

1. Representación de la Información.

La información que maneja el ser humano se representa por una serie de números y letras con los que se forman cantidades y palabras. Para las cantidades se emplea el sistema numérico decimal, que usa los dígitos del 0 al 9, mientras que para las palabras utilizamos el alfabeto del idioma que empleemos.

El ordenador no puede utilizar estos sistemas directamente ya que, como toda máquina electrónica, sólo "comprende" las señales eléctricas. En realidad, sólo utiliza señales eléctricas (0 y 5 v. o, también 0 y -3,3 v ...). Si por un cable "llega" carga eléctrica se entiende como 1 y si no como 0. A este lenguaje que emplea como símbolos únicamente el 0 y el 1 (dos símbolos) se le conoce como *lenguaje binario* o simplemente binario. Pues bien, todo, absolutamente todo lo que maneja un ordenador tiene que estar codificado en binario, ya sea texto, número, imagen, vídeo, sonido o cualquier otro tipo de información.

a) Representación de palabras: el código ASCII.

Cuando leemos un texto, independientemente del idioma en que esté¹, observamos que está compuesto por una serie de símbolos: letras mayúsculas y minúsculas, algunas de ellas con tilde que puede ser a su vez de distintos tipos (´ ` ¨ ^), signos de puntuación (. ; , :), y otros muchos símbolos: ! " \$ % & / () = ? ¿ ^ ` [] { } - _ < > \ / + * | @ # ~ ° ª ® º § ¥ £ µ € » ~ etc. En total más de 200 símbolos. ¿Cómo se pueden representar todos estos símbolos en el ordenador si, como hemos visto, sólo emplea ceros y unos?. La solución a este problema es la codificación.

Se denomina *bit* a la cantidad mínima de información que puede representar, almacenar o transmitir un ordenador. Es decir un 0 o un 1. Con un bit, sólo podemos representar dos² estados (dos códigos): 0 y 1. Si empleamos dos bits, podremos representar cuatro³ estados: 00, 01, 10 y 11. Con tres bits ocho⁴ estados: 000, 001, 010, 011, 100, 101, 110 y 111.

Necesitamos tener más de 200 códigos, ya que son más de 200 símbolos. Por tanto el número de bits necesario es 8, puesto que $2^7 = 128$ y $2^8 = 256$. Luego, para representar cualquier carácter se necesitan 8 bits, cantidad a la que se denomina *byte*.

¹ Exeptuando idiomas que emplean otro tipo de alfabeto: árabe, chino, ruso, griego...

² $2^1 = 2$

³ $2^2 = 4$

⁴ $2^3 = 8$

Sólo faltan dos cosas: asignar a cada símbolo un código binario de 8 bits (por ejemplo el 00101001) y que esa asignación sea universal, es decir, igual para todos los ordenadores, pues de otra forma no podrían compartir información. De esto se encargó un comité americano (ANSI), que creó un código estandar, el código ASCII (American Standar Code for Information Interchange o Código Estándar Americano para el Intercambio de Información). La tabla siguiente muestra algunos de los 256 códigos que componen el ASCII.

A	01000001	B	01000010	C	01000011	D	01000100	E	01000101
F	01000110	G	01000111	H	01001000	I	01001001	Z	01011010
a	01100001	b	01100010	c	01100011	d	01100100	e	01100101
f	01100110	g	01100111	h	01101000	i	01101001	j	01101010
k	01101011	l	01101100	m	01101101	n	01101110	ñ	11110000
o	01101111	p	01110000	q	01110001	r	01110010	s	01110011
,	00101100	:	00111010	;	00111011	(00101000)	00101001

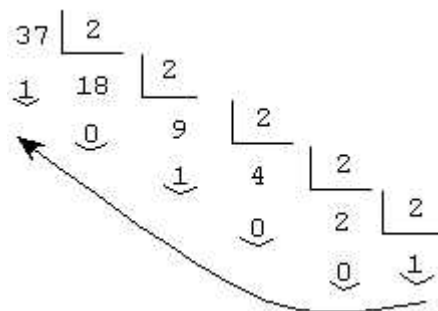
Cuando escribimos utilizando el teclado, cada letra se convierte a su correspondiente código binario. Así la palabra *Almenara* sería traducida a 01000001 01101100 01101101 01100101 01101110 01100001 01110010 01100001.

