ESCUELA DE PREPARACIÓN DE OPOSITORES

E. P. O.

C/. La Merced, 8 – Bajo A Telf.: 968 24 85 54

30001 MURCIA

INF10-SAI10

Representación interna de los datos.

Esquema.

1 Introducción.

Las computadoras son máquinas para el tratamiento automático de la información. Esta información no se almacena ni se representa al azar, sino que debe organizarse o estructurarse en forma adecuada para obtener un rendimiento razonable en su memorización, tratamiento y recuperación.

Los datos se introducen inicialmente en el computador según un código de E/S, de forma que es necesario realizar una conversión de notaciones pasando de la representación simbólica de E/S a otra notación que denominaremos representación interna. La representación interna de los datos depende del computador o del lenguaje de programación utilizado y del uso que el programador desee hacer con los datos. Es decir, el usuario, según las reglas del lenguaje de programación que utilice, puede optar entre varias representaciones posibles.

En este tema comenzamos estudiando el concepto de dato desde un enfoque lógico. Describimos los tipos de datos más usuales en informática, analizando algunas estructuras de datos utilizadas en programación, en sistemas operativos, o en el diseño físico de computadoras.

A continuación, estudiamos los aspectos relacionados con la representación de la información en el interior de las computadoras. Consideramos cuatro tipos distintos de información: textos, sonidos, imágenes y datos numéricos.

2 Tipos de datos.

Se denomina dato a cualquier objeto manipulable por la computadora. Datos son tanto las constantes definidas dentro de los programas como las variables utilizadas en dichos programas. Asimismo, son datos la información externa al programa, a la que se puede acceder mediante algún procedimiento, ya esté dicha información grabada en algún medio de memoria masiva o sea generada por algún periférico.

Cuando utilizamos una computadora para resolver un problema debemos hacer una abstracción de la información y de las magnitudes que influyen en ella. Dichas magnitudes deben ser representadas por datos. La representación de una magnitud como un dato se puede entender como una aplicación que hace corresponder un dato a cada valor de la magnitud.

Esta transformación es deseable que esté definida sobre todo el conjunto de valores de la magnitud. Es conveniente, además, que sea unívoca, es decir, que a dos valores de magnitud distintos les asocie datos distintos. Para que se pueda operar con los datos es necesario que existan unas operaciones internas en el conjunto de datos, que sean semejantes a las operaciones usuales en el conjunto de magnitudes. Dichas operaciones deben cumplir que: la imagen según la transformación T del resultado de una operación en el conjunto de magnitudes, sea igual al resultado de las operaciones correspondientes en el conjunto de datos sobre las imágenes de los operandos.

Si denotamos por + la suma en el conjunto de magnitudes, d, lo dicho anteriormente implica que debe existir una operación $\oplus$ sobre el conjunto de datos, D, que cumpla:

$$T(x+y) = T(x) \oplus T(y) = X \oplus Y \qquad \forall x,y \in d$$

Para que los resultados obtenidos en el conjunto de datos, D, puedan ser interpretados, es necesario que exista una transformación, T^{-1}, de éstos al conjunto de

magnitudes, d. Esta aplicación T^{-1} hará corresponder a cualquier dato, X, un valor de magnitud, x, cuya imagen por T es el propio dato X.

Se denomina tipo de dato al conjunto de la transformación, T, y de las operaciones y funciones internas y externas definidas sobre el conjunto de datos. Distintas transformaciones darán lugar a distintos tipos de datos, aun cuando el conjunto origen a representar sea el mismo.

Nos encontraremos tipos de datos en la representación de la información, tanto a nivel físico del computador, como en los lenguajes de programación. No todos los tipos de datos existen en todos los lenguajes de programación. Hay lenguajes más ricos que otros en tipos de datos.

2.1 Datos de tipo entero

El tipo entero es una representación del conjunto de los números enteros. La representación es posible para un subrango de magnitudes enteras centrado en el origen: números entre 2^{n-1}-1 y -2^{n-1}. La razón de esta limitación está en la necesidad de utilizar un espacio finito y fijo para cada dato, y en el hecho de que la transformación realizada entre los números enteros y el tipo de datos consiste en representar el número en binario y almacenarlo con un número fijo de bits.

El número de datos distintos de tipo entero que se pueden generar, cardinalidad del tipo, es 2^n (donde n es el número de bits que se utiliza en la representación). Por tanto, si se modifica el número de bits n se obtienen distintos tipos enteros. En consecuencia, no todos los números enteros se puedan representar.

A cada operación aritmética sobre el conjunto de los números enteros se le asocia, en el tipo entero, la operación correspondiente. Estas operaciones no pueden realizarse sobre cualquier par de datos enteros, ya que aparecen indeterminaciones debidas a la propia representación. Así, si el máximo valor positivo de tipo entero es 32767 (como ocurre si la representación se realiza con n=16), al realizar la operación: 32767+1 se obtendría un resultado no representable en el tipo. Este tipo de error se conoce como desbordamiento (*overflow*). Cuando esto sucede la computadora puede, o no, avisar al usuario. En cualquier caso, se obtendrá como resultado algún dato de tipo entero, pero éste no corresponderá con el valor que se obtendría al realizar la misma operación con números enteros. Cualquier operación con datos de tipo entero es exacta salvo que se produzcan desbordamientos.

2.2 Datos de tipo real.

El tipo de datos real es una representación del conjunto de los números reales. Esencialmente, la transformación realizada consiste en expresar el número en la forma $N=m\cdot B^e$ donde N es el número real a representar, B es la base utilizada para el exponente, e es el exponente del número y m es la mantisa. El número se almacena en la computadora yuxtaponiendo el signo, el exponente y la mantisa.

Esta representación no permite el almacenamiento de números muy grandes o muy pequeños, lo que conlleva que se produzcan desbordamientos (*overflows*) o agotamiento (*underflows*). Por otra parte, la limitación del número de bits usados para representar la mantisa provoca una falta de precisión en la representación. Esto es debido a que cada dato de tipo real es la imagen de un conjunto infinito de números reales, concretamente representa a un intervalo de la recta real.

Para los datos de tipo real están definidas las operaciones aritméticas. La suma y la multiplicación de datos de tipo real cumplen la propiedad conmutativa, pero no siempre la asociativa ni la distributiva. Esto sucede porque el orden en que se realizan las operaciones influye en el resultado. En cada operación se producen errores por falta de precisión en la representación (errores de redondeo), que se acumulan durante todo proceso de cálculo.

Usualmente, en una computadora se pueden utilizar diversas representaciones para los datos reales: simple, doble o cuádruple precisión. En cada uno de estos tipos el número de bits dedicados a representar la mantisa, y a veces, también el exponente es diferente.

2.3 Datos de tipo lógico.

Los datos de tipo lógico representan valores lógicos o booleanos. Pueden tomar uno de entre dos valores: verdadero o falso (abreviadamente V, F ó 0, 1).

