Bloque 1

La información digital

Prácticas de
Fundamentos de Computadores
Curso 2009-2010

SESIÓN 1

Sistemas de codificación numéricos

Objetivos

Esta práctica tiene como objetivo aplicar los conceptos básicos de códigos numéricos. Estos conceptos se pondrán en práctica con el programa ICB, que puede ser muy útil para que el alumno compruebe si ha aprendido los conceptos correspondientes.

Conocimientos y materiales necesarios

Para el adecuado aprovechamiento de esta sesión de prácticas, el alumno necesita:

- Conocer los sistemas de codificación numéricos, tanto de números naturales y enteros (binario natural, hexadecimal, signo-magnitud, complemento a 2 y exceso a Z) como de números reales (IEEE-754).

Desarrollo de la práctica

1. Introducción al programa de códigos numéricos

Para practicar conceptos relacionados con códigos numéricos se va a utilizar el programa ICB (Intérprete de Códigos Binarios). Ejecútalo siguiendo las instrucciones que te indique tu profesor. El programa tiene varias secciones para practicar distintos conceptos. Vamos a explorarlas en los siguientes apartados.

2. Bases de representación

Escoge la opción de menú *Ver→Bases*. En la pantalla que aparece puedes escribir números naturales en base binaria, decimal, hexadecimal u octal, y ver cuál sería la representación en otra base.

- ❏ Calcula «a mano» cuál es la representación en binario, hexadecimal y octal del número `35`. Anota el resultado de la conversión en la siguiente tabla.

Decimal	Hexadecimal	Octal
35		

- ❏ A continuación escribe ese número en el campo reservado para números decimales y comprueba si has realizado bien la transformación de bases.
- ❏ Introduce el número `0.4` en el campo reservado para los números decimales. ¿Cuál es la representación en hexadecimal?[1] ¿Es exacta la representación obtenida en binario? ¿Y en hexadecimal?[2] Comprueba tu respuesta borrando la casilla del número hexadecimal y volviendo a introducir el mismo valor que tenía antes para que lo pase a decimal. ¿Qué valor decimal se obtiene?[3] Piensa sobre la causa de este error y, si no la entiendes, pregúntale a tu profesor.

1

2

3

3. Representación de enteros

Escoge la opción de menú *Ver→Enteros*. En la pantalla que aparece puedes escribir números enteros y ver su representación en decimal, en signo-magnitud, en complemento a 2 y en exceso a Z.

- ❏ Vamos a trabajar con 5 bits. Para ello, escribe el valor `5` en la caja de arriba a la izquierda. ¿Qué rango hay en signo-magnitud?[4] ¿Qué rango hay en complemento a 2?[5] ¿Qué rango hay en exceso central?[6] ¿Por qué se puede representar un número más en C2 y en exceso central que en signo-magnitud?
- ❏ Si quisiésemos representar números entre -31 y 0 en exceso a Z, ¿qué valor de Z habría que utilizar?[7] Comprueba tu respuesta seleccionando ese valor en la penúltima caja.
- ❏ Vuelve a seleccionar el exceso central (es decir, Z=16).
- ❏ Codifica «a mano» los números 10 y -10 en signo-magnitud, complemento a 2 y exceso a Z y comprueba que los has codificado bien con el programa.
- ❏ Piensa qué secuencia binaria habría que introducir en la caja de exceso a Z para que la secuencia que aparezca en complemento a 2 sea 11001. Comprueba tu respuesta con el programa. ¿Cuál era la secuencia?[8].

4

5

6

7

8

4. Comparación de formatos

Escoge la opción de menú *Ver→Comparación de formatos*. Esta pantalla permite comprobar cómo una secuencia de bits tiene distintos significados según con qué formato la interpretemos.

- ❏ Deduce «a mano» con qué valor decimal se corresponde la secuencia `10111` si la interpretamos como binario natural, como signo-magnitud, como complemento a 2, como exceso a 9 y como un formato de coma fija que tiene dos bits de parte fraccionaria. Anota los resultados en la siguiente tabla.

