TELECO.
CAMPUS SUR·UPM
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA Y SISTEMAS DE TELECOMUNICACIÓN

POLITÉCNICA

dte POLITÉCNICA
Departamento de Ingeniería Telemática y Electrónica

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA Y SISTEMAS DE TELECOMUNICACIÓN

PROGRAMACIÓN I

Configuración UTF-8 en Windows

Curso 2024/25

ÍNDICE

1.- Introducción

El ordenador utiliza códigos numéricos para identificar los distintos caracteres de los diferentes idiomas. A la identificación de cada carácter con su código numérico correspondiente se le denomina *codificación de caracteres* o *tabla de códigos.*

Históricamente, los ordenadores y otros medios de trasmisión electrónica sólo utilizaban los caracteres del alfabeto inglés y era suficiente un número entre 0 y 127 para identificar cualquiera de ellos. Un número generado con 1 byte era por tanto adecuado para diferenciar los caracteres que había que mostrar en los terminales o que había que escribir en los ficheros de texto. Se utilizaba el sistema de codificación ASCII (*American Standard Code for Information Interchange*), desarrollado en EEUU para las trasmisiones telegráficas.

La necesidad de utilizar los caracteres de otros idiomas, por ejemplo la letra ñ o las letras acentuadas en el idioma español, hizo necesario utilizar números de más de 1 byte, con los que poder identificar más de 256 valores.

Actualmente, el estándar más utilizado para codificación de caracteres es el sistema llamado Unicode, en particular el sistema de codificación UTF-8 (*Unicode Transformation Format – 8-bit*). Se trata de un sistema de codificación que utiliza entre 1 y 4 bytes para codificar cada carácter. Permite representar más de un millón de símbolos, que incluyen los caracteres de todos los idiomas y numerosos símbolos gráficos de todo tipo.

Lamentablemente, Windows no utiliza UTF-8 por defecto en su terminal *Command Prompt* y, en muchas ocasiones, tampoco en los ficheros de texto que genera. En el terminal, utiliza lo que denomina "*página de códigos 850 MS-DOS Latin 1*". Esto es un inconveniente cuando se está programando en español, pues no son compatibles los códigos de la letra ñ, de los caracteres acentuados y otros caracteres especiales.

En los ficheros de texto, por ejemplo el código de los programas que escribamos en C, pasa algo parecido. En muchas ocasiones Windows utiliza sus propios sistemas de codificación de caracteres y puede suceder que no visualicemos bien algunos caracteres.

En este documento se va a explicar cómo configurar Windows para que utilice en el terminal la página de códigos 65001, que se corresponde con la codificación UTF-8, permitiendo así que nuestros programas escritos en C puedan utilizar en el terminal todos los caracteres del alfabeto español o de otros idiomas.

También aprenderemos a configurar VSCode para que trabaje con otras tablas de códigos distintas de UTF-8, con lo que podríamos hacer que la codificación de los programas se adaptara a la consola de Windows (CP-850), en lugar de la consola a los programas, como se indicó en el párrafo anterior

Por último, explicaremos cómo utilizar las herramientas proporcionadas por VSCode para convertir ficheros de unas tablas de códigos a otras, sin necesidad de modificar manualmente el texto.

2.- Configuración del terminal

En este apartado vamos a explicar varias formas de configurar el terminal de Windows *Command Prompt (Símbolo del Sistema)* para que pueda mostrar caracteres con codificación UTF-8, de forma que la tabla de códigos de la consola y la de VSCode estén en coherencia.

Hay que distinguir dos situaciones diferentes que requieren distintas soluciones:

- **Configurar nuestro terminal**: cuando los programas que escribimos los vamos a ejecutar en nuestro ordenador o en un ordenador al que tenemos acceso, podemos actuar configurando directamente el terminal o el propio sistema operativo Windows, de forma que utilice la codificación UTF-8.

- **Programas que se van a ejecutar en otros ordenadores**: cuando los programas que hacemos se van a ejecutar en otro ordenador al que no tenemos acceso, no podemos actuar directamente sobre Windows. En esos casos, tiene que ser el propio programa el que dé las órdenes al sistema operativo para que muestre los caracteres correctamente.