Cuando la letra se muestra en pantalla o se imprime, se hace la operación inversa pasando del código binario a la letra o símbolo correspondiente.

b) Representación de cantidades.

El sistema de numeración utilizado por el ser humano para representar cantidades es el sistema decimal o base 10, que emplea los dígitos del 0 al 9 y un conjunto de reglas para representar las cantidades. Básicamente, cada símbolo tiene un valor, pero este se multiplica por 1, 10, 100... (10^0 10^1 10^2 ...) en función de estar situado en la posición de las unidades, decenas, centenas etc. Lo mismo ocurre en el sistema binario, sólo que los valores de las posiciones por los que hay que multiplicar son 1, 2, 4, 8 etc. Es decir 2^0 2^1 2^2 2^3 ... Así, el número binario $100101_{(2)}$ equivale al $37_{(10)}$, pues $1 \cdot 2^5 + 0 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 32 + 4 + 1 = 37_{(10)}$

Para la operación inversa (pasar de decimal a binario), hay que dividir por la base (2) y coger los restos en orden inverso, como muestra la figura :



Por lo que, como era de esperar, $37_{(10)} = 100101_{(2)}$

c) Representación de imágenes.

Existen muchos formatos de imagen distintos (bmp, jpg, gif, pcx, tif...) y cada uno *codifica* la imagen de una forma diferente, obteniendo también distintas calidades de imagen y tamaño de archivo⁵. Pero todas las imágenes no son más que una sucesión de infinidad de puntos de colores o pixels. Cuando decimos que una imagen ocupa 120 x 60, significa que se compone de una matriz de 60 filas y 120 columnas de puntos, por lo que tendría en total $60 \times 120 = 7200$ pixels. La calidad de la imagen viene determinada por la resolución (número de pixels) y por el número de colores o tonos que emplee. Cuantos más puntos por pulgada (dpi o dot per inch) y más colores, más calidad. Aunque también es cierto que llegado un punto, el ojo humano es incapaz de distinguir entre tonalidades muy similares de un mismo color.

El formato bmp o de mapa de bits codifica cada color posible, asignándole un código binario, y guarda para cada punto de la imagen el código binario correspondiente a su color. Esto quiere decir que, si utilizamos 256 colores, necesitaremos 8 bits (1 byte) para guardar el color de cada punto o pixel de la imagen, pues $2^8 = 256$. Si utilizamos 65.536 colores, necesitaremos 16 bits (2 bytes) por punto, pues $2^{16} = 65.536$. Si son 16 millones de colores, 24 bits (3 bytes), ya que $2^{24} = 16.777.216$.

Ejemplo:

Una imagen bmp de 800x600 y 16 millones de colores posibles (aunque no todos se utilicen), ocuparía en memoria $800 \times 600 \times 3 \text{ bytes} = 1.440.000 \text{ bytes}^6$ (o lo que es lo mismo 1406 Kb o 1,37 Mb). Es decir, si guardamos la imagen en un disquete, ocuparía todo el disco.

Escala de magnitudes

Magnitud	Símbolo	Equivale a		
1 byte	B	8 bits		
1 Kilobyte	Kb	1.024 bytes		
1 Megabyte	Mb	1.024 Kb	1.048.576 B	
1 Gigabyte	Gb	1.024 Mb	1.048.576 Kb	1.073.741.824 B
1 Terabyte	Tb	1.024 Gb	1.048.576 Mb	1.073.741.824 Kb o 1.099.511.627.776 B

Pero bmp, creado por Microsoft, es un formato técnicamente muy malo. Por ello se utilizan otros formatos como jpg para fotografías (muchos colores) o

⁵ El simple echo de que la misma imagen ocupe más espacio en memoria en un formato que otro, no implica necesariamente que la calidad sea mayor.

⁶ En realidad 1.440.054 porque se utilizan 54 bytes adicionales para guardar información referente a la imagen (número de puntos y número máximo de colores utilizados) e información relativa al archivo (fecha de creación, fecha de último acceso, nombre, etc...)

gif para gráficos (pocos colores) puesto que son más eficientes. Es decir, manteniendo la misma calidad de imagen, ocupan mucho menos espacio en memoria.