Secuencia	B. natural	Sig-Mag	C. a 2	Ex-9	C. fija (2frac)
10111					

❏ Comprueba con el programa que has realizado bien todas las transformaciones.

5. Coma fija

Escoge la opción de menú *Ver→Coma fija*. Esta pantalla permite practicar conceptos relacionados con la representación de números reales utilizando coma fija. En la parte superior se puede escoger el número total de bits del formato. La posición del primer bit de la parte fraccionaria se puede escoger mediante una barra deslizante etiquetada como *Posición punto decimal*. Bajo este selector hay dos cajas de edición para introducir valores binarios o decimales. La caja activa depende del selector de modo situado sobre ellas.

❏ Escoge en la parte superior de la pantalla 10 bits como número de bits total. Asegúrate de que no está marcada la casilla *Complemento a 2*. Moviendo la barra deslizante, escoge que la posición del punto decimal valga 3.

❏ Calcula «a mano» cuál es la representación en este formato del número decimal `12.5`. Comprueba que lo has hecho bien con el programa.

❏ Si se marcase la casilla de *Complemento a 2*, ¿qué ocurriría con la precisión?[9] Comprueba tu respuesta activándola.

[9]

❏ Manteniendo la casilla de *Complemento a 2* activada y la secuencia de bits que te salió en los pasos anteriores, escoge el modo *Binario→Decimal* y observa lo que ocurre cuando mueves la posición del punto decimal. ¿Qué ocurre con el número decimal representado cuando lo mueves el punto decimal a la derecha?[10] Observa cómo cambian también el rango y la precisión.

[10]

6. Coma flotante

Escoge la opción de menú *Ver→Coma flotante*. En esta sección el programa permite practicar con diversos formato de coma flotante. Inicialmente hay definidos dos: el formato YEE, que es un formato de ejemplo que no se utiliza en ninguna máquina real, y el formato IEEE-754 simple[1].

❏ Escoge la opción de menú *Flotantes→IEEE*.

❏ Calcula «a mano» cuál es la representación del valor `-34.5` en este formato. Introduce ese valor en el campo reservado para los números decimales y comprueba que su representación se corresponde con la que habías calculado.

❏ Calcula «a mano» cuál es el valor en decimal del número cuya representación en este formato escrita en hexadecimal es `3FA00000h`. Introduce el valor (en binario) en la caja dedicada a números binarios y comprueba que tu respuesta fue correcta.

[1] El programa no tiene en cuenta los números desnormalizados.

Ahora vamos a crear un nuevo formato:

- ❏ Escoge la opción de menú *Archivo→Añadir/Modificar formatos...*
- ❏ Pulsa el botón *Añadir* y, en la pantalla que aparecerá, dale al formato el nombre «Prueba» y pulsa *OK*.
- ❏ En la pantalla que te aparece, escoge 10 bits como número de bits total. Selecciona las opciones inverso y bit implícito. Escoge 4 bits para el exponente y que el exponente esté en exceso a Z con Z igual a 8 y normalización todo fracción. Cuando estén todas las opciones adecuadamente seleccionadas pulsa el botón *Aceptar*.
- ❏ Vete a la opción de menú *Flotantes→Prueba*.
- ❏ Introduce el valor `1.25` en la caja reservada a números decimales. ¿Cuál es la representación de este número en el formato Prueba?[11] (Responde en hexadecimal.)
- ❏ ¿Qué valor decimal se corresponde con la representación formada por todo unos?[12]

11

12

El programa permite mostrar cómo se genera paso a paso una representación de un número en coma flotante en uno de los formatos que hayamos definido nosotros.

- ❏ Escoge la opción de menú *Ver→Traza coma flotante*.
- ❏ Pulsa con el botón derecho del ratón sobre la zona negra que aparecerá en el medio de la ventana y escoge la opción *Nuevo número*.
- ❏ En la ventana que aparecerá, introduce el número `1.25` y escoge como formato «Prueba».
- ❏ Al pulsar *OK* te aparecerá en la ventana un esquema que indica cómo se forma la representación. En la primera línea se muestra el paso del número de decimal a binario. En la segunda línea se muestra la operación que hay que hacer para normalizar la mantisa. A continuación tienes cómo se obtiene el signo, el exponente y la mantisa.