Para probar los resultados en cada caso, vamos a utilizar una modificación del típico programa "*Hola, mundo*", en la que se sustituye la frase clásica en inglés "*Hello, World!*", por otra que incluye algunos caracteres españoles: "*¡Hola, mundo, qué año llevamos!*". El código podría ser el siguiente:

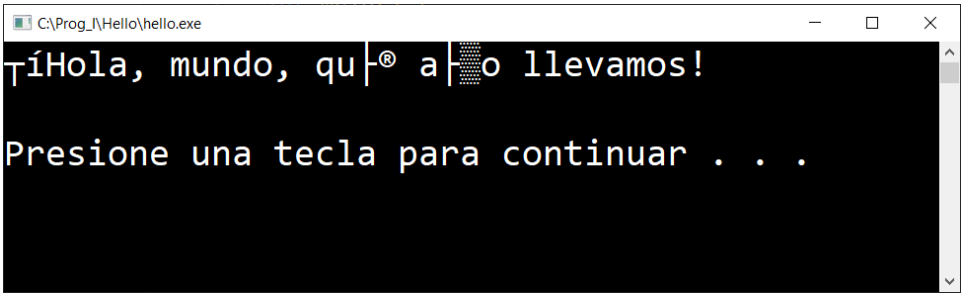
```
#include <stdio.h>
#include <windows.h>

int main() {
   printf("¡Hola, mundo, qué año llevamos!\n\n");

   system("pause");
   return 0;
}
```

En el código anterior, observa que se intenta imprimir en el terminal un mensaje que contiene varios caracteres específicos que no pertenecen al alfabeto inglés: el signo de apertura de exclamación, la letra "e" acentuada y la letra "ñ". Recuerda que los dos caracteres "`\n`" que aparecen al final de la frase entrecomillada indican al ordenador que haga un cambio de línea. En este caso se hacen dos cambios de línea tras escribir la frase.

Si ejecutamos este programa en el terminal, con la configuración por defecto de Windows, el resultado mostrará los caracteres especiales de manera incorrecta, tal y como visualiza la siguiente figura:



Observa como en los lugares donde deberían aparecer los caracteres especiales, se insertan dos símbolos gráficos sin ninguna lógica aparente.

En la figura anterior se ha utilizado el terminal *Command Prompt (Símbolo del sistema)* de Windows, el que en VSCode se denomina *terminal externo*, pero habríamos obtenido el mismo resultado si hubiéramos utilizado el terminal integrado de VSCode, pues no deja de ser el mismo terminal *Command Prompt* que proporciona Windows, pero ejecutado dentro del editor.

OBSERVACIÓN:

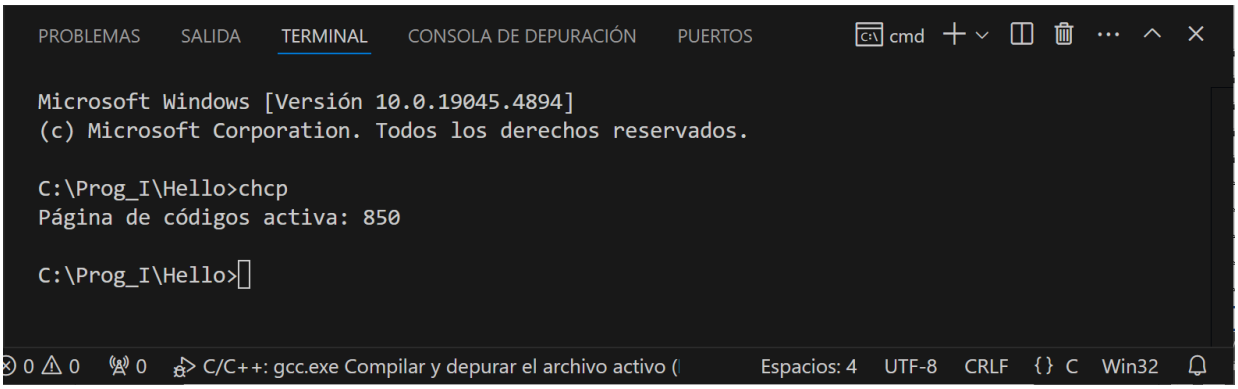
Aunque los apartados siguientes muestran varios procedimientos para llevar a cabo la personalización, la solución más eficiente es la Configuración global de Windows que se explica en el apartado 2.4. Si tienes prisa, vete directamente a ese apartado.

