ÍNDICE

1. INTRODUCCIÓN

Informática: El término 'informática' proviene de la fusión de los términos "INFORmación" y "autoMÁTICA", y se define como la ciencia que estudia el tratamiento automático y racional de la información, así como el conjunto de técnicas, métodos y máquinas aplicadas a dicho tratamiento. La Real Academia Española de la Lengua da la siguiente definicion: "Conjunto de conocimientos científicos y técnicas que hacen posible el tratamiento automático de la información por medio de computadoras electrónicas". De esta última definición podemos deducir que hay tanto una ciencia informática como unas técnicas informáticas.

Sistemas informáticos: Conjunto de elementos interconectados o relacionados para el tratamiento de información. El más básico es un ordenador típico. Los más complejos son las redes, sistemas de procesamiento en paralelo,...

Información: Comunicación o adquisición de conocimientos que permiten ampliar o precisar los que se poseen sobre una materia determinada. Podría entenderse que si no se consigue alguna de las dos finalidades señaladas, no habría información, pero es prácticamente imposible que no concurra alguna de ellas cuando un ser humano se encuentra ante una exposición de conocimientos. Para que la información sea la adecuada se tendrán que cumplir unos cuantos requisitos: precisión, significatividad, etc., que se expondrán más adelante.

Desde el punto de vista operativo y temporal, los ordenadores nacieron como herramientas que servían para el procesamiento de datos, como los primitivos procesadores que trabajaban con tarjetas perforadas mediante las que se les introducían datos e instrucciones, y producían datos procesados también en forma de tarjetas que se clasificaban para la siguiente fase. Fueron relativamente frecuentes en grandes corporaciones y centros de cálculo como el MIT (*Massachussets Institute of Technology*).

A medida que los datos se iban haciendo más y más complejos, mezclándose estructuras y añadiendo otras nuevas, apareció la necesidad de diseñar métodos para el procesado de la información, por lo que nacieron los primeros programas y aplicaciones informáticas, medianamente complejos, en lenguajes como el lenguaje máquina o el ensamblador. En la actualidad, la complejidad que existe en las bases de datos y conocimientos impone una tendencia hacia el desarrollo y fabricación de sistemas expertos para ciertas áreas específicas, como es el caso de la educación, la legislación, la sanidad, etc.

Las líneas de trabajo para un futuro a medio plazo se encaminan hacia la consecución de la inteligencia artificial, conocida como *AI* (*Artificial Intelligence*), en la que los sistemas no sólo son capaces de aprender sino de elegir el método más adecuado para hacerlo.

2. DATOS, TIPOS, REPRESENTACIÓN Y PROTECCIÓN

El funcionamiento de un sistema informático se puede asemejar al de una caja de entradas y salidas, uno de los modelos más simples para la interpretación del medio físico. Para este caso se pueden considerar dos tipos de objetos de trabajo: los datos y las instrucciones. Los datos son los caracteres y valores que son necesarios para el funcionamiento del sistema (ver punto anterior), mientras que las instrucciones señalan qué operaciones y procesos deben llevarse a cabo con esos datos.

Así, el ordenador permite la recepción de datos entrantes, que se procesan según indican las instrucciones que posee el sistema. Por último, proporciona unos datos de salida que son los resultados.

2.1 Definición de dato

La palabra datos proviene del latín *datum* (plural *data*) que significa “lo que se da”, en el sentido de “lo que acontece”. El diccionario de la Real Academia de la Lengua Española dice que los datos son: “antecedentes necesarios para llegar al conocimiento exacto de una cosa o para deducir las consecuencias legítimas de un hecho”.

Los datos suelen ser magnitudes numéricas directamente medidas o captadas, pero también pueden ser nombres o conjuntos de símbolos; o valores cualitativos; o frases enteras, premisas, imágenes, sonidos, colores, etc.

Los datos, como la información, se representa mediante secuencias de símbolos. Por ejemplo, en nuestra vida diaria representamos las palabras mediante letras tomadas de nuestro alfabeto. Éste es simplemente uno entre los muchos alfabetos existentes. En base a un alfabeto cualquiera que establecemos por un acuerdo cultural, podemos representar cualquier información compuesta de palabras y cantidades numéricas, y así el que lee entenderá al que escribe.

