CAPÍTULO 2

CODIFICACIÓN NUMÉRICA

El objetivo de este capítulo es mostrar el concepto de código, y específicamente y más en concreto presentar unas nociones básicas sobre la forma en que se codifican las cantidades mediante números.

Concepto de código.

Si se busca en el diccionario de la Real Academia Española el significado de la palabra código, se encuentra, entre otras acepciones, las siguientes:

- "Combinación de signos que tiene un determinado valor dentro de un sistema establecido. El código de una tarjeta de crédito."
- "Sistema de signos y de reglas que permite formular y comprender un mensaje."

Podríamos decir que un **código** es una relación más o menos arbitraria

que se define entre un conjunto de mensajes o significados a codificar y un sistema de signos que significan esos mensajes de forma inequívoca. El código no es tan solo el conjunto de signos, sino también la relación que asigna a cada uno de esos signos un significado concreto.

Ejemplos de códigos hay muchos: desde el semáforo que codifica tres posibles mensajes con sus tres valores de código diferentes (rojo, ámbar y verde) hasta el sistema de signos que, para comunicarse, emplean las personas sordas. O el código de banderas, o el sistema braile para los invidentes que quieren leer.

Para establecer un código es necesario cuidar que se verifiquen las siguientes propiedades:

1. Que quede bien definido el conjunto de significados o mensajes que se quieren codificar. En el ejemplo del semáforo, queda claro que hay tres mensajes nítidos: adelante / alto / precaución. No hay confusión, ni posibilidad de equívoco en estos tres mensajes.
2. Que quede bien definido el conjunto de los elementos que van a codificar o significar esos mensajes. En el caso del semáforo queda claro cuál es el código de colores y no hay espacio para la confusión. Excepto para quien tenga algún tipo de limitación con la vista.
3. Que no haya más significados que signos. Si hay más signos que mensajes a codificar, el código es válido, pero tendremos mensajes codificados de distintas maneras (redundancias) o tendremos signos que no signifiquen nada. Si hay más mensajes que signos, el código no es válido porque tendremos mensajes que no están codificados, o tendremos signos que signifiquen varias cosas diferentes, lo que será causa de equívocos.

Lo mejor es siempre que un código se formule mediante una **aplicación biyectiva**, que establezca una relación entre significados y signos, que asigne a cada significado un signo y sólo uno, y que todo signo signifique un significado y sólo uno.

Los números como sistema de codificación de cantidades.

Para significar cantidades se han ideado muchos sistemas de representación, o códigos.

Todos conocemos el sistema romano, que codifica cantidades mediante letras. Ese sistema logra asignar a cada cantidad una única combinación de letras. Pero, por desgracia, no es un sistema que ayude en las tareas algebraicas. ¿Quién es capaz de resolver con agilidad la suma siguiente: CMXLVI + DCCLXIX?

El sistema de numeración arábigo, o indio, es en el que nosotros estamos habituados a trabajar. Gracias a él codificamos cantidades. Decir CMXLVI es lo mismo que decir 946; o decir DCCLXIX es lo mismo que decir 769. Son los mismos significados o cantidades codificados según dos códigos diferentes.

Un **sistema de numeración** es un conjunto de símbolos y reglas de generación que permiten construir todos los números válidos en el sistema. Un sistema de numeración es un código que permite codificar cantidades mediante números. Las cantidades se codifican de una manera u otra en función del sistema de numeración elegido. Un número codifica una cantidad u otra en función del sistema de numeración que se haya seleccionado.

Un número es un elemento de un código. ¿Qué codifica un número?: Un número codifica una cantidad.

En la línea inmediatamente anterior hemos dibujado una serie de puntos negros.

Cuántos son esos puntos es una cuestión que en nada depende del

sistema de numeración. La cantidad es la que es. Al margen de códigos.

En nuestro sistema habitual de codificación numérica (base 10) diremos que tenemos 7 puntos. Pero si trabajamos en el sistema binario de numeración, diremos que tenemos 111 puntos. Lo importante es que tanto la codificación 7 (en base diez) como la codificación 111 (en base dos) significan la misma cantidad.