Sobre los valores lógicos pueden actuar los llamados operadores lógicos. Los operadores lógicos fundamentales son: Y, O y NO (en inglés: AND, OR y NOT, respectivamente). La definición de las operaciones se hace indicando su resultado para las cuatro posibles combinaciones de valores de los argumentos. En algunos lenguajes de programación hay definidos sobre los datos de tipo lógico otros operadores booleanos, como son: NO-Y, NO-O y O-exclusivo (en inglés NAND, NOR y XOR).

a	b	$a \wedge b$ *(a Y b)*	$a \vee b$ *(a ó b)*	$a \uparrow b$ *(a NoY b)*	$a \downarrow b$ *(a NoO b)*	$a \oplus b$ *(a Xor b)*	$\tilde{a}$ *(No a)*
0	0	0	0	1	1	0	1
0	1	0	1	1	0	1	1
1	0	0	1	1	0	1	0
1	1	1	1	0	0	0	0

Un caso particularmente importante de valor de tipo lógico es el obtenido como resultado de una operación de relación sobre datos de un tipo para el que existe una relación de orden. Son operadores de relación los siguientes: $\geq$, $>$, $\leq$, $<$, $=$, $\neq$. El resultado de una operación de relación es el valor lógico verdadero si la relación es cierta, y falso en caso contrario.

2.4 Datos de tipo carácter.

Los datos de tipo carácter representan elementos individuales de conjuntos finitos y ordenados de caracteres. El conjunto de caracteres representado depende de la computadora. Uno de los conjuntos más usuales es el ASCII.

No hay ninguna operación interna sobre datos de tipo carácter (salvo la asignación). Normalmente existen funciones de conversión de tipo.

2.5 Datos de tipo enumerado.

Los datos de tipo enumerado se definen explícitamente dando un conjunto finito de valores. El tipo de datos enumerado, como el tipo subrango y los tipos estructurados, no es un tipo normalizado. Puede haber muchos tipos de datos enumerados distintos dentro de un programa en un lenguaje determinado. Podemos considerar al tipo de dato enumerado como una clase de tipos de datos, a la que pertenecerán todos los tipos definidos por enumeración.

Los tipos de datos vistos en las secciones anteriores son usualmente tratados por la computadora a nivel hardware. Mientras que el tipo de datos enumerado y el tipo de datos subrango sólo son interpretados por el software. Internamente los datos de tipo enumerado se almacenan como valores enteros. A cada valor del tipo se le asocia un entero consecutivo, comenzando por cero. Existen, como en el tipo carácter funciones de conversión a entero. Pueden también existir funciones que generen el valor siguiente (sucesor) o anterior (predecesor) a uno dado, según el orden en que éstos se definieron.

2.6 Datos de tipo subrango.

El tipo subrango se define a partir del tipo entero, carácter o de un tipo enumerado. Un dato de tipo subrango puede tomar determinados valores del tipo original, a partir del cual se ha definido el subrango, entre un mínimo y un máximo. Con datos de tipo subrango se pueden realizar las operaciones definidas para el tipo original.

3 Estructuras de datos.

Los tipos de datos anteriores se suelen denominar elementales, se pueden utilizar para construir tipos de datos más elaborados. Una estructura de datos o tipo de dato estructurado es un tipo de dato construido a partir de otros tipos de datos. Un dato de un tipo estructurado está compuesto por una serie de datos de tipos elementales y alguna relación existente entre ellos.

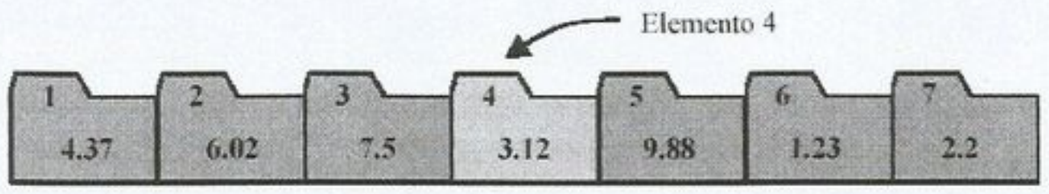
Una estructura de datos se dice que es homogénea cuando todos los datos elementales que la forman son del mismo tipo. En caso contrario se dice que la estructura es heterogénea.

Siempre que se utilice un dato en un programa debe estar determinado su tipo, para que el traductor sepa cómo debe tratarlo y almacenarlo. En el caso de datos de tipos elementales, el tipo del dato determina el espacio que se usa en memoria. Esto puede no ocurrir si el dato es de un tipo estructurado. Algunos tipos estructurados (listas y árboles) se declaran sin especificar el número de componentes que van a tener. En este caso el compilador les reserva el espacio de memoria mínimo que necesitan. Durante la ejecución del programa la estructura de datos puede ir creciendo. Una estructura de datos que es gestionada de esta forma se dice que es dinámica, ya que la memoria que necesita se asigna dinámicamente. Por el contrario, una estructura de datos que siempre ocupa el mismo espacio se dice que es estática. Los tipos de datos más utilizados son: arrays, cadenas de caracteres, registros, listas y árboles.

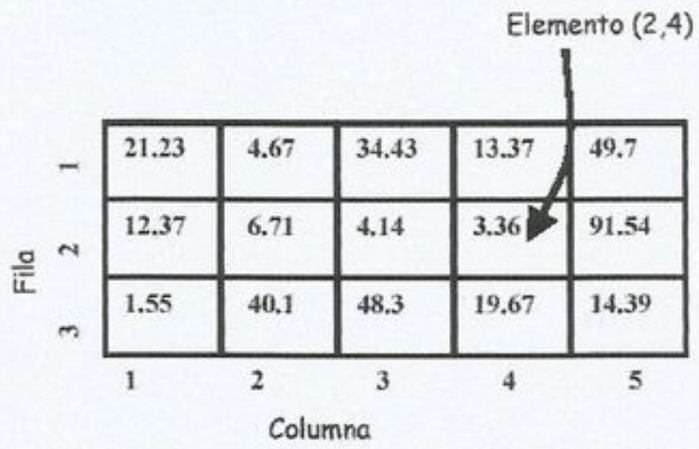
3.1 Arrays.

Un array es una estructura de datos formada por una cantidad fija de datos de un mismo tipo, cada uno de los cuales tiene asociado uno o más índices que determinan de forma unívoca la posición del dato en el array. Cada índice es un dato de tipo subrango. Para cada combinación posible de valores de índices existe uno y sólo un dato del tipo constituyente, o elemento del array.

Podemos imaginar un array como una estructura de celdas donde se pueden almacenar valores. La siguiente figura representa una matriz de un solo índice que toma valores de 1 a 7.



El array de la siguiente figura utiliza dos índices con valores entre 1 y 3, el primero, y entre 1 y 5, el segundo. Cada elemento de esta matriz está especificado por un par ordenado de números, el valor de los dos índices.



En general, al número de índices del array se le denomina número de dimensiones del array. La dimensión de la formación está dada por los valores máximos de los índices, y el número total de elementos es el producto de estos valores máximos.