Desde un punto de vista más operativo, un dato consiste en una información que ha sido preparada, frecuentemente en un formato particular, para un propósito específico. En el ámbito de la informática tiene tres acepciones diferentes.

En un programa informático, los datos pueden distinguirse de las instrucciones. Por ejemplo, en el código fuente las declaraciones de datos son diferentes de las declaraciones ejecutables. Así, en el momento de la ejecución, el espacio de almacenamiento se divide entre los datos -ya sean constantes o variables- y las instrucciones. Y los archivos de datos se distinguen de los archivos de programa.

En segundo lugar, en el contexto de un programa individual o en el de un equipo de programas, dato puede usarse en un sentido más restrictivo significando la entrada de información -inputs- frente a los resultados o salida de la misma -outputs-; como es el caso de la definición de datos o su diseño.

Por último, y de manera más general, se usa la palabra dato para separarla cada vez más de otros aspectos de los programas modernos como la voz, el texto o la imagen. Este uso enfoca sobre la naturaleza altamente formateada de los datos en las aplicaciones tradicionales destinadas al proceso de datos, como concepto opuesto a estructuras más libres como el texto en lenguaje natural, las comunicaciones de voz y las imágenes visuales, muy típicas de aplicaciones en entorno visual y/o multimedia.

2.2 Tipos de datos

Se pueden considerar tres tipos de datos, según el punto del proceso en el que se encuentren:

- Datos de entrada: son los que llegan al ordenador a través de alguno de los periféricos de entrada, tales como el teclado, lectores, etc.; o bien llegan desde unidades de almacenamiento, como son los discos. A veces este concepto se confunde con captura de datos -actividad de introducción de datos, casi siempre automatizada, en el que la recepción de los datos tiene una

importancia secundaria-, y con la preparación de los datos -labor que supone preparar los datos para su entrada en el sistema, formateándolos o codificándolos-.
- Datos intermedios: son los resultados que se van produciendo y que no forman parte de la salida porque no se especificó de esa manera en el diseño del programa.
- Datos de salida: Son los datos resultados del procesamiento de los datos de entrada y de los intermedios. La forma de obtenerlos para su análisis es por medio de un periférico de salida, como son las pantallas o las impresoras; o bien almacenarlos.

Para poder llevar a cabo este trabajo, el ordenador deberá contar con un intérprete, es decir, un sistema fijo y consistente que permita pasar un número o un carácter a un valor en bytes. Una vez que el ordenador ha utilizado ese valor para el procesamiento y ha obtenido un resultado -en bytes-, deberá usar el intérprete de nuevo, pero de manera inversa, para pasar esos bytes a una representación habitual para el operador. El proceso por el cual se pasa de un lenguaje máquina a un lenguaje comprensible por un usuario, o por otra máquina, se le conoce como traducción.

2.3 Representación de los datos

Este proceso consiste en tomar los datos tal como los maneja la máquina y, mediante una traducción previa de los mismos, convertirlos en datos legibles por el operador del sistema con la finalidad de plasmarlos en un medio que permita su lectura.

El periférico más utilizado para la representación de datos es el monitor, seguido por la impresora.

La representación concluirá, la mayor de las veces, con una salida de naturaleza muy similar a los datos con los que se trabaja desde el principio. Por ejemplo, si un usuario está manejando gráficos con un programa de diseño, lo más habitual es concluir con un gráfico en la pantalla o en la impresora. Si se trabaja con datos numéricos, o textos, o documentos, ocurre de manera similar. Es decir, la naturaleza de los datos de entrada y de salida suele ser la misma.

En otras ocasiones ocurrirá de manera diferente, pues un usuario puede introducir números en un ordenador y obtener una salida textual o gráfica.

2.4 Encriptación o protección de datos

Se trata de un proceso por el cual un mensaje -tal como un texto o un gráfico- se protege para que las personas que no estén autorizadas a recibirlo no puedan acceder a la información que contiene. Esta actividad se denomina criptografía, y la ciencia que la estudia y la desarrolla se denomina criptología.

Al mensaje original se le da el nombre de mensaje plano o texto plano, mientras que al mensaje convertido se le conoce como clave o cifra, texto cifrado o código.

