = 11110101_2$
  - Suma en binario: como en decimal pero 1 + 1 = 0 y llevo 1

$$\begin{array}{rr} & 00010100 \\ + & 11100001 \\ \hline & 11110101 \end{array}$$

Complemento a dos

- Es la utilizada en Java
- Números negativos en complemento a 2: $C2(x) = 2^n - x$
- Cálculo rápido: intercambiar unos y ceros y sumar 1
  - $C2(-97_{10}) = C2(01100001_2) = 10011110 + 1 = 10011111$
  - $C2(-3_{10}) = C2(011_2) = 100 + 1 = 101$
- Si empieza por 0 es positivo; si empieza por 1 negativo
- Con 3 bits podemos representar 7 números: 000 (0), 001 (+1), 010 (+2), 011 (+3), 100 (-4), 101 (-3), 110 (-2), 111 (-1)
- Rango con n bits: 2^n números, $[-2^{n-1}, 2^{n-1} - 1]$
  - Representación asimétrica: un número negativo más
- Permite operar aritméticamente

Complemento a dos

- Comprobemos que permite operar aritméticamente
- Ejemplo: cálculo de $20_{10} + (-97)_{10}$ pasándolos a binario, con 8 bits para representar los números, y haciendo la suma
  - $20_{10} + (-97)_{10} = (-77)_{10} = 11110101_2$
  - $C2(-97_{10}) = C2(01100001_2) = 10011110 + 1 = 10011111$
  - $10110011 = 10110010 + 1 = C2(01001101) = (-77)_{10}$

$$\begin{array}{rr} & 00010100 \\ + & 10011111 \\ \hline & 10110011 \end{array}$$

Representación de reales en base b

- Un real K se escribe en base b como $a_n a_{n-1} \ldots a_2 a_1 a_0 . d_1 d_2 \ldots d_m$
  - $K = a_n b^n + \ldots + a_1 b^1 + a_0 b^0 + d_1 b^{-1} + d_2 b^{-2} + \ldots + d_m b^{-m}$
  - $a_i, d_j \in \{0, 1, \ldots, b-1\}$
- Ejemplo: 12345.67_{10} =

  $1 \cdot 10^4 + 2 \cdot 10^3 + 3 \cdot 10^2 + 4 \cdot 10^1 + 5 \cdot 10^0 + 6 \cdot 10^{-1} + 7 \cdot 10^{-2} =$

  $1 \cdot 10000 + 2 \cdot 1000 + 3 \cdot 100 + 4 \cdot 10 + 5 \cdot 1 + 6 \cdot 0.1 + 7 \cdot 0.01 =$

  $10000 + 2000 + 300 + 40 + 5 + 0.6 + 0.007$
- El cambio de base se realiza convirtiendo por separado las partes entera y decimal

Paso de la parte decimal a binario

```
// Dígito 1: el más significativo
n := Número real a convertir
n := n - parteEntera(n)
i := 1
mientras n ≠ 0 repetir
principio
    n := n * 2;
    i-ésimo dígito := parteEntera(n)
    n := n - parteEntera(n)
    i := i + 1
fin
```

- Ejemplo: 0.6875_{10}
  - n = 0.6875 · 2 = 1.3750
  - n = 0.3750 · 2 = 0.75
  - n = 0.75 · 2 = 1.5
  - n = 0.5 · 2 = 1
  - n = 0
- Resultado: 0.1011_2

Paso de la parte decimal en binario a base 10

- Dado una parte decimal número $(d_1 d_2 \ldots d_m)_2$, calcular

$$\sum_{j=1}^{m} d_j \, 2^{-j}$$

- Ejemplo: 0.1011_2
- Resultado:

$\sum_{j=1}^{4} d_j \, 2^{-j} =$

$1 \cdot 2^{-1} + 0 \cdot 2^{-2} + 1 \cdot 2^{-3} + 1 \cdot 2^{-4} =$

$1/2 + 0 + 1/8 + 1/16 =$

0.6875_{10}

Paso de la parte decimal a binario

- Ejemplo: $0.1_{10} = 0.0\overline{0011}_2$
  - $0.1 \cdot 2 = 0.2$
  - $0.2 \cdot 2 = 0.4$
  - $0.4 \cdot 2 = 0.8$
  - $0.8 \cdot 2 = 1.6$
  - $0.6 \cdot 2 = 1.2$
  - $0.2 \cdot 2 = 0.4$
  - $0.4 \cdot 2 = 0.8$
  - $0.8 \cdot 2 = 1.6$
  - $0.6 \cdot 2 = 1.2$
  - …

Paso de la parte decimal en binario a base 10

- Ejemplo: $0.0\ 0011\ 0011\ 0011_2$ =

$$0 + 0 \cdot 2^{-1} + 0 \cdot 2^{-2} + 0 \cdot 2^{-3} + 1 \cdot 2^{-4} + 1 \cdot 2^{-5} + 0 \cdot 2^{-6} + 0 \cdot 2^{-7} + 1 \cdot 2^{-8} + 1 \cdot 2^{-9} + 0 \cdot 2^{-10} + 0 \cdot 2^{-11} + 1 \cdot 2^{-12} + 1 \cdot 2^{-13} =$$

$$0.0999755859375_{10}$$

- ¡Error de aproximación! $0.0999755859375 \neq 0.1$