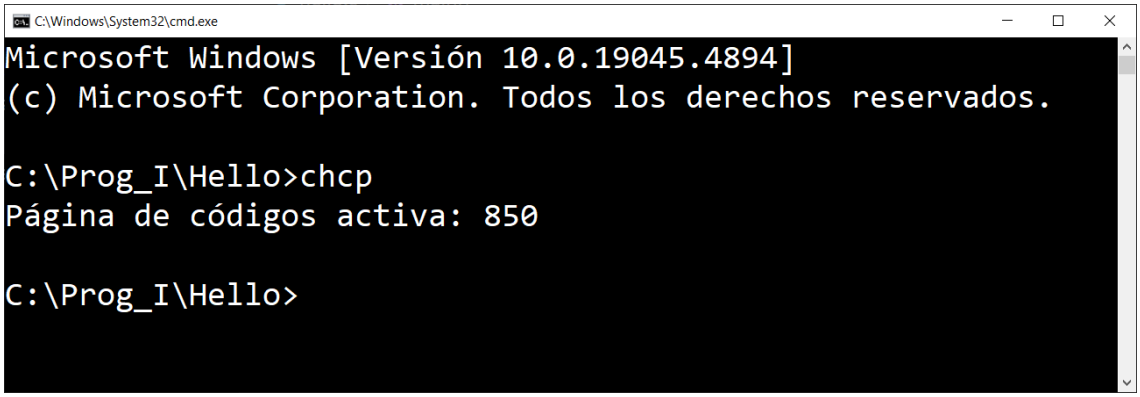
No obstante, los otros métodos tienen valor didáctico para comprender en profundidad aspectos importantes de nuestro ordenador, por lo que te recomendamos probarlos.

2.1.- Instrucción *CHange Code Page* (chcp)

El terminal de Windows ofrece la instrucción *chcp (CHange Code Page)* que permite visualizar la página de códigos activa o seleccionar una página de códigos concreta para el terminal.

Para ver qué página de códigos hay activa en el terminal, hay que teclear la instrucción *chcp*, sin argumentos adicionales. Las figuras siguientes muestran la página de codificación por defecto que utiliza Windows en el terminal integrado de VSCode y en el terminal externo:





NOTA: El programa *cmd.exe*

Observa que, en las dos figuras anteriores, el arranque del terminal es idéntico en VSCode y en el terminal de Windows. En realidad, es el mismo programa del sistema operativo, el programa *cmd.exe*, que puedes encontrar en el directorio `C:\Windows\System32`.

En la segunda figura, que corresponde a la ejecución del terminal directamente desde Windows, comprueba como en la barra del título de la ventana del terminal se visualiza la ruta y el nombre del programa: `C:\Windows\System32\cmd.exe`.

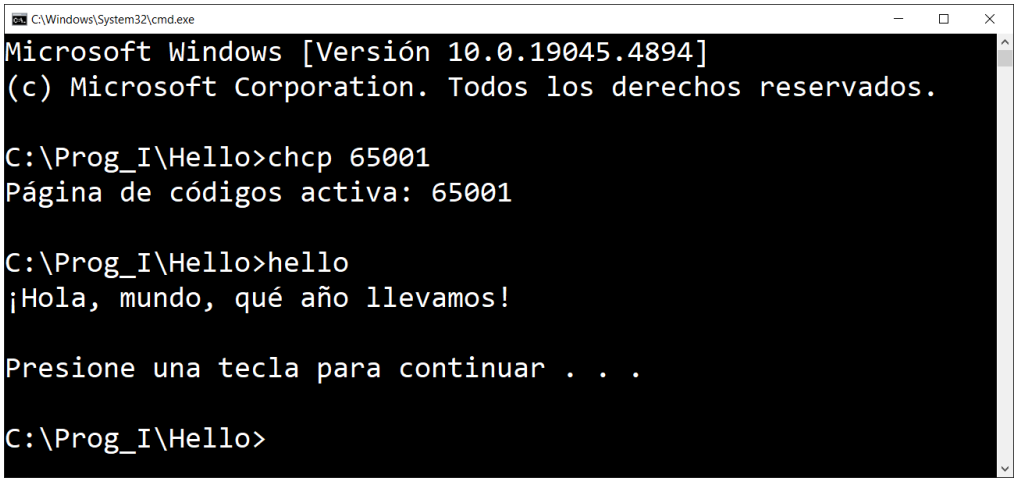
Lo que VSCode denomina terminal integrado es en realidad el programa `cmd.exe` ejecutado dentro del propio VSCode.

Para visualizar caracteres UTF-8 hay que activar la página de códigos 65001. Teclea en el terminal:

```
chcp 65001
```

Tras seleccionar la página de códigos 65001, el terminal visualizará correctamente los caracteres UTF-8.

La siguiente figura muestra una sesión de terminal en la que primero se selecciona la página de códigos 65001 y, a continuación, se ejecuta el programa de ejemplo tecleando su nombre. Observa como la salida de caracteres especiales es ahora totalmente correcta.



La figura anterior muestra el resultado de la ejecución en el terminal de Windows, pero el resultado habría sido el mismo si se hubiera utilizado el terminal integrado de VSCode.