En su forma más simple, el remitente y el destinatario poseen copias idénticas de una clave secreta, y además poseen un algoritmo o procedimiento matemático para generar secuencias pseudoaleatorias de bits que representan el mensaje. En una primera fase, el remitente utiliza ese algoritmo para aplicarlo a su mensaje y concluir en el documento cifrado. El remitente, aplicando la misma clave y el mismo algoritmo, realiza el procedimiento inverso y recrea el documento original.

Otro sistema alternativo es el del sistema del libro de claves, por el cual el remitente y el destinatario poseen copias de una tabla de sustitución secreta., la cual se mecaniza por su inclusión en un sistema informático que permite la traducción de mensajes con rapidez.

3. SISTEMAS DE CODIFICACIÓN

Un alfabeto no es más que un conjunto, fijado por acuerdo cultural, de símbolos elementales en base a los cuales se forma la información.

Es importante recalcar la arbitrariedad de cualquier alfabeto porque si la informática ha logrado el tratamiento automático de la información con máquinas, ha sido gracias a este concepto. No es necesario que el alfabeto que usa una máquina en su interior sea el mismo que el que utiliza el hombre que la ha construido y la maneja, basta con que la traducción de los símbolos internos a los externos o viceversa se efectúe de una manera cómoda, y a ser posible (y lo és) automáticamente por la propia máquina.

Cuando una información que originalmente venía representada en un alfabeto A1 es transcrita a un segundo alfabeto A2, se dice que ha sido codificada. Así, se puede definir un código como una representación biunívoca de la información de tal forma que a cada una de las unidaes de información se le asigna una combinación de símbolos determinada. Un ejemplo clásico es el código Morse empleado en los inicios de la telegrafía.

La acción de codificar, es decir la codificación, se puede definir como el proceso por el cual se transforma una información simbólica -el alfabeto fuente- en otra distinta -el alfabeto destino-, sin pérdida de información.

La primera fase se denomina codificación, mientras que la segunda fase se llama decodificación. Los programas que permiten realizar cada fase se conocen como codificador y decodificador, respectivamente, y pueden ser funcionales en forma hardware o software. Estas herramientas suelen consistir en un procedimiento de tipo algorítmico. Cuando existe un aparato que realiza estas tareas se le llama codificador-decodificador o, abreviadamente, y en el argot informático, por codec.

No confundir estos términos con los que se aplican para referirse al lenguaje con el que se escriben algunos programas, como es el caso del código máquina y el código fuente.

Conviene hacer un inciso y refrescar los distintos tipos de caracteres que se pueden encontrar en los componentes lógicos de los sistemas informáticos. Estos caracteres incluyen los alfanuméricos ingleses, los especiales y los caracteres de operación, incluidos todos ellos en los denominados caracteres gráficos. Por otro lado se encuentran los caracteres de control. Desde el punto de vista de la pantalla, se distinguen unos de otros porque los caracteres gráficos dejan algún tipo de marca impresa en la pantalla o un espacio en blanco, mientras que los caracteres de control producen un efecto particular.

3.1 Tipos de código

Según su morfología, se pueden dividir en dos tipos, numéricos y alfanuméricos:

- Representación numérica: como cadenas de números que tienen un significado numérico o textual. Los más comunes son el código binario, el hexadecimal (base 16), el octal (o bytes, base 8), y el de coma flotante. El que cotidianamente usamos es el decimal, en el que los datos se presentan por cadenas de cifras cuyos dígitos sólo varían entre 0 y 9.
- Representación alfanumérica: que son los que representan el alfabeto, los números, caracteres sintácticos, y caracteres especiales como los de control del ordenador.

3.1.1 Representación por medio de números

Normalmente, como se verá en el siguiente apartado, el valor de una cifra dependerá de los dígitos que contiene, repetidos o no, y de la posición que ocupan. Es decir, los sistemas numéricos de representación se basan en la posición de los dígitos en el seno de una cifra.