La principal operación que se puede realizar con arrays es la selección. La selección consiste en especificar un elemento determinado del array. Esta operación se efectúa dando un valor para todos y cada uno de los índices del array. Con el elemento seleccionado se pueden realizar las operaciones propias de su tipo.

Un array es una estructura de datos estática. Esto es, al definirla se especifica el número de elementos que la constituyen. Este dato lo utiliza el compilador para reservar el espacio necesario para almacenarla. Las matrices se almacenan en memoria ocupando un área contigua. Cada elemento ocupa el mismo número de palabras, que será el que corresponda al tipo de éstos. Los elementos se colocan en la memoria según un orden prefijado de los índices.

3.2 Cadenas de caracteres.

Una cadena de caracteres (*string*) es una estructura de datos formada por una secuencia de caracteres. Las constantes de este tipo se escriben normalmente entre comillas.

Sobre datos de tipo cadena de caracteres se pueden realizar las siguientes operaciones:

- Concatenación. Consiste en formar una cadena a partir de dos ya existentes, yuxtaponiendo los caracteres de ambas.
- Extracción de subcadena. Permite formar una cadena a partir de otra ya existente. La subcadena se forma tomando un tramo consecutivo de la cadena inicial.
- Comparación de cadenas. Es posible comparar dos cadenas en conjunto. Se considera menor aquella en que el primer carácter en que difieren ambas es menor.

- Obtención de longitud. La longitud de una cadena es un dato de tipo entero, cuyo valor es el número de caracteres que contiene.

3.3 Registros.

Un registro es una estructura de datos formada por yuxtaposición de elementos que contienen información relativa a un mismo ente. A los elementos que componen el registro se les denomina campos. Cada campo es de un determinado tipo. Los campos dentro del registro aparecen en un orden determinado, y se identifican por un nombre. Para definir un registro es necesario especificar el nombre y tipo de cada campo. Los campos pueden ser de un tipo estructurado.

Con los registros se pueden realizar asignaciones del registro completo a una variable de tipo registro, definida con los mismos campos en el mismo orden. Se puede realizar, al igual que en arrays, la selección de un campo. Esto se realiza especificando el nombre del campo.

3.4 Listas.

Una lista está formada por un número variable de datos (elementos) de un mismo tipo, ordenados según una secuencia lineal. Cada elemento, salvo el primero, tiene un predecesor en la lista. Todos los elementos, salvo el último, tienen un sucesor. La lista es una estructura dinámica. Formalmente, podemos definir una lista como una estructura de datos formada por registros de, al menos, dos campos, en que uno de ellos contiene información que permite localizar al siguiente registro en la lista según una secuencia dada.

Con una lista se pueden realizar las siguientes operaciones:

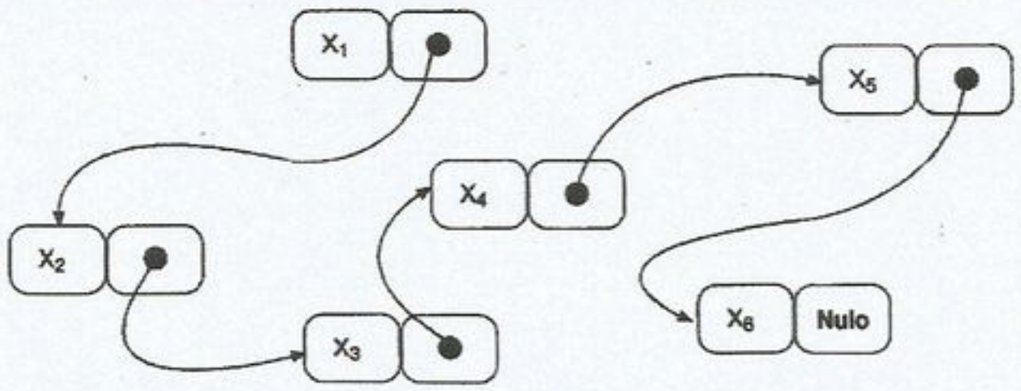
- Añadir un elemento. Esto puede realizarse al final de la lista, al principio, o entre dos elementos (inserción).
- Eliminar un elemento. Al eliminar un elemento no se pierde la secuencia lógica. Esto es, el predecesor del elemento eliminado pasa a ser el predecesor del elemento siguiente al eliminado.
- Acceder al primer elemento de la lista. El primer elemento es normalmente el único al que se puede acceder directamente.
- Acceder al elemento siguiente del último procesado. Éste es el mecanismo normal de acceso a la lista. Al acceder a un elemento éste no se elimina. La lista se puede leer desde el comienzo tanta veces como sea necesario.
- Saber si la lista está vacía. La lista está vacía si no contiene ningún elemento.

La lista no es direccionable, tan sólo se puede recuperar un elemento accediendo previamente a los que le anteceden y, por tanto, en cada momento hay sólo un elemento en disposición de ser procesado.

Un caso particular de lista es aquél en que se añaden y elimina elementos sólo en un extremo. Se denomina pila (*stack*) o lista LIFO (*Last Input First Output*) a una lista en que las inserciones y eliminaciones se realizan sólo al principio de la lista. Es decir, cualquier elemento añadido pasa a ser el primero de la lista. Además, no se puede eliminar más que el elemento que ocupa el primer lugar de la lista en ese momento. Las pilas se utilizan en hardware y software para almacenar las direcciones de instrucciones desde las que se hacen llamadas a subrutinas.

Se denomina cola a una lista en que las inserciones se realizan sólo en el final y sólo se puede acceder o eliminar en un instante dado el primer elemento de la lista. Las colas se suelen denominar también listas FIFO (*First In First Out*).

Las listas se memorizan utilizando punteros. Un puntero es un dato que contiene una dirección de memoria. Para cada elemento de la lista se almacena junto a dicho elemento un puntero que contiene la dirección del elemento siguiente. La siguiente figura muestra esquemáticamente cómo se puede almacenar una lista en memoria.