Ejemplo:

La misma imagen del ejemplo anterior, pero guardada como formato jpg ocuparía 57,66 Kb, es decir 24 veces menos. O, visto de otra forma, en un disquete no podríamos guardar una imagen, sino 24. En formato gif, un gráfico, del mismo tamaño, podría ocupar menos de 4 Kb. 350 veces menos.

d) Sonido, Vídeo...

Ni que decir tiene que el sonido y, sobre todo el vídeo, ocupan mucho más aún que las imágenes. Este es el motivo por el que la videoconferencia, para que tenga una calidad aceptable, necesita de cableado. A ser posible fibra óptica. Los medios de transmisión lentos como Internet o, incluso transmisión vía satélite, no son adecuados para este tipo de servicios.

2. Software

Como sabemos ya, en los sistemas informáticos existen dos componentes claramente diferenciables, el hardware y el software. El primero hace referencia a los elementos físicos: carcasa, circuitos electrónicos interiores, pantalla, teclado, ratón, disco duro, CD-ROM, etc. El software está formado por los datos y por el conjunto de programas que nos permiten controlar el funcionamiento del ordenador. Un **programa** es por tanto, un *conjunto de órdenes que indican al ordenador qué acciones hay que realizar sobre los datos para obtener los resultados que desea el usuario.*

Ambas partes son imprescindibles para que el ordenador funcione. De nada sirve un equipo sin el software para manejarlo, igual que no sirve de nada un buen programa sin un hardware donde ejecutarlo.

Dependiendo de los objetivos para los que haya sido creado, podemos clasificar el software en tres grupos: software de sistemas, de programación y de aplicación.

a) El **software de sistemas**, está formado por los programas que se encargan de controlar, coordinar y gestionar todo el hardware del ordenador. Algunos de estos programas reciben el nombre de **sistemas operativos**, y actúan como intermediarios entre los componentes físicos del ordenador y el usuario. Entre los principales sistemas operativos se pueden destacar: **Ms-dos**, **Dr-dos**,

Os/2, Windows (3.0, 3.1, 95, 98, 2000, NT, ME, XP), **Unix** (Aix, BSD, SVR5, SCO...), **Linux** (Red Hat 8, Suse, Mandrake 9.1, LinEx...). Otro tipo de software de sistemas son los drivers, programas que utiliza el s.o. para comunicarse con cada dispositivo y que crea el propio fabricante de hardware.

b) El **software de programación** reúne los programas que utilizan los programadores para crear nuevos programas. Estos últimos se crean utilizando un **lenguaje de programación**, o conjunto de palabras clave o instrucciones junto con unas reglas sintácticas que nos ayudan a diseñar un programa y "explicarle" al ordenador qué tiene que hacer o, mejor dicho, cómo hacerlo.

Existen miles de lenguajes de programación, cada uno tiene sus propias instrucciones y reglas sintácticas. Básicamente pueden clasificarse en: lenguajes de bajo nivel, lenguajes de alto nivel, lenguajes 4G o de cuarta generación y lenguajes orientados a objetos.

Los **lenguajes de bajo nivel** reciben este nombre porque están muy *cercanos* al hardware. Los ordenadores son máquinas electrónicas y, como hemos visto, el único lenguaje que *conocen* es el basado en la transmisión de señales eléctricas por un cable: el lenguaje *binario* (0 si la señal es de 0 V. Y 1 si es de 5 V.) Por ello, el primer lenguaje de programación que se utilizó fue el **lenguaje máquina**, i.e., un conjunto de instrucciones en binario. Pero, aunque los humanos podamos memorizar una determinada cantidad de códigos en binario, no es fácil trabajar con ellos. El lenguaje Ensamblador, un poco más avanzado, consiste en asignar a cada instrucción en binario un nombre, compuesto normalmente por tres letras (ADD para la suma, CMP para comparar, JNE para saltos condicionales...). Pero sigue siendo necesario conocer a fondo la arquitectura de la máquina para la que se va a trabajar.

move	dx, es	
cmp	es, ax	
jne	@@FIN	Pequeño extracto del código de
xor	ax, ax	un programa realizado en
add	bx, cl	lenguaje ensamblador.
move	ah, 30h	
int	21 h	

Con los **lenguajes de alto nivel** ocurre lo contrario, no es necesario conocer tan a fondo el funcionamiento del ordenador para realizar un programa en uno de estos lenguajes. Las instrucciones suelen ser palabras que se utilizan para hablar normalmente, aunque eso sí, en inglés. Existen infinidad de lenguajes de programación de alto nivel, con diversos niveles de abstracción y diseñados para finalidades muy distintas. Algunos de los más conocidos son: Basic, Cobol, C, Visual Basic, Pascal. HTML, el "*lenguaje de Internet*", no es un lenguaje de

programación como mucha gente piensa, sino más bien de *descripción de documentos*.