7. Ejercicios adicionales

- ✏ En un formato de codificación en coma fija con 3 bits para la parte entera y 2 bits para la parte fraccionaria se codifica el número 3.4. Determinar, ayudado por el programa de codificación, el error cometido en la representación.
- ✏ Interpretar la secuencia 11001 en formato complemento a dos, binario y exceso a 4. Verifica los resultados con el programa de codificación.
- ✏ Codificar el número -256.25 en formato IEEE-754. Verifica los resultados con el programa de codificación.
- ✏ Busca dos números de seis bits cuya suma genere carry y overflow simultáneamente. Verifica los resultados con el programa de codificación.

SESIÓN 2

Sistemas de codificación de caracteres

Objetivos

Esta práctica tiene como objetivo aplicar los conceptos básicos sobre codificación de caracteres. Estos conceptos se pondrán en práctica con diversos ejemplos.

Conocimientos y materiales necesarios

Para el adecuado aprovechamiento de esta sesión de prácticas, el alumno necesita:

- Conocer los sistemas de codificación alfanuméricos (*ASCII*, *ISO Latin-1* y *UTF-8*).

Desarrollo de la práctica

1. Introducción

Para practicar conceptos relacionados con la codificación de caracteres se va a utilizar un editor hexadecimal, en concreto el programa denominado *HxD*, que es gratuito y se puede descargar de Internet.

2. Codificación de caracteres en distintos formatos

Los caracteres, al igual que el resto de información que se trata en formato digital, deben ser codificados mediante secuencias de ceros y unos. Esta labor es exactamente la que realiza un editor de texto cuando almacena la información que el usuario ha escrito. Cada una de las teclas (o combinación de teclas) pulsadas se traducen a un carácter o una orden, para posteriormente codificar esta información y almacenarla en disco. La traducción de tecla pulsada a carácter no es trivial ya que, dependiendo de la configuración del teclado, la misma tecla puede representar distintos caracteres. Por ejemplo, en un teclado americano no hay 'ñ': la tecla que está a la derecha de la 'l' se usa para el carácter ';'.

Cuando se abre un fichero de texto, el editor decodifica la secuencia de ceros y unos para determinar qué caracteres se encuentran almacenados, y de esta forma mostrar la información tal y como el usuario la escribió. El proceso de decodificación puede ser problemático, ya que el editor debe determinar qué método de codificación de caracteres se usó para almacenar el fichero de texto. Para llevar a cabo el proceso, el *Bloc de notas*, editor básico de Windows, intenta *adivinar* la codificación en función del contenido, acertando en algunas ocasiones y fallando en otras. Otros editores simplemente decodifican la información utilizando un formato fijo.

A continuación, se va a proceder a crear un fichero de texto y a ver la forma en la que dicho fichero está codificado.

- ❏ Abre el *Bloc de notas* y escribe el texto «El murciélago grande voló alto». Guarda el fichero en el disco y observa el método de codificación que se está empleando (aparece en la parte inferior del diálogo de guardar el fichero). Por defecto aparece seleccionado el formato *ANSI*, que, en una versión de Windows con idioma en castellano, se refiere al formato de codificación de caracteres *Windows-1252*. Este formato es muy similar al *ISO Latin-1* y es el formato que se usa por defecto en una versión de Windows en castellano. En Windows a cada formato de codificación se le denomina *tabla de códigos* (*code page*). El formato *Windows-1252* tiene asignada la tabla de códigos número 1252.
- ❏ A continuación, abre con el editor hexadecimal *HxD* el fichero que has creado con el *Bloc de notas*. Un editor hexadecimal muestra la información almacenada en el fichero en formato crudo, es decir, como secuencias de ceros y unos que, por simplicidad, aparecen en formato hexadecimal. Además, generalmente hace una interpretación de la información en algún formato de codificación de caracteres. El *HxD* decodifica la información suponiendo siempre que ha sido codificada en formato *ANSI*, es decir, en formato *Windows-1252*. En este caso, el fichero se almacenó utilizando ese formato, con lo que la información decodificada es correcta como se puede observar.
- ❏ Utilizando el editor hexadecimal fíjate cómo se codifican los caracteres. ¿Cuál es la codificación del carácter 'E'?[1] ¿Cuál es la codificación del carácter 'e'?[2] ¿Cuál es la codificación del carácter 'é'?[3]
- ❏ El formato *Windows-1252* codifica los primeros 128 caracteres utilizando el formato ASCII. ¿Cuál es el peso del bit que cambia en la codificación de mayúsculas y minúsculas en formato ASCII?[4] Verifica la respuesta a partir de la codificación de la 'E' y de la 'e'.
- ❏ Utilizando el editor hexadecimal y lo que has aprendido anteriormente, cambia la codificación del primer carácter de cada una de las palabras de la frase para pasarlos a mayúsculas.
- ❏ Guarda los cambios que has hecho, cierra el editor hexadecimal (si no lo haces, más tarde tendrás problemas) y abre de nuevo el fichero con el *Bloc de notas*.