Es decir, un número X estará formado por un conjunto de dígitos, del siguiente modo:

$X = x_k\ x_{k-1}\ldots x_1\ x_0\ x_{-1}\ldots x_{-j}$

El subíndice indica la posición. Según este subíndice, cada posición tiene un peso que se conoce por ser potencia de la base de numeración. Llamando b a la base de numeración, se tiene la siguiente fórmula general para obtener el valor decimal de un número en cualquier base b:

$$X=\sum_{i=k}^{-j} x_i b^i = x_k b^k + x_{k-1} b^{k-1} + \cdots + x_0 b^0 + x_{-1} b^{-1} + \cdots x_{-j} b^{-j}$$

A partir de aquí, para obtener un nuevo sistema de numeración, solamente hay que elegir una base b, cumpliéndose, además, que los coeficientes (los dígitos que forman el número en una base b) cumplen:

$0 \le X_i < b$

Para indicar que un número está en una base numérica concreta, se termina el número con un paréntesis o un corchete que contiene la base. También se usa un sufijo. Por ejemplo, el número 2041 en código octal, se podría poner como:

$2041_{(8}$

o utilizar la siguiente tabla para añadirle el sufijo

Letras y códigos correspondientes

Sufijo	Base	Ejemplo
B	2	01111101b
o, q	8	175q
h	16	7Dh
d	10	123d

3.1.1.1 El código decimal

Consiste en representar cantidades por medio de cadenas de cifras, en los cuales los dígitos varían entre 0 y 9, ambos inclusive. Se piensa que es el método más natural y primitivo de cuantificación y representación dado que nuestros antepasados podían ayudarse de los dedos para proceder a las cuentas.

La introducción del 0 y, por lo tanto, la extensión hacia los números negativos fue obra de los hindúes, en una época contemporánea a la de Cristo. Los árabes, que tras la invasión de la

península ibérica trajeron su cultura, introdujeron en el mundo occidental esa numeración con algunos avances debidos a sus propios pensadores.

Los números, dentro de una cifra, están ubicados en una posición. Esa posición les asocia con una cantidad que es potencia de 10. Por ejemplo, 1234 en base 10 quiere decir:

$1\times103 + 2\times102 + 3\times101 + 4\times10$

Para indicar explícitamente que el número 1234 está en base 10, se representaría así: $1234_{(10}$.

Lo que se ha comentado hasta ahora del sistema decimal se puede resumir diciendo que es un sistema de numeración posicional, como ya se adelantó, lo que quiere decir que el valor de una cifra depende de la posición en la que se encuentre.

Otro ejemplo en base 10 (b=10), ahora con parte decimal:

$174'25_{(10} = 1\times10^2 + 7\times10^1 + 4\times100 + 2\times10^{-1} + 5\times10^{-2}$

En la actualidad este sistema es el adoptado por todos los países con escasas y puntuales excepciones.

3.1.1.2 Sistema binario

Cuando los símbolos a codificar (alfabeto A1) son transcritos a secuencias de un alfabeto (alfabeto A2) que sólo tiene dos símbolos, diremos que tenemos un sistema de codificación binaria. Estos sistemas son especialmente importantes en informática, pues son los que se usan habitualmente. El motivo para usar un alfabeto de tan sólo dos símbolos es de tipo técnico, como ya se verá.

En este caso, la base del sistema es el número 2. Funciona por tanto igual que el decimal, pero las potencias de cada cifra son potencias de base 2. De esta forma, el sistema de base 2 o código binario requiere sólo 2 dígitos diferentes, el 0 y el 1.

Número en base 10	Número en base 2
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Como se puede ver en esta tabla, y dado que 2 elevado a 3 (23 = 8) llega hasta 8, para representar los primeros 10 números decimales, necesitaremos 10 cifras con cuatro posiciones. Así, el 9 requiere un 23 y además un 1, de ahí que se represente binariamente por 1001.

Para pasar de un número decimal a un número binario sólo se procede a la división del decimal por el número 2 tantas veces como sea posible. El binario será entonces la secuencia de 0 y 1 que se leen desde el último cociente y su resto hacia el primer resto. Por ejemplo, el número 87, en binario, sería:

87 entre 2 = cociente 43 y resto 1
43 entre 2 = cociente 21 y resto 1
21 entre 2 = cociente 10 y resto 1
10 entre 2 = cociente 5 y resto 0
5 entre 2 = cociente 2 y resto 1
2 entre 2 = cociente 1 y resto 0

Entonces, el número binario, empezando por el último cociente y tomando los restos, sería:

1 0 1 0 1 1 1

Por contra, para pasar de un número binario a otro decimal, lo que se debe hacer es multiplicar cada dígito del binario por la potencia de 2 correspondiente a su posición y sumar todos los resultados. Así, con el ejemplo anterior, el resultado sería:

$1 \times 20 + 1 \times 2^1 + 1 \times 2^2 + 0 \times 2^3 + 1 \times 2^4 + 0 \times 2^5 + 1 \times 2^6$

es decir,

1 x 1 + 1 x 2 + 1 x 4 + 0 x 8 + 1 x 16 + 0 x 32 + 1 x 64, o sea,

1 + 2 + 4 + 0 + 16 + 0 + 64, lo que da 87

que es el número que teníamos al principio.