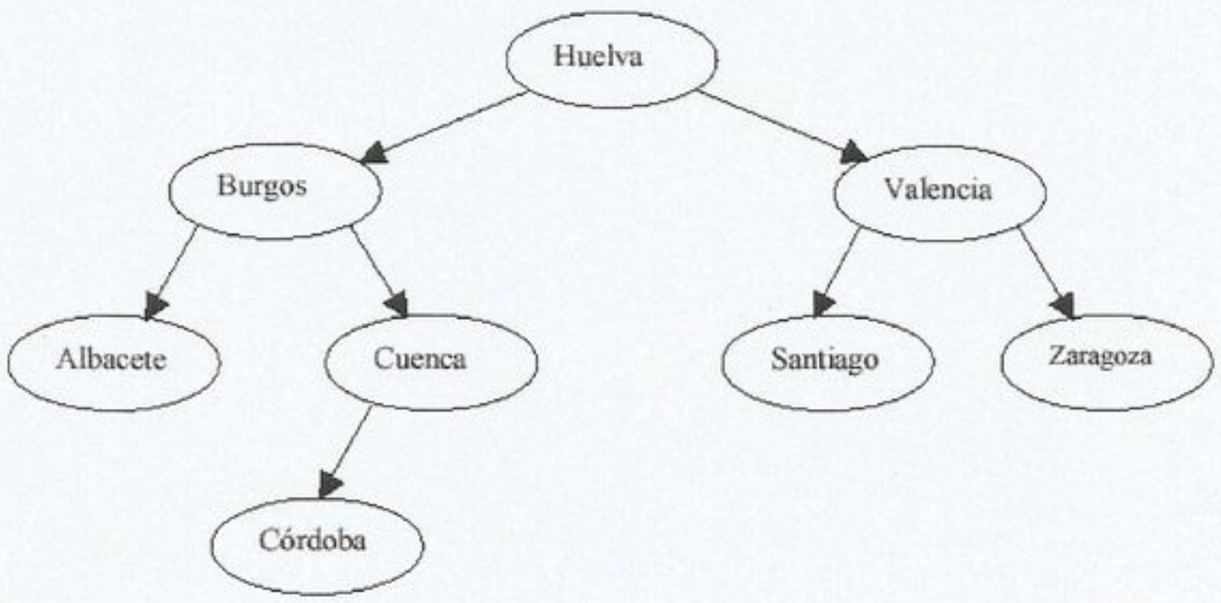


3.5 Árboles.

Un árbol es una estructura de datos formada por elementos del mismo tipo, llamados nodos, relacionados de tal modo que el árbol puede descomponerse en:

- Un nodo, llamado raíz,
- Un conjunto finito de objetos de tipo árbol, llamados subárboles del nodo raíz.

En la siguiente figura se muestra la representación gráfica de un árbol.



Se denomina hijo de un nodo a cada uno de los nodos que dependen de él. En la figura los nodos "Burgos" y "Valencia" son hijos del nodo "Huelva". Se dice también que "Huelva" es el padre de estos nodos. Se denomina grado de un nodo al número de subárboles que sustenta. En la figura el grado de "Huelva", "Burgos" y "Valencia" es dos, el de "Cuenca" es uno, y el de los demás es cero. El orden de un árbol es el mayor de los grados de sus nodos.

Un árbol es una estructura dinámica. Su representación en el interior de una computadora se realiza utilizando punteros. Cada nodo puede incluir varios punteros: uno para dirigirse al padre, y cada uno de los restantes para dirigirse a cada uno de los hijos. Esto permite moverse con gran facilidad dentro del árbol en cualquier dirección

(hacia arriba o hacia abajo), pero presenta el inconveniente de que el número de punteros para cada nodo no es fijo y puede ser excesivamente grande.

El árbol de la figura anterior es un árbol de orden dos o binario que se puede representar en memoria tal y como se muestra en la siguiente figura.

Dirección	Información del nodo	Punteros padre	Primer hijo	Segundo hijo
1	Huelva	-	2	5
2	Burgos	1	4	7
3	Santiago	5	-	-
4	Albacete	2	-	-
5	Valencia	1	3	8
6	Córdoba	7	-	-
7	Cuenca	2	6	-
8	Zaragoza	5	-	-

Normalmente se utiliza otra estructura con sólo tres punteros por nodo, uno para el padre, otro para el primer hijo, y el tercero para el siguiente hermano.

4 Representación interna de los distintos tipos de datos.

El resultado final de la representación externa a interna depende del lenguaje de programación; así, por ejemplo en FORTRAN se contemplan siete tipos de datos: entero, real simple precisión, real doble precisión, complejo simple precisión, complejo doble precisión, lógico, y carácter.

Por otra parte, estos datos irán estructurados en palabras (una o más). Los datos complejos se representan por parejas de números reales almacenados en posiciones consecutivas de memoria.

Los datos de tipo lógico representan un valor del álgebra de Boole binaria; es decir, 0 (falso) ó 1 (verdad). La representación interna de este tipo de datos es muy variada. Así, en el IBM 370 el 1 (0) lógico se representa haciendo 1 (0) el bit situado más a la derecha de la palabra.

Los datos tipo carácter, representan sencillamente cadenas de caracteres ensambladas en las palabras del ordenador y representados según el código de E/S. Para este tipo de datos, por tanto, no es necesario efectuar una reconversión de representación.

Los datos de tipo entero y tipo real son los más utilizados, y la mayoría de otros tipos y estructuras de datos se basan en ellos, por eso analizamos su representación en detalle en las siguientes secciones.

5 Representación de textos.

Podemos representar cualquier información escrita (texto) por medio de caracteres. Los caracteres que se utilizan en informática suelen agruparse en cinco categorías:

1. Caracteres alfabéticos: son las letras mayúsculas y minúsculas del abecedario inglés: A, B, C, D, E,.... X Y Z, a, b, c, d,.... x, y, z
2. Caracteres numéricos: están constituidos por las diez cifras decimales: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.
3. Caracteres especiales: son símbolos ortográficos y matemáticos no incluidos en los grupos anteriores.

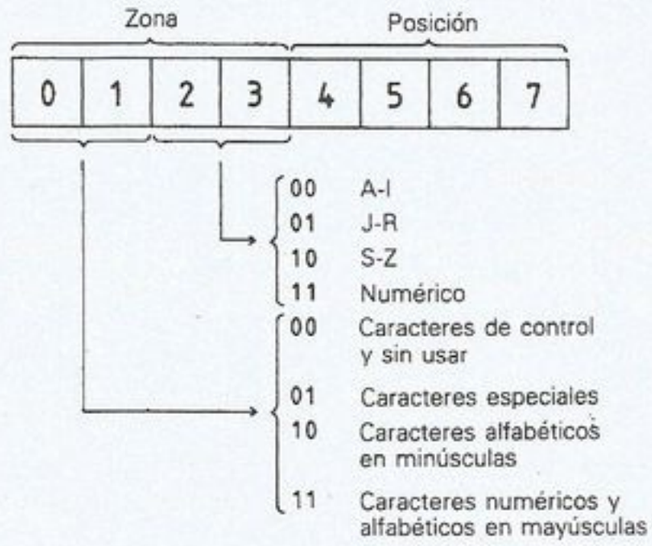
4. Caracteres geométricos y gráficos: Son símbolos o módulos con los que se pueden representar cuadros, formas geométricas o iconos elementales.
5. Caracteres de control: representan órdenes de control, como el carácter para pasar a la línea siguiente, para ir al comienzo de una línea, etc. Los caracteres de control son insertados en los textos por los usuarios o por los programas de control de periféricos o de comunicación.

Al introducir un texto en un computador, a través del periférico correspondiente, los caracteres se codifican con un código de entrada/salida de forma que a cada carácter se le asocia una determinada combinación de *n* bits. La codificación de un texto en un código es muy sencilla ya que consiste en cambiar cada carácter por su código correspondiente, sin más.

Los códigos más utilizados actualmente son EBCDIC, ASCII, y UNICODE.

5.1 Código EBCDIC.

El código EBCDIC (*Extended Binary Coded Decimal Interchange Code*) utiliza 8 bits para representar cada carácter, con el significado que se indica a continuación.