El problema de esta solución es que, cada vez que se abre el terminal, hay que teclear la instrucción *chcp* antes de ejecutar el programa para que se reconozcan los caracteres UTF-8.

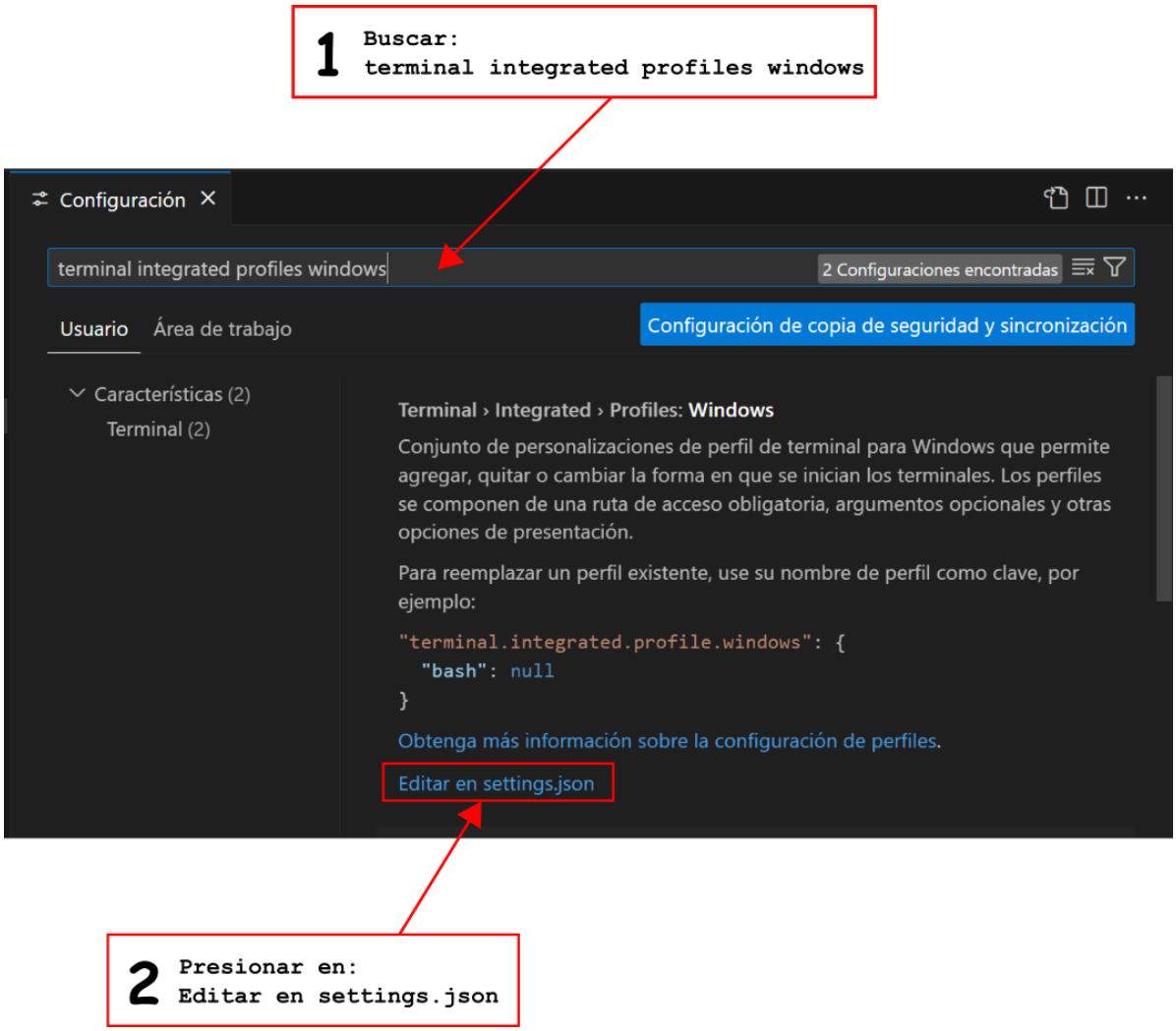
NOTA: Activar la consola externa desde VSCode

Recuerda que, para activar el terminal externo desde VSCode, puedes pinchar en la zona de edición de texto para que adquiera el foco y luego utilizar el atajo de teclado `CTRL+SHIFT+C`. Como resultado, se abrirá un terminal externo situado en la carpeta raíz abierta en el IDE.

2.2.- Personalización del terminal interno de VSCode

Se puede configurar VSCode para que, cada vez que arranque el terminal interno, ejecute la instrucción *chcp*. Para ello, hay que modificar el fichero *settings.json*, que es un fichero que utiliza VSCode para guardar las configuraciones personalizadas del usuario.