3.1.1.3 Código octal

Se trata de un código que funciona exactamente igual que el decimal o que el binario, pero la base de su numeración es el número 8.

Así, los dígitos que emplea son los que se encuentran entre el 0 y el 7. Los procedimientos para pasar de decimal a octal, y viceversa, son idénticos a los descritos para el caso binario. Por lo demás, sólo queda resaltar que es la base para el cálculo de bytes en el diseño de memorias, transferencia de datos, direccionamientos, etc.

Por ejemplo, el número 1.068, en código octal, será el número

2 0 5 4

y para pasar este número a código decimal de nuevo, se haría

$4 \times 8^0 + 5 \times 8^1 + 0 \times 8^2 + 2 \times 8^3$, es decir,

4 x 1 + 5 x 8 + 0 + 2 x 512 = 4 + 40 + 1024 = 1068

3.1.1.4 Código hexadecimal

La base de esta numeración es el número 16. Aquí se plantea un problema, ¿cómo representar los números que superan al 9 en la representación del código hexadecimal?

Por convenio, se ha establecido que los dígitos de la base que superen al 9 se representarán por medio de letras del alfabeto, consideradas por orden desde la letra A para el 10, es decir, los dígitos 10, 11, 12, 13,...; se representan por las letras A, B, C, D,...

Así, el número 42.424, en código hexadecimal se representaría por:

42.424 entre 16 = cociente 2651 y resto 8
2651 entre 16 = cociente 165 y resto 11
165 entre 16 = cociente 10 y resto 5

es decir, A 5 B 8

y para devolverla a su aspecto decimal, quedaría hacer las operaciones siguientes:

$8 \times 160 + B(=11) \times 161 + 5 \times 162 + 10 \times 163$, que queda

8 x 1 + 11 x 16 + 5 x 256 + 10 x 4.096 = 8 + 176 + 1.280 + 40.960 = 42.424

3.1.1.5 Cambios de base de numeración

Para pasar de una base m a otra base n, se procede como sigue: primero se pasa de base m a base 10 (método conocido) y luego de base 10 a base n (también conocido).

En ocasiones para pasar de una base m a otra base n no es necesario hacer lo que se acaba de indicar, sino que se puede usar un método más simple. Para explicar esto, conviene presentar el concepto de correspondencia entre sistemas: cualquier base de numeración que sea potencia de otra base, como ocurre entre las bases 2, 8 y 16, tienen una correspondencia. De este modo, por ejemplo, podemos:

- Representar cada dígito octal en forma de una combinación de tres dígitos binarios
- Representar cada dígito hexadecimal como cuatro dígitos binarios

De este modo, cuando estemos convirtiendo un número expresado en una base a otra base que sea potencia de la primera, podemos usar la propiedad que se acaba de citar. Haciendo un pequeño esfuerzo de imaginación inversa, se puede pasar de la base mayor a la menor sin problemas, tal como se muestra en estos ejemplos en los que se pasan los números en base 8 y 16 a números en base 2:

3 7 6 (8		
0 1 1	1 1 1	1 1 0

B 3 E 6 (16			
1 0 1 1	0 0 1 1	1 1 1 0	0 1 1 0

3.1.1.6 Trabajo con números decimales

Para representar los números fraccionarios se usa el mismo sistema de trabajo que para los enteros. Lo único a considerar es que el método se extiende para exponentes menores que cero. Por ejemplo, para representar un número decimal en base 2 en representación en base 10, sería:

$0{,}1011_{(2} = 1\times2^{-1} + 0\times2^{-2} + 1\times2^{-3} + 1\times2^{-4} = 1\times1/2 + 0\times1/4 + 1\times1/8 + 1\times1/16 = 0{,}6875(10$

