Introducción a los Lenguajes de Marcas

Uno de los componentes básicos en un sistema informático son los **datos:**

- cuales se pueden introducir en él
- cómo los almacena
- cómo los muestra una vez almacenados.

Se podría decir, que **en un sistema informático** cualquiera, las únicas **tareas que se desarrollan**, consisten en **almacenar datos para procesarlos por medio de un programa** que:

- o bien aportará algún tipo de información.
- o bien los utilizará para generar otros datos.

Uno de los componentes básicos en un sistema informático son los **datos:**

- cuales se pueden introducir en él
- cómo los almacena
- cómo los muestra una vez almacenados.

Entre las **características** interesantes sobre los **datos**

- A quién van dirigidos (información de representación)
  - ✓ A los humanos: estructura concreta, con unos formatos determinados, con textos decorados de alguna manera.
  - ✓ A los programas: fácilmente identificables, de qué tipo son y qué significan para poderlos tratar automáticamente.

- La posibilidad de reutilizarlos :
  - ✓ las personas
  - ✓ los programas.

- Que se puedan compartir:
  - ✓ en sistemas operativos totalmente diferentes
  - ✓ en máquinas funcionamientos muy diversas (arquitecturas)

Los ordenadores (dada su arquitectura) utilizan el binario para el **almacenamiento de datos.**

- Para almacenar cualquier tipo de dato (imágenes, vídeos, texto . . .), debe haber algún proceso que convierta los datos a una representación en formato binario.
- Tradicionalmente, en los ordenadores los datos se organizan de dos maneras:
  - Datos de texto
  - Datos binarios

Almacenar los datos de forma binaria es la forma natural de almacenar datos en ordenadores .

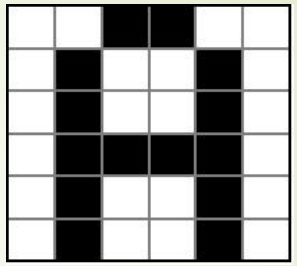
- Generalmente están optimizados para ocupar el **espacio** necesario
- Los ordenadores los **leen fácilmente**.
- Pueden tener **estructura**.
- Es relativamente fácil añadir **metadatos**.

Para almacenar el número 150 hace falta convertir este valor decimal a su representación en binario y almacenarlo. Es trivial comprobar que puede ser almacenado en un solo byte (8 bits)

valor decimal	valor binario
150	10010110

Los **metadatos** son datos sobre los datos. Los datos no se almacenan directamente tal como están, sino que se procesan para optimizarlos:

- Se almacena información sobre su contenido
- Se aplican procedimientos de optimización



Para representar esta imagen en un ordenador se podría almacenar el color de cada uno de sus cuadros. Para almacenarla en un archivo almacenamos el color de cada cuadro

Podemos definir el color blanco (0) y el negro (1)

001100
010010
010010
011110
010010
010010

00110001001001001001111001001001001O

Para optimizar el espacio observamos las repeticiones de los colores. Un punto blanco (0), pero cuatro puntos blancos (40) en vez de 0000.

2021301201201201204120120120120l0

ocupan un **10%** menos de espacio

En la representación binaria hay valores que no son datos de la imagen (2, 3 y 4); son datos que hacen referencia a la forma en que se han almacenado los datos. Estos datos se denominan " **metadatos** ".

Para tratar los datos deben tener algún tipo de **estructura**

- Los datos, en la forma en que los generamos los humanos, no están en un formato que facilite el tratamiento automático por parte de un ordenador.
- Se convierten en formatos que faciliten su tratamiento..
- Los tipos de **datos estructurados** son agrupaciones de otros tipos de datos (normalmente tipo más sencillos).
- Los datos binarios se suelen estructurar agrupándolos en registros con información repetitiva de un dato en concreto.