Este tipo de codificación permite codificar hasta 2^8=256 símbolos distintos, es decir, posibilita que se representen una gran variedad de caracteres: incluye las letras minúsculas y mayor número de caracteres especiales. También es posible, y se hace con las combinaciones que empiezan por 00, codificar caracteres de control.

5.2 Código ASCII.

El código ASCII (*American Standard Code for Information Interchange*) básico utiliza 7 bits y hoy día es de los más usados. Se puede decir que la mayor parte de las transmisiones de datos entre dispositivos se realizan en esta codificación, y corresponde a la normalización ANSI X3.4-1968 o ISO 646. Usualmente se incluye un octavo bit para detectar posibles errores de transmisión o grabación (bit de paridad).

Existen numerosas versiones ampliadas de este código que utilizan 8 bits y respetan los códigos normalizados del ASCII básico, aprovechando las combinaciones no usadas para representar símbolos adicionales. Entre ellas se encuentran los códigos ISO 8859-n, donde *n* es un número que identifica el juego de los nuevos caracteres introducidos, dependiendo de los lenguajes. Por ejemplo, la norma, ISO 8859-1, también denominada ISO Latín-1, se proyectó para América y Europa occidental, e incluye vocales con acentos, tildes con signos diacríticos, y otras letras latinas no usadas en los países anglosajones. Corresponde a la página de códigos 819 de los PC. En ISO

Latín-1 quedan combinaciones libres, que pueden usarse para codificar otros caracteres obteniéndose así nuevos códigos no normalizados, pero compatibles con el ISO Latín-1, como es la página de códigos 850 de los PC.

5.3 Unicode.

Unicode es un código de E/S propuesto por un consorcio de empresas y entidades que trata de hacer posible escribir aplicaciones que sean capaces de procesar texto de muy diversos sistemas de escritura. Está reconocido como estándar ISO/IEC 10646, y trata de ofrecer las siguientes propiedades:

1. Universalidad, ya que persigue cubrir la mayoría de lenguajes escritos existentes en la actualidad.
2. Unicidad, ya que a cada carácter se le asigna exactamente un único código.
3. Uniformidad, ya que todos los símbolos se representan con un número fijo de bits, concretamente 16.

Algunas características de Unicode son:

- Cada carácter está formado por una cadena de 16 bits, pudiendo, por tanto, codificarse en total 2^{16}=65356 símbolos.
- No se contempla la codificación de caracteres de control.
- Incluye caracteres combinados, es decir, caracteres que van asociados arriba o abajo con otro símbolo. También la combinación de caracteres está permitida para poder crear nuevos elementos de texto, pero debe realizarse por software.
- Unicode no determina la forma o imagen concreta de cada carácter, sino que cada combinación representa un concepto abstracto. Un mismo carácter puede ser escrito de distintas formas, así en arábigo una misma letra tiene 4 formas diferentes dependiendo de si se escribe aisladamente, al principio, en medio o al final de una palabra. Todas estas variantes se codifican con una única combinación. No ocurre lo mismo con las letras mayúsculas y minúsculas de los caracteres latinos que tienen códigos distintos.
- También con la misma idea de evitar duplicidades, caracteres muy parecidos en idiomas distintos, tienen igual posición en el código. Esto ocurre, por ejemplo, con los ideogramas japoneses, chinos y coreanos; aunque su imagen sea distinta, si su significado es el mismo tienen igual código.

Unicode se está utilizando cada vez más, y los sistemas operativos Windows NT y Windows 2000, lenguajes como Java, y aplicaciones como Netscape lo reconocen. Es previsible que las nuevas versiones de lenguajes de programación introduzcan un nuevo tipo de datos específico para caracteres Unicode. La utilización de Unicode facilitará la compatibilidad de programas y datos a través de todo el mundo.

6 Representación de datos numéricos.

Los códigos de E/S representan los datos numéricos como cualquier otra secuencia de caracteres. Si se va a realizar algún cálculo matemático la representación de los datos numéricos como textos es inapropiada. Cuando queramos realizar un cálculo matemático, obviamente lo mejor es representar los datos numéricos en alguna forma que esté basada en los sistemas de numeración matemáticos.

La solución adoptada para representar datos numéricos es la siguiente. Cuando se introduce en la computadora un número se codifica y almacena como un texto o cadena de caracteres cualquiera. Ahora bien, dentro de un programa, cada dato tiene asociado un tipo de dato determinado. El programador debe asociar a cada dato o variable el tipo adecuado, en consonancia con las operaciones que se realicen con él. Así, por ejemplo, y por lo que respecta a datos numéricos, en el lenguaje C los principales tipos de datos aritméticos son los que se indican en la siguiente tabla. El programador elige el tipo de dato más adecuado de acuerdo con los objetivos de la variable que define, y teniendo en cuenta que cuanto mayor es el rango y precisión del número, más ocupará la variable en la memoria.

Tipo		*Nº de bits*	*Rango de valores*	*Precisión (dígitos decimales)*
Tipos enteros	Carácter	8	-128,127	3
	Carácter sin signo	8	0 a 255	3
	Entero corto	16	-32.768 a 32.767	3
	Entero corto sin signo	16	0 a 65.535	5
	Enumerado	16	-32.768 a 32.767	5
	Entero	*	*	*
	Entero sin signo	*	*	*
	Entero largo	32	$-2.147_{1}484.648$ a 2.147_{1} 484.648	10
	Entero largo sin signo	32	0 a $4.294_{1}967.295$	10
Tipos reales	Coma flotante	32	±[3,4E-38 a 3,4E38], 0	7
	Coma flotante doble	64	±[1,7E-308 a 1,7E308], 0	15
	Coma flotante doble largo	80	±[3,4E-4932 a 1,1E4932], 0	19

* En máquinas de 16 bits igual a entero corto, y en máquinas de 32 bits a entero largo

Cada tipo de datos ocupa un número determinado de bits. En cualquier caso, el dato completo debe encajar en palabras de memoria, de forma que si el tamaño del dato es mayor que la longitud de palabra de memoria, aquél se trocea convenientemente. Una vez dividido el patrón de bits que representa al dato en porciones que encajen exactamente en posiciones sucesivas de memoria, dichas porciones se almacenan en direcciones consecutivas de memoria; pero hay dos posibilidades de almacenamiento:

- Almacenar primero (en las direcciones más bajas de memoria) la parte menos significativa del dato; este convenio se denomina criterio del extremo menor.
- Almacenar primero la parte más significativa del dato; éste es el criterio del extremo mayor.

Los mismos criterios se utilizan para almacenar cadenas de caracteres. Aunque seguir un criterio u otro es irrelevante, no hay un convenio que establezca cuál de los dos debe utilizarse, así hay computadoras que siguen el primero y otras el segundo.

Una vez definidos los datos numéricos de un programa o de una aplicación, una rutina de la biblioteca del lenguaje de programación se encarga de transformar la cadena de caracteres que simboliza el número en la representación numérica.