Este es un ejemplo de paso de binario a decimal. Para pasar de decimal a binario, por ejemplo, pues a cualquier base sería igual, pero cambiando el 2 por la base correspondiente, lo que hacemos es multiplicar la fracción decimal repetidamente por la base del nuevo sistema. Los números a la izquierda de la coma en los resultados parciales se escriben como parte del resultado final binario, y se extraen de la operación dejando siempre un cero en su lugar. El

proceso se repite hasta que en la parte decimal de un resultado parcial aparece sólo el dígito 0. Por ejemplo:

0,6875	0,375	0,75	0,5	
× 2	× 2	× 2	× 2	
1,375	0,75	1,5	1,0	0,1011

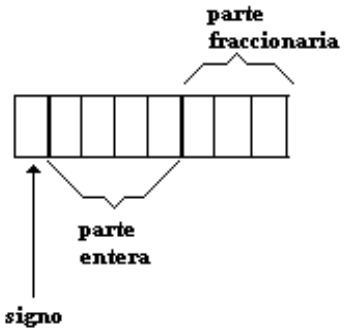
$0,6875_{(10}=0,1011_{(2}$

Cuando un número tenga parte entera y parte fraccionaria, para cambiarlo de base cambiaremos por separado la parte entera y la parte fraccionaria, cada una de ellas con el método correspondiente.

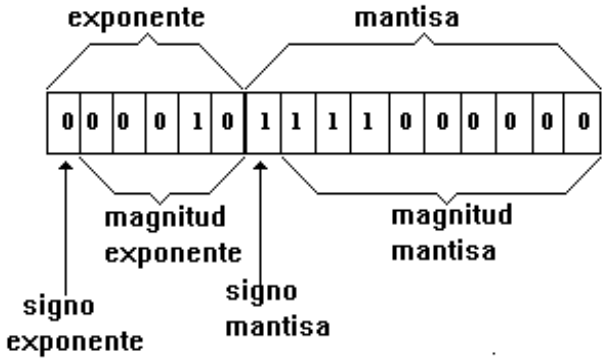
3.1.1.7 Representación de números reales

Existen dos opciones, según cómo se quiera representar la existencia de un signo negativo o no:

- Representación en coma fija: Se parte de un número fijo de bits para representar la parte entera del número, y otro número fijo de bits para representar la parte fraccionaria. Esos números fijos de bits tienen que llegar de un convenio o consenso entre los miembros de la comunidad científica y/ o técnica. También habrá que reservar un bit para el signo del número.
Un ejemplo, si disponemos de un byte para representar cada número real, podría ser el siguiente: 1 bit de signo, cuatro bits para la parte entera y tres para la parte fraccionaria.



- Representación en coma flotante. Se basa en el principio matemático de que cualquier número real N que esté en una base b se puede expresar como
$N=M\cdot b^E$
donde M recibe el nombre de mantisa y E es un exponente. Gracias a esta propiedad, el problema de representar el número N se reduce a representar M y E (no hace falta representar b, ya que se supone conocida).
Por ejemplo, se puede aplicar lo anterior para el número $38,4_{(10}$:
$38,4=384\cdot10^{-1}=0,384\cdot10^{2}=3,84\cdot10^{1}=3840\cdot10^{-2}$…
y como se puede ver, no existe una representación única de la forma $N=M\cdot b^E$. En realidad, existen infinitas. Siempre se elige una de las dos primeras mostradas: mantisa entera sin ceros a la derecha o mantisa fraccionaria sin ceros tras el punto (son los dos modos normalizados que se usan).
Si se dispone de n bits en total para representar el número, reservaremos p bits para la mantisa y q bits para el exponente (n = p + q).
Por ejemplo, si se dispone de 16 bits para cada número y se usan 6 para el exponente (uno de ellos para el signo del exponente) y 10 para la mantisa (uno de ellos también para el signo de la mantisa) y se quiere representar el número $-3,5_{(10}$, se haría:
$-3,5_{(10}= -11,1_{(2}= -0,111\cdot1010$
como se ve, lo primero que se hace es pasar el número a binario, y luego se obtiene una de las dos representaciones normalizadas. En este caso se ha supuesto que se trabaja con mantisa fraccionaria sin ceros tras el punto. La representación de este número sería la siguiente:



3.1.1.8 Razones para el uso del sistema binario

Como ya se ha comentado, el sistema con el que trabajan los ordenadores es el sistema binario. Las razones para ello son las siguientes:

- Toda la circuitería lógica necesaria para procesar la información en binario (decodificadores, etc.) es relativamente sencilla de diseñar y está sumamente estudiada.
- La aritmética en base 2 es la más fácil de implementar. Las reglas de suma, resta, multiplicación y división son las más breves y simples a la hora de construir un circuito lógico que las cumpla. Al disponer de sólo dos símbolos (0 y 1), las tablas son muy simples:

		+	×
0	0	0	0
0	1	1	0
1	0	1	0
1	1	10	1

La circuitería que “recuerda” estas tablas es simple; la circuitería que “recordara” las tablas para, por ejemplo base 10, tendría que ser forzosamente voluminosa y cara de construir.
- Existen un gran número de dispositivos biestables (con 2 estados) que se pueden emplear para almacenar información codificada en binario. Se trata de disponer de una variable física en la que podamos distinguir dos valores de referencia suficientemente diferenciados como para evitar estados ambiguos. Por ejemplo:
-- corriente eléctrica (voltaje): distinguir entre 10 o más niveles de voltaje es delicado y caro; distinguir entre pasa/no pasa corriente es muy económico y concede un amplio margen de tolerancia.
-- intensidad de luz: luz apagada/luz encendida.
-- perforación en papel o cartulina
-- sentido de magnetización: distinguir entre distintos valores de campo magnético es complicado; distinguir entre magnetización Norte-Sur y su contraria Sur-Norte es mucho más fácil y fiable.

3.1.2 Representación alfanumérica

Debido a que el ordenador no sólo maneja datos de índole numérica, sino además de tipo carácter, y en mayor medida, según avanzan las generaciones de PC´s, se hace necesario diseñar un código binario que represente las letras del alfabeto y otros caracteres necesarios, con la finalidad de poder representar nombres de ficheros, caracteres especiales, signos monetarios, mayúsculas, flechas de dirección, etc.

Por fuerza se debe de tratar de un código binario dado que el ordenador no tiene otra manera de trabajar dada su arquitectura actual.

3.1.2.1 Código ASCII

Su nombre resulta el acrónimo de *American Standard Code for Information Interchange*, es decir, se trata de un código estándar para el intercambio de información y fue diseñado en 1963.

En total está formado por 256 caracteres, que se suelen presentar en formato tabla y se publican en la mayoría de los libros de informática, sobre todo los de edición o los dedicados a diseño gráfico.

La mayoría de los caracteres que figuran en la tabla ASCII son imprimibles por el ordenador, pero pueden presentarse excepciones debido a que en algunos equipos, algunos códigos de ASCII representan procedimientos de función. En general se puede afirmar que siempre los 128 primeros caracteres serán imprimibles. Entre estos cabe destacar:

- Los 26 códigos que representan las letras en mayúscula, de la A a la Z
- Los 26 siguientes códigos, que representan las letras en minúscula, de la a a la z
- Los siguientes 32 códigos que representan caracteres especiales de los que figuran en el teclado

Cada letra y cada carácter tiene asignado un código que varía de entre 1 a 3 cifras. por ejemplo, la letra H tiene asignado el código 72. Dado que el ordenador trabaja con bytes, es decir, código binario agrupado en 8 dígitos, es decir, en octetos, su codificación binaria será:

72 entre 2 = cociente 36 y resto 0
36 entre 2 = cociente 18 y resto 0
18 entre 2 = cociente 9 y resto 0
9 entre 2 = cociente 4 y resto 1
4 entre 2 = cociente 2 y resto 0
2 entre 2 = cociente 1 y resto 0

por lo que el código en byte para la letra H será

0 1 0 0 1 0 0 0

Hay que tener en cuenta que cuando se procede a las divisiones sucesivas entre 2 para averiguar los 0 y 1 que forman el byte, podría ocurrir que se acabara la división antes de reunir los 8 dígitos, por lo que entonces se rellena por la izquierda con 0 hasta llegar al octeto.