1

2

3

4

A continuación, se va a ver qué ocurre cuando se guarda el fichero utilizando otro formato de codificación.

- ❏ Dentro del *Bloc de notas*, vete al menú *Archivo* y haz clic en *Guardar como...*. A continuación, en la parte inferior de diálogo, selecciona el formato *UTF-8*, pulsa *Guardar* y sobrescribe el fichero.

- ❏ Antes de continuar, y sabiendo la codificación del carácter 'é' (respondida anteriormente), determina cuál será la codificación de dicho carácter en formato *UTF-8*[5]. Repasa la teoría si no recuerdas cómo se codifica en *UTF-8*. Además, recuerda que el código Unicode de los primeros 256 caracteres coincide con el formato *ISO Latin-1*, y que este formato es prácticamente igual que el formato *Windows-1252*, coincidiendo en la codificación de la mayoría de caracteres, incluyendo el carácter 'é'.

5

- ❏ Abre el fichero con el programa *HxD* y verifica la respuesta anterior.
- ❏ Como se puede comprobar, la interpretación que hace el *HxD* de la secuencia de ceros y unos contenida en el fichero es incorrecta, ya que decodifica suponiendo el formato *Windows-1252* y en realidad los caracteres están codificados en formato *UTF-8*. Este fallo de interpretación sólo afecta a los caracteres acentuados, ya que el resto se codifican igual en *Windows-1252* y en *UTF-8*. Además, se pueden apreciar unos caracteres especiales (`EF BBh`) al comienzo del fichero. Estos caracteres son el *BOM* (Byte Order Mark) que se utilizan para indicar que el fichero está codificado en *UTF-8*. Para codificar un fichero de texto en formato *UTF-8* no es obligatorio incluir el *BOM*, aunque muchos programas lo añaden.
- ❏ Cierra el *HxD* y abre de nuevo el fichero con el *Bloc de notas*. Guarda de nuevo el fichero, pero en este caso, selecciona el formato Unicode, que en realidad se refiere al formato *UTF-16*. En *UTF-16* se codifica cada carácter con un número variable de bytes: dos o cuatro. Si el carácter a codificar tiene asignado un número Unicode menor o igual de 65535 se codifica con dos bytes, en caso contrario se codifica con cuatro bytes. Para obtener más información, se puede buscar por Internet.
- ❏ Abre de nuevo el fichero con el programa *HxD* y observa cómo se han codificado los caracteres en esta ocasión. Cada carácter se ha codificado utilizando dos bytes, por tanto el fichero ocupa el doble. Por ejemplo, la codificación del carácter 'o', que en *Windows-1252* era `6Fh`, ahora resulta ser `00 6Fh`. Debido al formato de ordenación utilizado[1] aparece como `6F 00h`.

A continuación, se va a analizar lo que ocurre cuando se trabaja con ficheros desde el símbolo del sistema.