Es habitual que los lenguajes de programación definir datos estructurados. Por ejemplo, para estas tareas, el lenguaje C utiliza struct.

```
1  struct alumno {
2          char nombre [10];
3          char apellido [10];
4          int nota;
5  }
```

Se puede identificar a qué dato corresponde cada uno de los caracteres. Los diez primeros son el nombre, los 10 siguientes son el apellido y los cuatro siguientes son el número entero (32 bits).

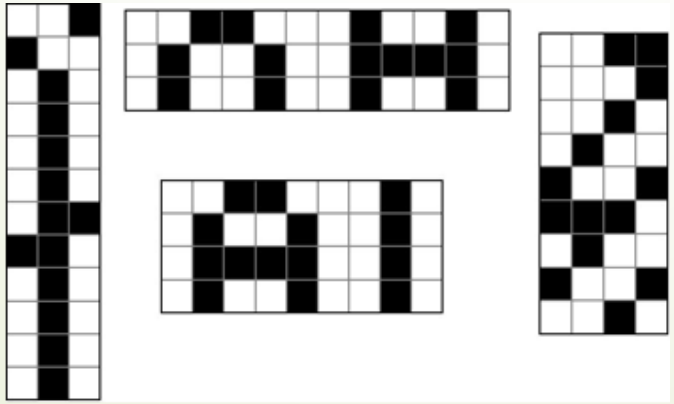
Los **datos binarios** tienen problemas a la hora de ser **compartidos**:

- Se deben compartir en máquinas en las que no han sido generados, con **SO diferentes**, con **distintas arquitecturas** , etc.
- La o**ptimización** de SISTEMA en que se generan no siempre será bien entendidos por los otros SISTEMAS
- Al dar **estructura a los datos**, sólo la entenderán los programas que tengan información sobre la estructura.

Al definir los datos que utiliza un programa, se define qué tamaño tendrá cada campo y de qué manera se guardarán los datos dentro del archivo binario.

En el ejemplo anterior, para que un programa pueda representar la imagen de manera correcta, es necesario tener información sobre:

- Es una imagen.
- Se va a guardar cada punto de color con un solo carácter.
- La equivalencia de colores : 0 es blanco, 1 es negro.
- La imagen es de 6 caracteres de largo por 6 caracteres de ancho.
- Si se ha optimizado, los valores numéricos superiores a 1 indican que este valor no es un color, es el número de repeticiones del color siguiente.



Los datos en **formato binario**:

- Son ideales para ser almacenadas en máquinas;
- Se debe utilizar un programa para que un humano los pueda ver su representación

Se han definido algunos **estándares de ficheros binarios**, permitiendo que cualquiera pueda hacer programas que puedan interpretar estos archivos.

- JPG - Estándar ISO/IEC 10918
- GIF - Especificación del W3C GIF89a
- PNG - Estándar ISO/IEC 15948

Los archivos de imágenes JPG, PNG, GIF…, pueden ser leídos por diferentes programas ya que su especificación es pública.

Los **archivos de texto almacenan la información letra por letra**, de una manera similar a como lo haría un humano al escribir.

Los humanos al escribir ya están utilizando una codificación. Si utilizamos esta codificación, tendremos los datos en un formato

- Fácil de usar y entender
- No tendrá problemas para ser leído por los programas.

En archivos binarios el componente de información más pequeño es el **bit**, en ficheros de texto el componente más pequeño es el **carácter**

Representar los datos en un ordenador en forma de **texto**, implica

- Cada palabra deberá ser codificada en binario
- Determinar una cantidad de bits predefinida para marcar un carácter
- Asociar un valor numérico a cada uno de los caracteres

La equivalencia entre los **caracteres y sus valores numéricos**, no se puede hacer de manera aleatoria.

Si se quiere conseguir que los datos se puedan leer en diferentes sistemas, hay que seguir algún tipo de norma conocida por todos. Para esto aparecieron los **estándares de codificación de caracteres**.

Uno de los **estándares** adoptado mayoritariamente es **ASCII** (**A**merican **S**tandard **C**ode for **I**nformation **I**nterchange).