Hay dos formas básicas de representar los datos numéricos: como números enteros o como números reales. La representación de números reales se suele conocer también como representación en coma flotante o representación científica o representación exponencial.

6.1 Datos de tipos enteros.

Se distinguen dos formas básicas de representar en el interior de la computadora los datos de tipos enteros: representación binaria y representación de dígitos decimales codificados en binario (o representación BCD). A su vez, dentro de la representación binaria se tienen dos situaciones, representación sin signo y representación con signo. En este último caso es usual considerar cuatro casos diferentes: signo y magnitud, complemento a 1, complemento a 2, y representación sesgada (o por exceso). Los lenguajes de programación admiten simultáneamente distintas representaciones para los números enteros, y el programador elige para cada variable que define la más adecuada.

La forma más utilizada en la actualidad para representar los números enteros con signo es la de complemento a dos. Ello se debe a la facilidad de efectuar las sumas y restas con este tipo de representación. La representación sesgada se utiliza para almacenar los exponentes de los datos de tipo real.

6.1.1 Enteros sin signo.

En este caso todos los bits del dato representan el valor del número expresado en binario natural (sistema de numeración base 2). En general, como hay n bits para representar el número, los valores mayor y menor representables son: $N(\text{mín})=0$, $N(\text{máx})=2^n-1$.

6.1.2 Enteros en signo y magnitud.

El signo se representa con el bit más significativo del dato (bit n-1). Éste bit es 0 si el número es positivo y 1 si el número es negativo. El resto de los bits representan el valor absoluto del número en binario natural.

En general, como hay n-1 bits para representar la magnitud del número, los valores mayor y menor representables son: $N(\text{mín})=-(2^{n-1}-1)$, $N(\text{máx})=2^{n-1}-1$.

Hay dos representaciones para el cero. Este hecho puede dar lugar a problemas ya que si se desea comprobar si el resultado de una operación es 0, habrá que comprobar tanto si es igual a +0 como a -0, caso de que esta eventualidad no haya sido prevista por el compilador o la unidad aritmético lógica.

6.1.3 Enteros en complemento a uno.

El signo se representa de la misma forma que en el caso de signo y magnitud. El resto de los bits representan, si el número es positivo el valor absoluto del numero en binario natural, y si no su complemento a 1. En general, como hay n-1 bits para representar la magnitud del dato, los números mayor y menor representables son: $N(\text{mín})=-(2^{n-1}-1)$, $N(\text{máx})=2^{n-1}-1$. Al igual que ocurre en la representación en signo y magnitud hay dos representaciones para el cero; teniéndose que realizar las mismas previsiones que en aquel caso.

6.1.4 Enteros en complemento a dos.

El signo se representa de la misma forma que en el caso de signo y magnitud. El resto de los bits representan, si el número es positivo el valor absoluto del número en binario natural, y si no su complemento a 2. Aquí no hay dos representaciones para el cero, teniendo una combinación más para representar un número negativo más. En este caso, por tanto, los números mayor y menor representables son: $N(\text{mín})=-2^{n-1}$, $N(\text{máx})=2^{n-1}-1$.

6.1.5 Representación sesgada

En la representación sesgada, sencillamente se le suma al número N un sesgo S, de forma tal que el número resultante siempre es positivo, no siendo necesario reservar explícitamente un bit de signo.

El dato almacenado es sencillamente el valor de $N+S$ en binario natural. Usualmente se toma como sesgo $S=2^{n-1}$. Esta notación también se le suele denominar representación con exceso. En este caso los números mayor y menor representables son: $N(\text{mín})=-2^{n-1}$, $N(\text{máx})=2^{n-1}-1$. También hay que hacer notar que en la representación sesgada los números positivos empiezan por 1 y los negativos por 0.

6.1.6 Datos enteros representados con dígitos decimales codificados en binario (BCD).

En ocasiones, los datos de tipo entero se representan internamente codificando aisladamente cada dígito decimal con cuatro dígitos binarios, según la siguiente tabla.

Dígito decimal	**Valor binario**
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

De esta forma, en un byte se pueden representar 2 dígitos decimales, denominándose esta representación BCD empaquetada, o bien un único digito decimal, obteniéndose una representación BCD desempaquetada.

Esta forma de codificar es poco eficiente, puesto que de las $2^4=16$ combinaciones posibles de 4 bits, sólo se utilizan 10. No obstante; a veces se utiliza por la proximidad a nuestro sistema decimal y por la gran facilidad de codificar en BCD, sin más que considerar aisladamente cada dígito decimal. Los circuitos decodificadores y los algoritmos de transformación de código de E/S a BCD son muy sencillos.

En la representación BCD de datos con signo se suelen utilizar cuatro bits para representar al signo. Puede ser, por ejemplo, 0000 para el signo positivo y 1001 para el signo negativo.

6.2 Datos de tipos reales. Normalización IEEE 754.

Cuando se opera con números muy grandes se suele utilizar la notación exponencial. Según esta notación todo número N lo podemos expresar en la forma $N=M\cdot B^E$. Podemos transformar la representación del número N, conservando su valor, cambiando el exponente, E, y reajustando adecuadamente la mantisa, M: si aumentamos (disminuimos) en una unidad E, debemos dividir (multiplicar) M por B.

La representación y manejo de los datos puede ser responsabilidad del hardware de la computadora o de los traductores de lenguajes. Los microprocesadores actuales disponen internamente de un procesador de coma flotante (FPU, o *Float Point Unit*) que

contiene circuitos aritméticos para operar con este tipo de datos. En la década de los ochenta, cuando el microprocesador no incluía la FPU, existían coprocesadores aritméticos (por ejemplo, los circuitos integrados 8087, 80287 y 80387) que contenían la FPU. Si el hardware no dispone de circuitería para coma flotante, y un lenguaje de programación dispone de este tipo de datos, diversas rutinas de la biblioteca de programas del traductor correspondiente descomponen las operaciones en coma flotante en términos de las operaciones que presenta el lenguaje máquina, obteniéndose en este caso un rendimiento (velocidad) mucho menor en la ejecución de los programas.

Hasta la década de los años ochenta cada fabricante utilizaba un sistema propio para la representación de números reales; pero enseguida se observó la necesidad de algún sistema normalizado. Con este objetivo, de 1977 a 1985 la asociación IEEE desarrollo un sistema normalizado de representación, denominado Normalización IEEE 754 que afortunadamente hoy día tiene una aceptación prácticamente universal.

La base del exponente es B=2, es decir, está predeterminada, por lo que sólo es necesario almacenar M y E, con sus signos respectivos. No se almacena directamente el signo, el exponente y la mantisa, sino que estos elementos sufren una transformación: realmente se memoriza lo que se denomina campo del signo (s), campo del exponente o característica (e), y campo de la mantisa (m). Concretamente se utiliza un bit, s, como campo del signo del número, un número fijo de bits, ne, para almacenar el campo del exponente (incluyendo su signo), y otro cierto número fijo de bits, nm, para almacenar el campo de la mantisa. Es decir, siendo n el número total de bits utilizados para representar el número real, se verifica: n=1+ne+nm.