- ❏ Cierra el *HxD* y abre de nuevo el fichero sobre el que se ha estado trabajando con el *Bloc de notas* y guárdalo en formato ANSI.
- ❏ Abre el fichero con el *HxD* y verifica que la codificación es la que cabría esperar. Después cierra el *HxD*.
- ❏ Abre el símbolo del sistema (*Inicio→Ejecutar*, escribe *cmd* y pulsa *Aceptar*).
- ❏ Cambia al directorio (carpeta) donde tengas guardado el fichero sobre el que se ha estado trabajando. Para cambiar de directorio utiliza el comando *cd* (ejemplo de uso: *cd directorio*, donde *directorio* es el nombre de la carpeta a la que quieres acceder), y para listar el contenido de un directorio utiliza el comando *dir*.
- ❏ Cuando hayas llegado al directorio donde se encuentra el fichero muestra el contenido con el comando *type* (ejemplo de uso: *type fichero.txt*). ¿Se ve correctamente el contenido?[6]. Ábrelo de nuevo con el *Bloc de notas* y verifica si el contenido es correcto.

6

[1] El criterio *Little Endian* de ordenación de bytes se estudiará en el tema de la arquitectura Intel.

- ❏ El problema es que desde el símbolo del sistema se decodifica el contenido del fichero suponiendo que está codificado en un formato distinto al *Windows-1252*, con el que se codificó en realidad. Esta discrepancia es debida a que los programas que se ejecutan desde el símbolo del sistema decodifican los caracteres suponiendo un formato de codificación diferente al *Windows-1252*. La razones para esta discrepancia son históricas: por mantener la compatibilidad con los programas de MS-DOS. En este sistema operativo, anterior a Windows y, por tanto, anterior al formato *Windows-1252*, se utilizaba el ASCII extendido, y este formato de codificación es el que se sigue usando hoy en los programas que se ejecutan desde el símbolo del sistema, como por ejemplo el programa *type*.
- ❏ Como puedes observar el problema está en la decodificación de los caracteres acentuados. Esto es debido a que dichoscaracteres no pertenecen al formato ASCII, y el *Windows-1252* y el ASCII extendido los codifican de distinta forma. Abre la siguiente página Web donde se muestra el formato *Windows-1252*:

  `http://www.microsoft.com/globaldev/reference/sbcs/1252.mspx`

- ❏ Fíjate en la codificación de cada uno de los caracteres. El código que el formato asigna al carácter se encuentra en el número de fila y de columna. El número que aparece bajo el carácter es su número Unicode. ¿Cuál es la codificación de caracter 'ó'?[7].
- ❏ Ahora, abre la siguiente página web donde se muestra el formato de codificación *OEM-850* utilizado por el símbolo del sistema cuando el sistema operativo está instalado en idioma castellano:

  `http://www.microsoft.com/globaldev/reference/oem/850.mspx`

- ❏ ¿Cuál es el carácter en formato *OEM-850* que se corresponde con el código que tenía asignado el carácter 'ó' en *Windows-1252*?[8]
- ❏ No cierres aún el símbolo del sistema.

[7]

[8]

Con este pequeño ejercicio deberías comprender por qué cuando has ejecutado el comando *type* se mostraron esas discrepancias con el texto escrito.

Es posible cambiar el formato de codificación utilizado en el símbolo del sistema para poder trabajar con ficheros de texto codificados en formato *Windows-1252*. Para ello sigue los siguientes pasos:

- ❏ Ejecuta el comando *chcp*. Esto te dirá el número de tabla de códigos activa.
- ❏ Cambia la tabla de códigos activa a la 1252 (*chcp 1252*).
- ❏ Pulsa con el botón derecho sobre la barra del título del símbolo del sistema y abre las propiedades.
- ❏ En el apartado del tipo de fuente, selecciona la fuente *Lucida Console* y acepta. Cuando te pregunte, escoge aplicar los cambios sólo a la ventana actual.
- ❏ Vuelve a mostrar el contenido del fichero con *type*. Si has realizado todos los pasos correctamente el contenido se debería de mostrar tal y como lo escribiste.