3.1.2.2 Código EBCDIC

Acrónimo de *Extended Binary Coded Decimal Interchange Code*. Se trata de un esquema de codificación para caracteres basado también en grupos de 8 bits. Este sistema se empezó a usar hace bastantes años en las primeras máquinas de IBM y se generalizó para el caso de los grandes sistemas y medios sistemas de IBM.

El sistema de codificación maneja los mismos caracteres gráficos, pero asigna grupos de 8 bits diferentes que en el ASCII, por lo que la codificación resultará diferente.

4. MEDIDA DE LA INFORMACIÓN

Se trata de un sistema diseñado para estimar la cantidad de información manejada por un sistema informático, por un programa o por un determinado procedimiento. Para ello se definen los siguientes términos:

- **Bit**: Un elemento biestable (con dos posibles estados) en el que se diferencian dos valores, es decir, es una variable binaria. A efectos de representación, escribiremos los dos posibles valores de la variable binaria como 0 y 1. De esta forma, el valor tomado en un instante dado por la variable binaria vendrá dado por un dígito binario que valdrá 0 o 1.
Como el término dígito binario es algo largo de escribir, se abrevia con la palabra bit, que proviene de la contracción de la expresión inglesa *BInary digiT*, que además, en inglés, significa "trocito".
Podemos codificar cualquier alfabeto de partida en binario, o sea mediante bits. Cuantos más símbolos contenga el alfabeto más número de bits nos harán falta para codificarlo, pero, en definitiva, no hay información que no podamos codificar en binario. La prueba es que hoy en día tanto la información visual como la auditiva de alta fidelidad se codifica en binario.
- **Byte**: Como el bit es una unidad de medida muy pequeña, es costumbre usar unidades de magnitud superior. Un bytes es un conjunto de 8 bits. Así, en lugar de decir que un mensaje tiene 32 bits, podemos decir que tiene 4 bytes.
Un byte puede almacenar 8 dígitos binarios, es decir, dos dígitos hexadecimales. El número de valores posibles que se pueden almacenar es de 28=256.
Los bits de un byte se numeran de derecha a izquierda del 0 al 7, es decir, se corresponden con los exponentes de las potencias de base 2. La agrupación de los 4 bits (superiores o inferiores) de un byte se llama nibble. Por tanto, un byte contiene 2 nibbles. El que corresponde a los bits 0 a 3 se llama nibble inferior, y el de los bits 4 a 7 se llama nibble superior.
El nibble es una unidad de trabajo mucho más cómoda que el bit, ya que en cada nibble se almacena un dígito hexadecimal.
- **Carácter**: Es la unidad de información a nivel de lenguaje humano. Un carácter es, de hecho, cualquier símbolo del alfabeto. Constituye una buena medida de información en términos directamente aplicables a textos. Podemos clasificar los caracteres en:
-- alfabéticos: letras y algún que otro carácter asimilado
-- numéricos: los dígitos numéricos del 0 al 9
-- especiales: todos los restantes (letras de alfabetos extranjeros, letras griegas, signos de puntuación, signos monetarios, signos de operaciones aritméticas, etc).
- **Múltiplos**: Normalmente, en un ordenador, para representar un carácter se usa un tamaño de 1 byte. Cuando las cantidades de información a medir son grandes, se utilizan múltiplos de las unidades mencionadas.
La K es un factor de multiplicacion de 2^{10}=1.024. Así que 1 Kbit=1.024 bits y 1 Kbyte=1.024 bytes=8.192 bits. Se toma el valor de 1.024 en vez de 1.000 precisamente por ser 1.024 una potencia de 2, y en consecuencia, un valor mucho más conveniente para máquinas que trabajan en sistema binario.
La M es la abreviatura de Mega y representa el factor de multiplicación 2^{20}= 1.048.576.
La G es abreviatura de Giga y repesenta el factor de multiplicación 2^{30}= 1.073.741.824.

Utilizando el byte como unidad de medida, el esquema o la tabla de medición para la capacidad de los distintos dispositivos de almacenamiento o de memoria, RAM, discos duros, etc., quedará:

Tipo de unidad	Capacidad (bits)
1 byte	8
1 Kilobyte (Kb)	1.024
1 Megabyte (Mb)	1.024 Kb = 1.048.576
1 Gigabyte (Gb)	1.024 Mb = 1.073.741.824
1 Terabyte (Tb)	1.024 Gb =