El orden de almacenamiento es: campo de signo (s), campo de exponente (e) y campo de mantisa (m). Este orden se sigue para que los elementos y bits más significativos queden ordenados de izquierda a derecha, y así los algoritmos de comparación entre números enteros sean también válidos para la representación de números reales.

El campo del signo es cero para los números positivos y uno para los números negativos.

El exponente se almacena en la forma de entero sesgado; es decir, el exponente almacenado e se obtiene sumando al exponente del número, E, un sesgo S dado por: $S=2^{ne-1}-1$, de forma que $e=S+E=2^{ne-1}+E-1$. De esta forma, en los ne bit reservados para el exponente se pueden incluir exponentes positivos o negativos sin utilizar un bit explícito de signo.

Por lo general, el exponente se ajusta de forma tal que el 1 más significativo de la mantisa se encuentre en la posición 0 (posición de las unidades); es decir, $2>M>1$. Cuando el número se encuentra ajustado de esta forma se dice que está normalizado, en caso contrario se dice que está desnormalizado.

Como la base del exponente es B=2, si aumentamos una unidad el exponente se debe dividir por 2 la mantisa binaria, o lo que es lo mismo, desplazar la coma decimal una posición a la izquierda, y viceversa. Por el contrario, si disminuimos una unidad el exponente, se debe multiplicar por 2 la mantisa, o lo que es lo mismo, desplazar la coma decimal una posición a la derecha, y viceversa.

El campo de la mantisa se obtiene almacenando sólo la parte fraccionaría del número normalizado; es decir, no se almacena la información "1". Esto se hace así porque todos los números normalizados empiezan siempre por 1, por lo que se ahorra espacio de memoria no almacenando esta cabecera; diciéndose que el 1 está implícito, u

oculto o que el número se encuentra empaquetado. Obviamente cuando la ALU (o el traductor) realice cualquier operación, debe restituir el 1, haciéndolo explícito.

Podemos destacar las siguientes situaciones especiales.

- Cuando el campo del exponente toma su valor mínimo; es decir, e=0, el 1 más significativo de la mantisa no se encuentra implícito, y entonces la mantisa se almacena desnormalizada. En este caso el sesgo es: $S=2^{ne-1}-2$.
- El número N=0 se representa con todos los bits del campo del exponente y del campo de la mantisa a cero.
- Si todos los bits del campo del exponente son unos (es decir, adquiere su valor máximo), el dato:
  - Si m=0, representa más o menos infinito.
  - Si $m\neq0$, no representa un número (es un código NaN, *No a Number*). Estos patrones de bits se utilizan para almacenar valores no válidos.

Un problema que se plantea al representar números reales es que, por lo general, un numero decimal real, incluso aunque tenga un número finito de cifras significativas, no puede ser representado exactamente con un número fijo, nm+1, de cifras binarias, lo que implica tener que utilizar técnicas de redondeo. También la ALU, en la realización de cálculos intermedios, debe realizar redondeos cuando el resultado de una operación dé lugar a un número de bits mayor de los utilizados en la representación del número.

La normalización IEEE 754 recomienda efectuar un redondeo al más próximo; y si el error es igual en ambos sentidos se hace un redondeo al par, que consiste en redondear por defecto o por exceso, pero siempre de forma que el bit menos significativo del número resultante sea 0. En otras palabras, si la cifra menos significativa que se retiene es 0: se trunca el número, y si es 1, se le suma un 1. Como, en general, existe igual probabilidad de que la cifra menos significativa sea 0 o 1, el sistema de redondeo al par es equitativo, ya que en la situación problemática, por término medio el 50 por 100 de las veces, se hace el redondeo por defecto y el otro 50 por 100 por exceso.

El estándar IEEE 754 considera cuatro tamaños o precisiones posibles de datos (cuanto mayor sea nm, mayor es el número de cifras significativas y, por tanto, mayor es la precisión): simple precisión (n=32), simple ampliada, doble y doble ampliada; aunque el estándar sólo especifica completamente las precisiones sencilla y doble.

	Precisión			
	Simple	**Simple ampliada**	**Doble**	**Doble ampliada**
nm + 1 (bits de precisión)	24	32	53	64
E (máx.)	127	1.023	1.023	16.383
E (mín)	–126	–1.022	–1.022	–16.383
S (sesgo del exponente)	127	(n.e.)	1.023	(n.e.)

(n.e.: no especificado por el estándar)

7 Representación de sonidos

Una señal de sonido se capta por medio de un micrófono que produce una señal analógica esto es, una señal que puede tomar cualquier valor dentro de un determinado intervalo continuo. Posteriormente, la señal analógica es amplificada para encajarla dentro de dos valores límites, por ejemplo entre -5 voltios y +5 voltios.

En un intervalo de tiempo continuo se tienen infinitos valores de la señal analógica, por lo que para poder almacenarla y procesarla utilizando técnicas digitales se realiza un proceso de muestreo. El muestreo selecciona muestras de la señal analógica a una frecuencia F_s determinada; así cada $T_s=1/F_s$ segundos se dispone de un valor de la señal.

Simultáneamente al muestreo, las muestras se digitalizan (transforman a binario) con un conversor analógico/digital. En definitiva, la señal de sonido queda representada por una secuencia de valores, por ejemplo de 8 bits, correspondiendo cada uno de ellos a una muestra analógica.

A partir de las muestras digitales se puede recuperar la señal. En el muestreo intervienen fundamentalmente dos parámetros:

1. La frecuencia de muestreo, F_s, debe ser igual o superior a un determinado valor que depende de la calidad del sonido a recuperar; en otras palabras, dentro de un intervalo de tiempo dado deben tomarse suficientes muestras para no perder la forma de la señal original.
2. El número de bits (precisión) con el que se representa cada muestra debe ser el adecuado.

Obviamente, cuanto mayor es la frecuencia de muestreo y el número de bits por muestra, mayor será el volumen de los archivos que almacenan el sonido; por lo que ambos parámetros deben elegirse en función de la calidad requerida. En la siguiente tabla se especifican parámetros usuales para obtener distintas calidades de sonido.

	N° de bits/muestra	Frecuencia de muestro (F_s, KHz)	Periodo de muestreo (T_s, µsegundos)
PCM teléfono	8	8	125
Calidad telefónica	8	11,025	90,7
Radio	8	22,05	45,4
CD	16[1]	44,1	22,7

[1] Numero de bits/muestra por canal, con sonido estereofónico hay que multiplicar por 2

8 Representación de imágenes.

Las imágenes se adquieren por medio de periféricos especializados tales como escáneres, cámaras de vídeo o cámaras fotográficas. Como todo tipo de información, una imagen se representa por patrones de bits, generados por el periférico correspondiente.

Desafortunadamente hay sistemas de codificación de imágenes muy diversos. En la siguiente tabla se describen algunos de ellos.