- ❏ Ahora ejecuta desde el símbolo del sistema el programa *edit*, ¿podrías explicar porqué los bordes se ven mal? (Para comprobar cómo se verían de forma correcta, abre una nueva ventana del símbolo del sistema (*Inicio→Ejecutar*, escribe *cmd* y pulsa *Aceptar*) y ejecuta el comando *edit*).
- ❏ Cambia de nuevo el número de tabla de página al original, 850, y vuelve a ejecutar el *edit* para ver los bordes bien.

Para terminar esta parte de la sesión de prácticas se proporcionan tres ficheros de texto: *prueba1.txt*, *prueba2.txt* y *prueba3.txt*. En los tres fichero se ha escrito la misma frase: «Mi mamá me mima». Aplicando los conocimientos adquiridos y usando las herramientas proporcionadas, determinar cuál fue el formato de codificación de caracteres utilizado para codificar la información almacenada en *prueba1.txt*[9], *prueba2.txt*[10] y *prueba3.txt*[11].

9

10

11

3. Codificación de caracteres y páginas Web

A continuación, se va a ilustrar la codificación de caracteres mediante otro ejemplo: se va a hacer una página web mínima:

- ❏ Abre el *edit* y escribe el código siguiente:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
        <head>
                <title>
                        Este es el título
                </title>
        </head>
        <body>
                Esta es mi primera página web.
        </body>
</html>
```

- ❏ Escoge la opción de menú *Archivo→Guardar como...* y guárdalo en tu directorio de trabajo con el nombre $prueba.html$.
- ❏ Abre el navegador web *Firefox*, escoge la opción *Archivo→Abrir archivo...* y carga el archivo que has creado. ¿Se ve correctamente?[12]
- ❏ Abre ahora el navegador web *Internet Explorer*, escoge la opción *Archivo→Abrir...* y carga el archivo creado. ¿Se ve correctamente?[13]

12

13

Como podrás observar, los caracteres no acentuados, que compartimos con el inglés, se ven bien en cualquier caso; la razón es que casi todos los sistemas de codificación los codifican con el mismo código. Sin embargo, en los caracteres acentuados es más habitual que haya problemas porque distintos sistemas codificación representan el mismo carácter de distinta forma o asignan al mismo código distinto carácter. Cualquier programa, incluyendo los visualizadores de páginas web como el *Firefox* y el *Internet Explorer*, tienen que interpretar los ficheros que leen suponiendo que siguen un sistema de codificación. Si no saben en qué sistema está codificado, intentan adivinarlo, pero no siempre aciertan. Para evitar que tengan que adivinarlo, lo mejor es indicar en el propio fichero cual es la codificación utilizada.

- ❏ En el *Explorer*, selecciona la opción de menú *Ver→Codificación*. ¿Qué codificación está marcada?[14]

14

- ❏ En el *Firefox*, selecciona la opción de menú *Ver→Codificación*. ¿Qué codificación está marcada?[15]
- ❏ Añade al fichero, debajo de la línea que contiene «<head>», la siguiente línea:

  ```
  <meta http-equiv="Content-Type" content="text/html; charset=cp850"/>
  ```

  Esta línea le dice al navegador que el contenido está con el juego de caracteres *OEM-850*.
- ❏ Guarda el fichero modificado.
- ❏ Recarga el fichero en los dos navegadores para comprobar que ahora se ve bien.

15

¿Pero quién decide qué codificación se utiliza? En el caso de editar la página web con el *edit*, la codificación la decide el propio programa: *OEM-850*. En el caso de utilizar el *Bloc de notas* la codificación la decide el usuario, el creador del archivo, cuando se graba, como vas a comprobar ahora:

- ❏ Abrir el *Bloc de notas* y cargar el fichero *prueba.html*. Se puede comprobar que el contenido del fichero ha sido interpretado utilizando la codificación *Windows-1252*, por lo que los caracteres acentuados se mostrarán de forma incorrecta. Para solucionar este problema corrige los caracteres acentuados y modifica la opción *charset* de la segunda línea del fichero para que sea de la forma:

  ```
  charset=Windows-1252
  ```
- ❏ En el *Bloc de notas*, escoge la opción *Archivo→Guardar como...*. Si te fijas, en la parte inferior hay una lista de selección que permite escoger la codificación y por defecto el valor escogido es ANSI. Este es el nombre que muestra el *Bloc de notas* para la codificación *Windows-1252*.

Fíjate que debe coincidir la codificación con la que grabas y la codificación que indicas en el fichero. Hagamos la prueba de hacer que ambas no coincidan para ver lo que ocurre:

- ❏ En el cuadro de diálogo de *Guardar como...* escoge la codificación *UTF-8* y guarda el archivo.
- ❏ Recarga el archivo en los dos navegadores. ¿Se ve bien en alguno?

Como habrás comprobado, en el *Explorer* se sigue viendo bien. Si examinas la codificación con la que lo está interpretando (menú *Ver→Codificación*), verás que, a pesar de que se le dice en el archivo que utilice la codificación *Windows-1252* utiliza *UTF-8* y además es imposible cambiarlo. El *Explorer* está identificando que es *UTF-8*, basándose en que los primeros bytes se corresponden con códigos *UTF*, a pesar de que en el código *HTML* se indica lo contrario. El *Explorer* se intenta proteger contra este tipo de errores y, si encuentra que no tiene sentido lo que lee si lo interpreta con la codificación incorrecta, intenta adivinar cuál es la correcta. Esto a veces es bueno —evita algunos errores del programador— pero también puede ser malo y en los ejercicios adicionales se muestra con el *Bloc de notas* un ejemplo de cuándo esto funciona mal.

Como último ejemplo, vamos a examinar una página en japonés.

❏ Vete, a través del *Firefox*, a la dirección

`http://www.google.co.jp/`

Esta es la versión de *Google* en japonés. ¿Estará utilizando como codificación *Windows-1252* o alguna similar al *ISO Latin-1*? No puede hacerlo porque tiene que mostrar caracteres en japonés y en el alfabeto latino. Por eso tiene que utilizar alguna codificación de Unicode.

❏ Escoge la opción de menú *Ver→Codificación*. ¿Qué codificación se está utilizando? [16] Si sabes japonés, verás que se están mostrando el texto adecuado.

16

❏ Vamos a cambiar la codificación en el navegador para que interprete los mismos bytes que recibe con otro formato. Selecciona en el menú *Ver→Codificación→Windows-1252*. El resultado son un conjunto de caracteres extraños (aes con diéresis, virgulillas, etc.) que, en cualquier caso, no tienen aspecto de japonés. Se puede cambiar la codificación a otros idiomas para ver otros tipos de caracteres. Por ejemplo, seleccionando *Ver→Codificación→Más códigos→Asia del Este→Chino simplificado* puedes ver caracteres chinos.

Como ves, el mundo de la codificación de caracteres es muy complejo y fundamental: si no se sabe con qué está codificado un archivo, es imposible leerlo y entenderlo. La idea principal es que una secuencia de ceros y unos no significa nada a no ser que se interpreten con un determinado formato de codificación. Esto, por supuesto, es aplicable a la codificación de cualquier tipo de información, por eso, por ejemplo, hay que saber con qué está codificado una secuencia de vídeo digital para poder visualizarla correctamente.

4. Ejercicios adicionales

✏ Abre el *Bloc de notas* y escribe: «this app can break» (sin las comillas). Guarda el documento en disco, cierra el *Bloc de notas* y abre de nuevo con el *Bloc de notas* el documento que acabas de guardar... Probablemente no te esperes el resultado. Utiliza tus conocimientos de *Google* para encontrar la explicación a este fenómeno.

✏ Abre el *Bloc de notas* y escribe de nuevo el texto «El murciélago grande voló alto». Guarda el fichero en el disco usando la codificación ANSI. Sabiendo que el formato de codificación que utiliza Windows en idioma griego es *Windows-1253*, ¿sabrías decir que va a leer un griego si abriese tu fichero en su ordenador? El formato *Windows-1253* se encuentra en la siguiente página:

`http://www.microsoft.com/globaldev/reference/sbcs/1253.mspx`