Veamos un ejemplo de un programa, algo simple, en Pascal:

```
Program prueba;
Uses WinCrt, WinTypes, WinProcs;

      (* esto es un comentario, el
      compilador lo va a ignorar *)

Var      (* declaración de datos *)
  i: integer;
  n, color: longint;
  x,y: integer;

begin      (* inicio del programa *)
  repeat
    x := random(80);
    y := random(40);
    CursorTo (x,y);
    Write('I.E.S. Almenara. Vélez-Málaga (MÁLAGA)');
    For n := 0 to 8000000 do n := n+0;
    ClrScr;
  until KeyPressed;
end.
```

Este programa, realizado para Windows, abre una ventana y va escribiendo la frase *I.E. .S. Almenara. Vélez-Málaga (MÁLAGA)* en distintos (y aleatorios) puntos de la pantalla. La ejecución del programa finalizará cuando pulsemos cualquier tecla o cerremos la ventana.

Un programa como este en lenguaje Pascal puede ser *relativamente fácil* de entender para nosotros, pero no para el ordenador, por lo que habrá que traducirlo de alguna forma a lenguaje máquina (binario). Para realizar esta tarea se emplean los intérpretes y los compiladores, también programas, que previamente habrán creado otros programadores.

El **intérprete** toma el programa creado con el lenguaje de alto nivel (*programa fuente*) y lo va traduciendo y ejecutando instrucción a instrucción. Una ventaja del intérprete es, que si el programa tiene errores, permite al programador corregirlos sobre la marcha y continuar la ejecución. El principal inconveniente es que cada vez que se desea ejecutar el programa es necesario volver a traducirlo.

Un **compilador**, primero traduce todas las instrucciones del programa fuente y crea un programa traducido a lenguaje máquina llamado *programa objeto*. La ventaja es que el programa objeto podrá ser ejecutado todas las veces que quiera el usuario sin tener que realizar más traducciones.

Los **Lenguajes de 4ª Generación** nacieron para solucionar problemas muy concretos. Hasta entonces los lenguajes de programación eran de propósito general, pero no se adaptaban bien a determinados campos más específicos como la programación de bases de datos, sistemas expertos etc. Un ejemplo es SQL (Structured Query Language) para gestionar y consultar bases de datos.

Por último, los **Lenguajes orientados a objetos**, suponen un nivel más de abstracción. El programa sigue constando de instrucciones, pero el diseño del mismo se basa en objetos que interactúan entre sí, comunicándose a través de métodos. Cada objeto puede servir de base para crear objetos más complejos mediante el mecanismo de herencia. La programación orientada a objetos (POO) es, conceptualmente, mucho más complicada, pero también más potente y eficaz cuando se quieren realizar programas de cierta complejidad.

Java es sin duda el mejor lenguaje orientado a objetos por su estabilidad, seguridad, y sobre todo portabilidad (un programa creado en Java funciona en cualquier sistema operativo, mientras que otros lenguajes y . . . Otros lenguajes de este tipo son: C++, JavaScript, VBScript.

c) Por **software de aplicación** se conoce al conjunto de programas que utilizan los usuarios para trabajar con el ordenador. Estos están creados utilizando lenguajes de programación y se ejecutan sobre un determinado sistema operativo. Por ejemplo: cualquier procesador de textos, hoja de cálculo, juego, navegador para Internet o, también el programa del ejemplo anterior una vez traducido mediante un compilador Pascal a código máquina o binario.

3. Licencias de software y normativa legal.


La primera referencia que existen en nuestro país sobre la propiedad de programas de ordenador es la Ley de Propiedad Intelectual de 1983.

Posteriormente en mayo de 1991 aparece una Directiva de la UE, y por tanto de aplicación en España, sobre la Protección Jurídica de Programas de Ordenador. El principio básico de esta directiva-ley consiste en tratar los programas como obras literarias, considerándose como creación intelectual de su autor, siempre y cuando el programa sea original. No se protege la idea principal del programa (ej, una hoja de cálculo), sino el diseño y la forma en que se ha desarrollado.