Tipo	Formato	Origen	Descripción
Mapa de bits	BMP (BitMap)	Microsoft	Usado en aplicaciones Windows
	PICT (PICTure)	Apple Comp.	Usado en Macintosh
	TIFF (Tagged Image File Formats)	Microsoft y Aldus	Usado en PC y Macintosh, muy poco compatible con otros formatos.
	JPEG (Joint Photographic Experts Group)	Grupo JPEG	Muy buena calidad para imágenes naturales. Incluye compresión, Muy usado en la web
	GIF (Graphic Interchange Format)	CompuServe	Incluye compresión. Muy usado en la web.
	PNG (Portable Network Graphics)	Consorcio www	Evolución de GIF. Muy buena calidad de colores. Incluye muy buena compresión
Mapa de vectores	DXF (Document eXchange Format)		Formato normalizado para imágenes CAD (AutoCAD , CorelDRAW, etc.)
	IGES (Initial Ghaphics Exchange Specification)	ASME/ANSI	Formato normalizado para modelos CAD (usable en AutoCAD , CorelDRAW, etc.)
	EPS (Encapsulated Poscript)	Adobe Sys.	Ampliación para imágenes del lenguaje Poscript de impresión.
	TrueType	Apple comp....	Alternativa de Apple y Microsoft para el EPS

8.1 Mapas de bits.

Una imagen está compuesta por infinitos puntos, y a cada uno de ellos se le asocia un atributo que puede ser su nivel de gris, en el caso de una imagen en blanco y negro, o su color, si la imagen es en color. En consecuencia, para codificar y almacenar la imagen hay que tener en cuenta dos factores: número de puntos a considerar y código de atributo asociado a cada uno de ellos. Como no podemos almacenar y procesar los atributos de los infinitos puntos, los sistemas de captación consideran la imagen dividida en una fina retícula de celdas o elementos de imagen (píxeles), y a cada uno de ellos se le asigna como atributo el nivel de gris medio de la celda o el color medio de la celda correspondiente.

La resolución de la imagen, o (número de elementos por línea) x (número de elementos por columna) determina la calidad de imagen.

La imagen de una fotografía típica también se forma por puntos, y representándola con una resolución del orden de 1280 x 1024 el ojo humano la considera como continua. En la siguiente tabla se indican las resoluciones usualmente utilizadas para digitalización o representación de imágenes. Obviamente el tamaño en que se capta o visualiza la imagen influye también en su calidad. Para una misma resolución, cuanto mayor es el tamaño, peor será la calidad.

		Resolución (horizontal x vertical)	*Movimiento*
Convencionales	Fax (A4)	(100, 200,400) x (200, 300, 400) ei/"	Estática
	Foto (8"x11")	128, 400, 1200 ei/pulgada	Estática
Televisión	Videoconferencia	176 x 144 ei/imagen	10 a 36 imágenes/s
	TV	720 x 480 ei/imagen	30 imágenes/s
	HDTV (TV alta definición)	1920 x 1080 ei/imagen	30 imágenes/s
Pantalla computador	VGA SVGA XGA	640 x 480 ei 800 x 600 ei 1024 x 768 ei	

Además de la resolución, un factor determinante en la representación de un gráfico es el código del atributo del punto de imagen. En el caso de imágenes en blanco y negro, se asigna un valor al nivel de gris; así, si se requieren 256 niveles de grises, por cada punto de imagen se almacenará un byte ($2^8 = 256$).

La imagen se representa sencillamente almacenando los atributos de los puntos de la imagen sucesivos en orden, de izquierda a derecha y de arriba abajo.

En el caso de imágenes en color, éste se descompone en tres colores básicos: rojo (R), verde (G) y azul (B), y la intensidad media de cada uno de ellos en cada celda se codifica por separado. Para conseguir una gran calidad de colores (calidad fotográfica), cada color básico debe codificarse con 8 bits; es decir se necesitarían 3 bytes para codificar cada elemento de imagen, lo que da lugar a que se necesiten archivos de una gran capacidad para almacenar una imagen.

Existen técnicas para la compresión de imágenes, que permite representar una imagen con un número menor de bytes, con las consiguientes reducciones del espacio de memoria para almacenarla y del tiempo para transmitirla.

8.2 Mapas de vectores.

Otros métodos de representar una imagen se fundamentan en descomponer ésta en una colección de objetos tales como líneas, polígonos y textos con sus respectivos atributos o detalles (grosor, color, etc.) modelables por medio de vectores y ecuaciones matemáticas que determinan tanto su forma como su posición dentro de la imagen. Cuando se visualiza una imagen en una pantalla o impresora determinadas, un programa evalúa las ecuaciones y escala los vectores generando la imagen concreta a ver. Algunas características de este tipo de representación son las siguientes:

- Son adecuadas para gráficos de tipo geométrico y no para imágenes reales, ya que los primeros presentan gran cantidad de elementos regulares fácilmente modelables, cosa que no ocurre con los del segundo tipo. En particular resulta muy adecuada en aplicaciones de diseño con ayuda de computador (CAD).
- En comparación con la representación con mapas de bits, la representación con mapa de vectores genera usualmente archivos que ocupan menos espacio, y las imágenes son más fáciles de reescalar a cualquier tamaño y de procesar. Por el contrario, la calidad y la fidelidad de la imagen es peor.

9 Conclusiones.

Se denomina dato a cualquier objeto manipulable por la computadora. Se denomina tipo de dato al conjunto de la transformación de la información y de las operaciones y funciones internas y externas definidas sobre el conjunto de datos.

Los tipos de datos elementales son los datos de tipo entero, los datos de tipo real, los datos de tipo lógico, los datos de tipo carácter, los datos de tipo enumerado, y los datos de tipo subrango.

Una estructura de datos o tipo de dato estructurado es un tipo de dato construido a partir de otros tipos de datos. Un dato de un tipo estructurado está compuesto por una serie de datos de tipos elementales y alguna relación existente entre ellos. Los tipos de datos más utilizados son: arrays, cadenas de caracteres, registros, listas y árboles.

El resultado final de la representación interna depende del lenguaje de programación. Los datos de tipo entero y tipo real son los más utilizados, y la mayoría de otros tipos y estructuras de datos se basan en ellos.

Para la representación de textos los códigos más utilizados actualmente son EBCDIC, ASCII, y UNICODE.

Hay dos formas básicas de representar los datos numéricos: como números enteros o como números reales. Se distinguen dos formas básicas de representar los datos de tipos enteros: representación binaria y representación de dígitos decimales codificados en binario (o representación BCD). A su vez, dentro de la representación binaria se tienen dos situaciones, representación sin signo y representación con signo. En este último caso es usual considerar cuatro casos diferentes: signo y magnitud, complemento a 1, complemento a 2, y representación sesgada (o por exceso). La notación IEEE 754 establece las normas para representar los datos de tipos reales.

Para la representación de sonidos se muestrea la señal y se digitaliza dicha muestra con un número de bits adecuado en función de la precisión requerida.

Para la representación de imágenes hay sistemas de codificación muy diversos de entre los que destacamos los mapas de bits y los mapas de vectores.