Y es que trabajar en base 10 no es la única manera de codificar cantidades. Ni tampoco es necesariamente la mejor.

Fundamentos matemáticos para un sistema de numeración. Bases, dígitos y cifras.

Todo número viene expresado en una **base** $B > 1$.

Una base es un conjunto finito y ordenado de signos.

$B = \{\alpha_i \,/\, \alpha_0 = 0\,;\ \alpha_{i+1} = \alpha_i + 1, \forall\, i = 0, 1, \dots, B-1\}$.

Sus propiedades pueden resumirse en las tres siguientes:

- El primer elemento de la base es el cero.
- Los sucesivos elementos ordenados de la base son tales que cualquier elemento es igual a su inmediato anterior más uno.
- El máximo valor de la base es igual al cardinal de la base menos uno.

La base $B = 10$, por ejemplo, está formada por los siguientes elementos:

$B = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$.

En cada base, todo número entero $a > 0$ puede ser escrito de modo único en la forma

$$a = a_k \cdot B^k + a_{k-1} \cdot B^{k-1} + a_{k-2} \cdot B^{k-2} + \dots + a_1 \cdot B^1 + a_0 \cdot B^0. \qquad (2.1.)$$

donde $k > 0$ es un entero, y cada uno de los a_i son enteros que verifican

$0 \leq a_i \leq B-1$, para $j = 1, 2, 3, \ldots, k$, y $a_k \neq 0$ (2.2.)

A los coeficientes a_i se les llama **dígitos del número** a. A la expresión (2.1.) se la llama **expansión del número**. El número habitualmente se representa como

$a = (a_k a_{k-1} a_{k-2} \ldots a_1 a_0)_B$. (2.3.)

Cualquier número viene representado, en una base determinada, por una serie única de coeficientes a_i (ver 2.3.). A cada serie de coeficientes, que codifica un número de modo único en una determinada base, se le llama **cifra**.

En una cifra importa tanto la posición relativa de cada dígito dentro de ella, como el valor de cada uno de esos dígitos. Cuanto más larga pueda ser la serie de dígitos que se emplean para codificar, mayor será el rango de números que podrán ser representados. Por ejemplo, en base $B = 10$, si disponemos de tres dígitos podremos codificar 1000 valores diferentes (desde el 000 hasta el 999); si disponemos de cinco dígitos podremos codificar 100.000 valores (desde el 00000 hasta el 99999).

Como se sabe, y como se desprende de esta forma de codificación, todo cero a la izquierda de estos dígitos supone un nuevo dígito que no aporta valor alguno a la cantidad codificada.

La expansión del número recoge el valor de cada dígito y su peso dentro de la cifra. El dígito a_0 del número a puede tomar cualquier valor comprendido entre 0 y $B-1$. Cuando se necesita codificar un número mayor o igual que el cardinal de la base (B) se requiere un segundo dígito a_1, que también puede tomar sucesivamente todos los valores comprendidos entre 0 y $B-1$. Cada vez que el dígito a_0 debiera superar el valor $B-1$ vuelve a tomar el valor inicial 0 y se incrementa en uno el dígito a_1. Cuando el dígito a_1 necesita superar el valor $B-1$ se hace necesario introducir un tercer dígito a_2 en la cifra, que también podrá tomar sucesivamente todos los valores comprendidos entre 0 y $B-1$,

incrementándose en uno cada vez que el dígito a_1 debiera superar el valor $B-1$. El dígito a_1 "contabiliza" el número de veces que a_0 alcanza en sus incrementos el valor superior a $B-1$. El dígito a_2 "contabiliza" el número de veces que a_1 alcanza en sus incrementos el valor superior a $B-1$. Por tanto, el incremento en uno del dígito a_1 supone B incrementos del dígito a_0. El incremento en uno del dígito a_2 supone B incrementos del dígito a_1, lo que a su vez supone B^2 incrementos de a_0. Y así, sucesivamente, el incremento del dígito a_j exige B^j incrementos en a_0.