- ASCII codifica cada uno de los caracteres con siete bits definiendo un valor numérico para cada carácter.

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

carácter	decimal	binario
H	72	1001000
o	111	1101111
l	108	1101100
a	97	1100001

Unicode,

- es un intento de sustituir todos los códigos de caracteres existentes en uno genérico, que sirva para todas las lenguas,
- pretende superar todos los problemas de incompatibilidad en entornos multilingües añadiendo los caracteres no latinos.
- da a cada uno de los símbolos un identificador único universal
- se pueden utilizar en el mismo documento idiomas diferentes
- evita los problemas de representación.

A pesar de sus ventajas, Unicode también recibió críticas de la comunidad anglófona, porque hacía que los textos acabaran ocupando más del doble que en ASCII

Unicode,

- Define tres formas de codificación básicas **UTF** (**U**nicode **T**ransformation **F**ormat)
  - ✓ U TF-8 Sistema basado en un byte con algunos símbolos de longitud variable
  - ✓ U TF-16 Sistema de longitud variable basada en dos bytes
  - ✓ U TF-32 Sistema de longitud fija que utiliza 32 bits para cada carácter

Actualmente casi todos los sistemas operativos utilizan alguna variedad de Unicode (el Linux suele utilizar UTF-8 y el Windows adapta UTF-16).

Almacenar los datos en **formato de texto** aporta grandes **ventajas**:

- Los programas pueden usarlos (editores de texto, navegadores, etc.).
- Pueden ser leídos por humanos.

- ✓ Un programa (editor de texto), puede crear un documento que se **podrá compartir con cualquier persona** que entienda el idioma en que ha sido escrito.
- ✓ Los SO incorporan programas capaces de **compartir archivos**, pudiendo enviar archivos que serán perfectamente interpretados cuando se reciban en el destino.

Almacenar los datos en **formato de texto** también tiene **inconvenientes**:

- Ocupan **más espacio** en disco que los binarios.
- Hay **múltiples códigos** de caracteres diferentes.
- La **forma de tratarlos** los diferentes SO.
- Falta de **estructuración** de los datos.

Aún así, son la forma más sencilla de **asegurarnos de que podemos compartir la información** que contienen con otras personas.

- Los **programas requieren** que los datos que son tratadas estén definidos con algún tipo de **estructura**.
- Existen sistemas para hacer que los datos de los **ficheros de texto** puedan ser **estructurados**.
- El formato **CSV** (**C**omma **S**eparated **V**alues) es uno de los formatos usado para **exportar datos estructurados** (contenidos en bases de datos u hojas de cálculo) a texto,
  - ✓ separa cada uno de los registros de la estructura en líneas
  - ✓ los campos se separan con comas
  - ✓ los tipos de datos de texto van entre comillas los datos de texto, el resto no

CSV es una manera sencilla de guardar datos estructurados en formato de texto que permite a un programa **identificar los diferentes datos que contiene cada registro** y además interpretar de qué tipo son.

```
"Manuel", "Perez", "Garcia", 8
"Pedro", "González", "Parada", 5
"María", "Pozos", "Calle", 7
```

un programa puede deducir fácilmente a partir del ejemplo anterior que los tipos de datos

dato	resultado
Manuel	Dato de texto porque está entre comillas
Perez	Dato de texto porque está entre comillas
Garcia	Dato de texto porque está entre comillas

Si necesitamos **añadir más datos en cada registro** es casi seguro que obligará a hacer **cambios en el programa** que los tratará

```
"Sr.", "Manuel", "Perez", "Garcia", 8
"Sr.", "Pedro", "González", "Parada", 5
"Sra", "María", "Pozos", "Calle", 7
```

Al utilizar el mismo programa que antes, éste puede pensar que los nombres de las personas son "Sr" y "Sra" y que las notas son "Garcia", "Perez", etc.