Como cualquier obra literaria u obra de arte, los derechos que el autor tiene sobre el programa incluyen la explotación, reproducción, transformación,

distribución y comunicación pública del mismo, durante 50 años. Transcurrido este periodo de tiempo, el programa pasa a ser de dominio público. Naturalmente una obra literaria conserva su valor o lo incrementa con el paso del tiempo, pero un programa queda obsoleto rápidamente.

Todos los programas de ordenador comerciales tienen licencia de uso. Dependiendo de esta licencia, se clasifican en:

- ☞ **Freeware.** Son programas gratuitos. Se pueden copiar y/o distribuir libremente, pero no se puede comerciar con ellos.
- ☞ **Shareware.** No es un tipo de programa sino una forma de distribución. Permite a los usuarios probar el programa, incluso distribuirlo libremente, pero tienen una restricción que puede ser temporal (pasado un tiempo, deja de funcionar) o funcional (por ejemplo no se puede imprimir o guardar). El objetivo de la empresa es que el usuario pruebe el programa y si le conviene, lo compre.
- ☞ **Software libre** u OpenSource. El programa no sólo es gratuito sino que su código fuente está disponible para que cualquier persona o empresa pueda estudiarlo y ampliarlo o mejorarlo. El movimiento openSource nació en 1991 cuando Linus Torvalds, un joven ingeniero finlandés de la Universidad de Helsinki, después de un año de investigación y desarrollo, creó el núcleo (Kernel) de un sistema operativo al que llamó Linux. Después de haberlo creado, colocó el código fuente en Internet y, sorprendentemente personas de todo el mundo, estudiando primero el código original, siguieron mejorando y añadiendo cosas a Linux. Este hecho causó toda una revolución tecnológica en el mundo del desarrollo de software, hasta el punto de suponer la única amenaza real para la hegemonía de Microsoft y su sistema operativo Windows. 
- ☞ **Software propietario.** Requieren licencia de uso que se obtienen al comprar el programa. A veces las condiciones de uso del programa se recogen en un contrato, que especifica el número de copias que se puede hacer, condiciones de mantenimiento etc. Si no se indica un número de copias, el comprador está autorizado a realizar una única copia de los discos originales para utilizarla en caso de deterioro o pérdida de estos.

Piratería informática.

La Ley de Protección Jurídica de Programas de Ordenador, establece tres tipos de infracciones:

- Poner en circulación una o más copias de un programa de ordenador conociendo su naturaleza ilegítima.

- Poseer con fines comerciales y económicos una o más copias de un programa de ordenador careciendo de las correspondientes licencias.
- Poner en circulación o poseer con fines comerciales y económicos cualquier medio, programa o dispositivo cuyo único uso sea facilitar la eliminación o neutralización de cualquier dispositivo técnico utilizado para la protección de un programa de ordenador.

Por otro lado, muchas empresas y gobiernos están apostando por el software libre y no sólo por el importante ahorro del dinero de las licencias. Como ejemplos: el gobierno de Brasil, el ayuntamiento de Munich en Alemania o la Junta de Extremadura y, más tímidamente, en Andalucía.

Bill Gates, presidente de Microsoft:

«Gracias a la Ley de Moore los procesadores son hoy un millón de veces más rápidos que cuando abandoné los estudios. Dentro de un par de décadas, los procesadores volverán a ser otra vez un millón de veces más veloces. Hay una gran diferencia entre lo que podían hacer los ordenadores de hace diez años y lo que son capaces ahora. Esto es lo que hace de la informática un campo de actividad apasionante».

« Casi asusta pensar lo deprisa que va esto y el impacto que puede llegar a producir, pero es algo grande asistir en primera fila mientras las cosas ocurren. ».

Andy Grove, presidente de Intel:

«Resulta difícil predecir a dónde nos puede llevar la informática durante los próximos años, porque se trata de una industria con apenas diez o quince años de historia real y eso es muy poco tiempo como referencia. Lo que sí está claro es que nos encontramos en un momento en que la informática sigue llenándolo todo. Quiero decir que se hace presente en aspectos de nuestras vidas donde antes no lo estaba. Y en áreas del mundo en que antes no existía. Y creo que esta penetración continuará produciéndose».