Todos los dígitos posteriores a la posición j codifican el número de veces que el dígito j ha recorrido de forma completa todos los valores comprendidos entre 0 y $B-1$.

De todo lo dicho ahora se deduce que toda base es, en su sistema de numeración, base 10: Dos, en base binaria se codifica como 10; tres, en base 3, se codifica como 10; cuatro, en base 4, se codifica como 10…

Sistemas de numeración habituales en la informática.

El sistema de numeración más habitual en nuestro mundo es el sistema decimal. Si buscamos un porqué a nuestra base 10 quizá deduzcamos que su motivo descansa en el número de dedos de nuestras dos manos.

Pero un ordenador no tiene manos. Ni dedos.

Como hemos visto en el capítulo anterior, el circuito electrónico básico de la memoria de los ordenadores, tal como hoy se conciben, está formado por una gran cantidad de circuitos electrónicos que tiene dos estados estables posibles.

¿Cuántos estados posibles?...: DOS.

Por eso, porque los ordenadores "sólo tienen dos dedos", es por lo que ellos trabajan mejor en base dos. Es decir, sólo disponen de dos

elementos para codificar cantidades. El primer elemento, por definición de base, es el valor cero. El segundo (y último) es igual al cardinal de la base menos uno y es igual al primer elemento más uno. Esa base está formada, por tanto, por dos elementos que llamaremos: $B = \{0, 1\}$

Otras bases muy utilizadas en programación son la base octal (o base ocho) y la base hexadecimal (o base dieciséis).

Lo de la base hexadecimal puede llevar a una inicial confusión porque no nos imaginamos qué dígitos podemos emplear más allá del dígito nueve. Para esa base se extiende el conjunto de dígitos haciendo uso del abecedario: $B = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F\}$.

Sistema Binario.

Aprender a trabajar en una base nueva no está exento de cierta dificultad. Habría que echar mano de nuestra memoria, de cuando éramos infantes y no sabíamos contar. No nos resultaba sencillo saber qué número viene (en base diez) después del noventa y nueve.

Noventa y ocho,… Noventa y nueve,… Noventa y diez.

¡No!: cien.

Trabajemos en base diez:

0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29

y… ¿en base dos?: después de cero el uno. Y después del uno… ¡el diez!

0	1	10	11	100	101	110	111	1000	1001
1010	1011	1100	1101	1110	1111	10000	10001	10010	10011
10100	10101	10110	10111	11000	11001	11010	11011	11100	11101

En ambos cuadros están codificadas las mismas cantidades. En base

diez el primero, en base dos o base binaria el segundo.

Además de contar, es necesario aprender las operaciones matemáticas básicas. Al menos sumar y restar. De nuevo hay que volver a la infancia y aprender la aritmética básica de los clásicos cuadernos de sumas. Las reglas básicas para esas dos operaciones son:

```
0 + 0 = 0                  0 – 0 = 0

0 + 1 = 1                  0 – 1 = 1 "y debo 1"

1 + 0 = 1                  1 – 0 = 1

1 + 1 = 0 "y llevo 1"      1 – 1 = 0
```

Y así, se puede practicar con sumas de enteros de más o menos dígitos:

```
  10100      11101     1011011011      1010000110     100001001
  +1110      +10111      +1100011     +1100001110     +11101001
 100010     110100     1100111110     10110010100     111110010
```

Para las restas haremos lo mismo que cuando restamos en base diez: el minuendo siempre mayor que el sustrayendo: en caso contrario se invierten los valores y al resultado le cambiamos el signo:

```
  10100      11101     1011011011      1100001110     100001001
  -1110     -10111       -1100011     -1010000110     -11101001
  00110      00110     1001111000      0010001000     000100000
```

El mejor modo de aprender es tomar papel y bolígrafo y plantearse ejercicios hasta adquirir soltura y destreza suficiente para sentirse seguro en el manejo de estos números del en un sistema de numeración binario.

Al final del capítulo se recoge una sugerencia útil para practicar las operaciones aritméticas: realizarlas en la calculadora de Windows y probar luego a realizar esas mismas operaciones a mano.

Cambio de Base.

Las cantidades que se codifican no dependen del sistema de codificación. Antes hemos visto los treinta primeros números naturales (comenzando por el cero) codificados en base diez y en base dos. ¿Cómo obtener, a partir de una cantidad codificada en base dos, su codificación en base diez? ¿Y viceversa?

Paso de base dos a base diez: Para este cambio de base es suficiente con desarrollar la expansión del número. Por ejemplo:

$$(10011101)_2 = 1 \cdot 2^7 + 0 \cdot 2^6 + 0 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 =$$
$$= 128 + 16 + 8 + 4 + 1 = (157)_{10}.$$

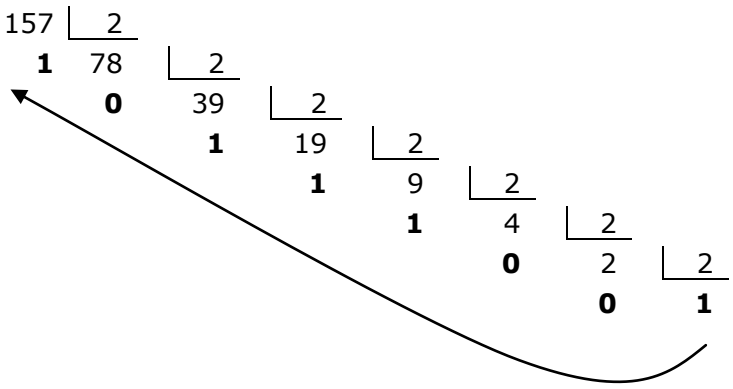
Paso de base diez a base dos: Para este cambio se divide el entero por dos (división entera), y se repite sucesivamente esta división hasta llegar a un cociente menor que la base. Simplemente vamos dividiendo por la base el número original y vamos repitiendo el procedimiento para los cocientes que vamos obteniendo.

Los restos de estas divisiones, y el último cociente, son los dígitos buscados (siempre deberán estar entre 0 y $B - 1$). El último cociente es el dígito más significativo, y el primer resto el menos significativo.

Por ejemplo, en el ejemplo recogido en el cuadro 2.1. vemos que el valor 157 expresado en base diez es, en base dos, 10011101.

Las bases octal y hexadeximal, a las que antes hemos hecho referencia, facilita el manejo de las cifras codificadas en base dos, que enseguida acumulan gran cantidad de dígitos, todos ellos ceros o unos.

Para pasar de base dos a base dieciséis es suficiente con separar la cifra binaria en bloques de cuatro en cuatro dígitos, comenzando por el dígito menos significativo. Al último bloque, si no tiene cuatro dígitos, se le añaden tantos ceros a la izquierda como sean necesarios.



Cuadro 2.1.: De base diez a base dos.

La equivalencia entre la base dos y la base dieciséis es inmediata sabiendo que dieciséis es dos a la cuarta. En la tabla 2.1. se recogen todas las equivalencias.

BINARIO	DEC	HEX	BINARIO	DEC	HEX
0000	0	0	1000	8	8
0001	1	1	1001	9	9
0010	2	2	1010	10	A
0011	3	3	1011	11	B
0100	4	4	1100	12	C
0101	5	5	1101	13	D
0110	6	6	1110	14	E
0111	7	7	1111	15	F

Tabla 2.1.: Equivalencias binario – hexadecimal

Por ejemplo, la cantidad 10011101 expresada en base dos, queda, en base diez, 157 y, en base hexadecimal, 9D: los cuatro últimos dígitos binarios son 1101 que equivale al dígito D hexadecimal. Y los otros cuatro dígitos binarios son 1001, que equivalen al 9 hexadecimal.

No es necesario, para cambiar de base decimal a base hexadecimal, hacer el paso intermedio por la base binaria. **Para pasar de cualquier base a la base decimal basta con la expansión del número**: ver expresión 2.1. Por ejemplo, el número 4E8, expresado en base

hexadecimal, sería, en base diez, el siguiente:

$(4E8)_{16} = 4 \cdot 16^2 + 14 \cdot 16^1 + 8 \cdot 16^0 = 4 \cdot 64 + 14 \cdot 16 + 8 \cdot 1 = (1256)_{10}$.

donde, como se ve, se cambian los valores de los dígitos mayores que nueve por su equivalente decimal.

El cambio a la base octal es muy semejante a todo lo que se ha visto para el cambio a la base hexadecimal. De la tabla 2.1. basta tener en consideración la columna de la izquierda. Los dígitos de la base octal son los mismos que para la base decimal, excluidos el 8 y el 9.

Complementos a la Base.

Vamos a introducir dos conceptos nuevos, muy sencillos. Los de complemento a la base y complemento a la base menos uno.

Supongamos el número N expresado en una base B determinada. Y supongamos que ese número tiene k dígitos. Llamamos **Complemento a la base** de ese número expresado en esa base determinada, a la cantidad que le falta para llegar a la cifra de $(k + 1)$ dígitos, en el que el dígito más significativo es el uno y los demás son todos ellos iguales a cero.

De una forma más precisa, definiremos el complemento a la base de un número N codificado con k cifras en base B como

$$C_B^k(N) = B^k - N. \qquad (2.4.)$$

Por ejemplo, en base diez, el complemento a la base del número $(279)_{10}$ es la cantidad que hace falta para llegar a $(1000)_{10}$, que es $(721)_{10}$.

En esta definición hay que destacar que, si el número N viene expresado de forma que a su izquierda se tienen algunos dígitos iguales a cero, entonces el complemento a la base es diferente que si estuviese sin esos dígitos, aunque la cantidad codificada sería la misma. Siguiendo con el ejemplo anterior, el complemento a la base del número codificado como

$(0279)_{10}$ ya no es $(721)_{10}$, sino $(9721)_{10}$, porque ahora no se trata de calcular lo que falta para llegar a $(1000)_{10}$, sino para llegar a $(10000)_{10}$, puesto que ahora la cantidad numérica está codificado con cuatro dígitos.

El concepto de **Complemento a la Base menos uno** es muy semejante: es la cantidad que dista entre el número N, codificado con k dígitos en la base B, y el número formado por k dígitos, todos ellos con el valor del mayor elemento de la base B en la que se trabaja.

De una forma más precisa, definiremos el complemento a la base menos uno de un número N codificado con k cifras en base B como

$$C_{B-1}^{k}(N) = B^k - N - 1. \qquad (2.5.)$$

Por ejemplo el complemento a la base menos uno del número $(541)_{10}$ expresado en base diez es la cantidad que hace falta para llegar a $(999)_{10}$, que es $(458)_{10}$.

De nuevo aquí también cambia el complemento a la base menos uno de un número si ponemos en su codificación más o menos ceros a su izquierda.

Es inmediato también deducir que la relación entre los dos complementos es que la diferencia entre ambos es igual a uno. De hecho se puede definir el Complemento a la Base menos uno de un número N codificado con k dígitos en la base B, como el Complemento a la Base de un número N codificado con k dígitos en la base B, menos 1.

$$C_{B-1}^{k}(N) = C_{B}^{k}(N) - 1. \qquad (2.6.)$$

Una curiosidad de los complementos es que, en cierta medida, facilitan el cálculo de las restas. Se pueden ver algunos ejemplos en base diez. Se cumple que la resta de dos enteros se puede también calcular haciendo la suma entre el minuendo y el complemento a la base del sustrayendo, despreciando el posible acarreo final. Por ejemplo:

619 — 492 127	El complemento a la base de 492 es 508	619 + 508 ~~1~~127

Donde si, como se ha dicho, despreciamos el último acarreo, tenemos que se llega al mismo resultado:127.

También se puede realizar un procedimiento similar si se trabaja con el complemento menos uno. En ese caso, lo que se hace con el último acarreo, si se tiene, es eliminarlo del resultado intermedio y sumarlo para llegar al resultado final. Por ejemplo:

619 — 492 127	El complemento a la base menos uno de 492 es 507 Sumamos el acarreo a la cantidad obtenida sin acarreo	619 + 507 1126 +1 127

Hasta ahora todo lo expuesto puede aparecer como un enredo algo inútil porque, de hecho, para el cálculo del complemento a la base es necesario realizar ya una resta, y no parece que sea mejor hacer la resta mediante uno de los complementos que hacerla directamente.

Pero las cosas cambian cuando se trabaja en base dos. Veamos algunos ejemplos de cálculo de los complementos en esa base:

N	$C_2(N)$	$C_1(N)$
10110	01010	01001
11001111001	00110000111	00110000110
101	011	010

Si se compara la codificación de cada número N con su correspondiente complemento a la base menos uno, se descubre que todos los dígitos están cambiados: allí donde en N corresponde el dígito 1, en $C_1(\mathrm{N})$ se

tiene un 0; y viceversa.

Es decir, que calcular el complemento a la base menos uno de cualquier número codificado en base dos es tan sencillo como cambiar el valor de cada uno de los dígitos de la cifra. Y esta operación es muy sencilla de realizar con circuitos electrónicos.

La ventaja clara que aportan los complementos es que no es necesario incorporar un restador en la ALU, puesto que con el sumador y un inversor se pueden realizar restas.

Recapitulación.

En este capítulo hemos visto que las cantidades pueden ser expresadas de muy variadas formas, de acuerdo con el sistema de codificación numérico que se quiera emplear. Y hemos estudiado las equivalencias entre los números codificados en diferentes bases.

Es conveniente que los conceptos introducidos en este capítulo estén bien trabajados: no tienen especial complicación, pero requiere un mínimo de esfuerzo y trabajo afianzar la noción de base. Es conveniente practicar diferentes cambios de base, y acostumbrarse a operar en base dos y en base hexadecimal.

También es conveniente practicar en la búsqueda de complementos: de hecho, como ya veremos en el próximo capítulo, el ordenador echa mano de los complementos a la base para almacenar la información. Vale la pena que se realicen diferentes cálculos y se verifique que realmente se ha llegado a resultados correctos.

Ejercicios.

- ***Cambio de base:*** Expresar $(810)_{10}$ en hexadecimal.

  Podemos emplear dos caminos: o pasarlo a base 2 (por divisiones

sucesivas por esa base) y posteriormente pasarlo a base hexadecimal; o pasar el valor directamente a base hexadecimal (dividiendo sucesivamente por 16). Tomamos esa última vía:

810	16	
10	50	16
	2	**3**

Y el resultado es $(810)_{10} = (32A)_{16}$: el dígito más significativo, el último cociente (aquel que ya es menor que el divisor, que es la base); y luego, uno detrás de otro, todos los restos, desde el último hasta el primer resto obtenido.

En la base hexadecimal existe un elemento de la base que codifica la cantidad 10: la letra 'A'. Así lo podemos ver en la tabla 2.1.

Podemos expresar ese número en cualquier otra base: por ejemplo, en base octal $(810)_{10} = (1452)_8$. Por último, como tercer ejemplo, lo pasamos a base 5 $(810)_{10} = (11220)_5$. Las divisiones realizadas para estas dos conversiones son las siguientes:

810	8		
2	101	8	
	5	12	8
		4	**1**

810	5			
0	162	5		
	2	32	5	
		2	6	5
			1	**1**

Que se puede verificar prontamente calculando la expansión (expresión 2.1.) del número en base 5, para pasarlo a la base decimal:

$$(11220)_5 = 1 \cdot 5^4 + 1 \cdot 5^3 + 2 \cdot 5^2 + 2 \cdot 5^1 + 0 \cdot 5^0 = 625 + 125 + 50 + 10 = (810)_{10}$$

Realizar sumas, restas, y productos en base binaria:

							1	1	1	1	0	0	1	1
							*	1	0	0	0	1	1	1
							1	1	1	1	0	0	1	1
						1	1	1	1	0	0	1	1	
					1	1	1	1	0	0	1	1		
+	1	1	1	1	0	0	1	1						
1	0	0	0	0	1	1	0	1	1	0	0	1	0	1

Que en base 10 resulta ser … 243 * 71 = 17253.

10010010010101100 + 10000111110011011 = 100011010001000111

que en base 10 resulta ser … 74924 + 69531 = 144455.

10010010011111110 – 10000110111101100 = 1011100010010

que en base 10 resulta ser … 75006 – 69100 = 5906.

- ***Verificar la resta anterior mediante la suma del minuendo y el complemento a la base menos 1 del sustrayendo.***

  Primero calculamos el complemento a la base menos uno de sustrayendo: $C_1\ (10000110111101100) = 01111001000010011$.

  Luego calculamos el complemento a la base sin más que sumar 1 al valor previo obtenido: $C_2\ (10000110111101100) = 01111001000010100$.

  Y ahora hacemos la suma del minuendo más este último valor

$$\begin{array}{r} 10010010011111110 \\ +01111001000010100 \\ \hline 100001011100010010 \end{array}$$

  Y despreciando el último acarreo, nos queda: 1011100010010, que el mismo que habíamos obtenido con la diferencia directa.

- ***Calcular los complementos a la base de los siguientes números (expresados en base 10):***

$C_2(0193) = 9807$	$C_1(0193) = 9806$
$C_2(00710) = 99290$	$C_1(00710) = 99289$
$C_2(6481) = 3519$	$C_1(6481) = 3518$
$C_2(009999) = 990001$	$C_1(009999) = 990001$
$C_2(98) = 2$	$C_1(98) = 1$

Sugerencia

Un modo cómodo y rápido de lograr muchos ejemplos de cambio de base y de codificación de enteros es utilizar la calculadora de Windows.

Figura 2.1.: Calculadora de Windows

Si en el menú Ver se selecciona la calculadora Científica, se obtiene una como la presentada en la figura 2.1. Debajo del visor, a su izquierda, se puede seleccionar la base en que se quieren introducir los números. Si se introduce un valor en base decimal (Opción "Dec"), luego se pueden ver esa misma cantidad codificada en cualquiera de las otras tres bases que ofrece esa calculadora: hexadecimal (opción "Hex"), octal (opción "Oct") ó binaria (opción "Bin"). Dependiendo de la base seleccionada, quedan activados o desactivados unos u otros de los botones de introducción de dígitos en la calculadora.

En la figura 2.2. puede verse la calculadora en la que se ha introducido un entero en base decimal, y luego, en la figura 2.3. puede verse ese mismo valor numérico codificado en base binaria.

Como se ve en la figura 2.3., cuando se trabaja en base hexadecimal, octal o binaria, en la parte derecha y debajo del visor, aparece un selector de cuatro posibilidades. Si se selecciona la más a la derecha, la que pone "*Byte*", entonces la calculadora trabajará con enteros de 8 bits. Si elegimos "*Word*", entonces trabajará con 16 bits (2 bytes). Con la siguiente opción, "*DWord*" llegamos a 32 bits. Y la última opción, "*QWord*" podremos trabajar incluso con 64 bits (enteros de 8 bytes). Con eso, podemos ver, por ejemplo, que el número $(8451)_{10}$ se expresa,

en base binaria, como $(10000100000011)_2$.

Figura 2.2.: A la calculadora le hemos introducido un valor en base decimal.

Para trabajar con mayor comodidad cuando se ven los números en base dos, se puede seleccionar, en el menú Ver de la calculadora, la opción "Número de dígitos en grupo". Así, todas las cantidades expresadas en binario vendrán agrupadas en dígitos de cuatro en cuatro, mostrando con facilidad la equivalencia entre esa expresión binaria y la misma utilizando dígitos hexadecimales.

Figura 2.3.: El mismo valor recogido en la figura 2.2., expresado ahora en base binaria.