Abre el fichero *settings.json* utilizando la opción de menú *Fichero -> Preferencias -> Configuración*. En la barra de búsqueda de propiedades, escribe "*terminal integrated profiles windows*" y pulsa en "*Editar en settings.json*", como se indica en la siguiente figura:



De esta forma, el fichero *settings.json* se abrirá en el editor de texto. Dependiendo de las personalizaciones que haya realizado el usuario, el contenido del fichero será diferente. Observa un ejemplo particular a continuación:

```
{
    "workbench.startupEditor": "none",
    "terminal.integrated.defaultProfile.windows": "Command Prompt",
    "explorer.confirmDelete": false,
    "editor.minimap.enabled": false,
    "explorer.confirmDragAndDrop": false,
    "terminal.integrated.profiles.windows": {
        "PowerShell": {
            "source": "PowerShell",
            "icon": "terminal-powershell"
        },
        "Command Prompt": {
            "path": [
                "${env:windir}\\Sysnative\\cmd.exe",
                "${env:windir}\\System32\\cmd.exe"
            ],
            "args": [],
            "icon": "terminal-cmd"
        },
        "Git Bash": {
            "source": "Git Bash"
        }
    },
    "terminal.external.windowsExec": "C:\\pp.bat",
}
```

El fichero *settings.json* que hemos abierto en el editor es el que guarda las opciones de personalización de VSCode que estemos utilizando. Cada vez que usemos la opción *Archivo -> Preferencias -> Configuración* y modifiquemos alguna de las opciones por defecto de VSCode, este fichero recogerá la modificación y VSCode la utilizará.

Por ejemplo, observa la segunda línea:

```
"terminal.integrated.defaultProfile.windows": "Command Prompt",
```

Le indica a VSCode que el terminal integrado que debe usar por defecto en Windows es el Command Prompt. Es la modificación que indicábamos en el Manual de instalación de VSCode que tienes disponible en el Moodle de la asignatura. De hecho, deberías tener activada esta opción para seguir correctamente este manual.

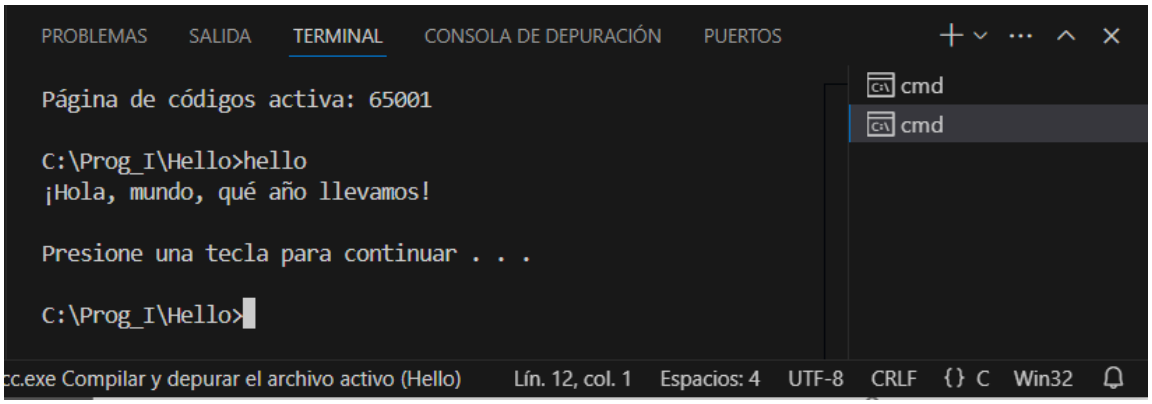
Ahora estamos interesados en el perfil de los terminales en Windows. Es la sección que empieza con la línea: `"terminal.integrated.profiles.windows"`. En el archivo de ejemplo se ofrecen tres perfiles de terminal: Power Shell, Command Prompt y Git Bash. Queremos modificar el perfil del Command Prompt, en concreto, en la línea que pone `"args":[]`. Debemos escribir entre los corchetes los parámetros que queremos pasar a la llamada a *cmd.exe*. Rellena entre los corchetes, de forma que la línea quede así:

```
"args": ["/K", "chcp", "65001"],
```

Lo que estamos haciendo es decirle a VSCode que, cuando abra el terminal, lo primero que debe hacer es ejecutar la instrucción *chcp 65001.*

Graba el fichero *settings.json* con la modificación, activa en el editor la pestaña del código fuente del programa *hello.c* y pulsa `CTRL+SHIFT+Ñ` para crear un nuevo terminal. Verás que ahora, al abrirse el terminal, muestra

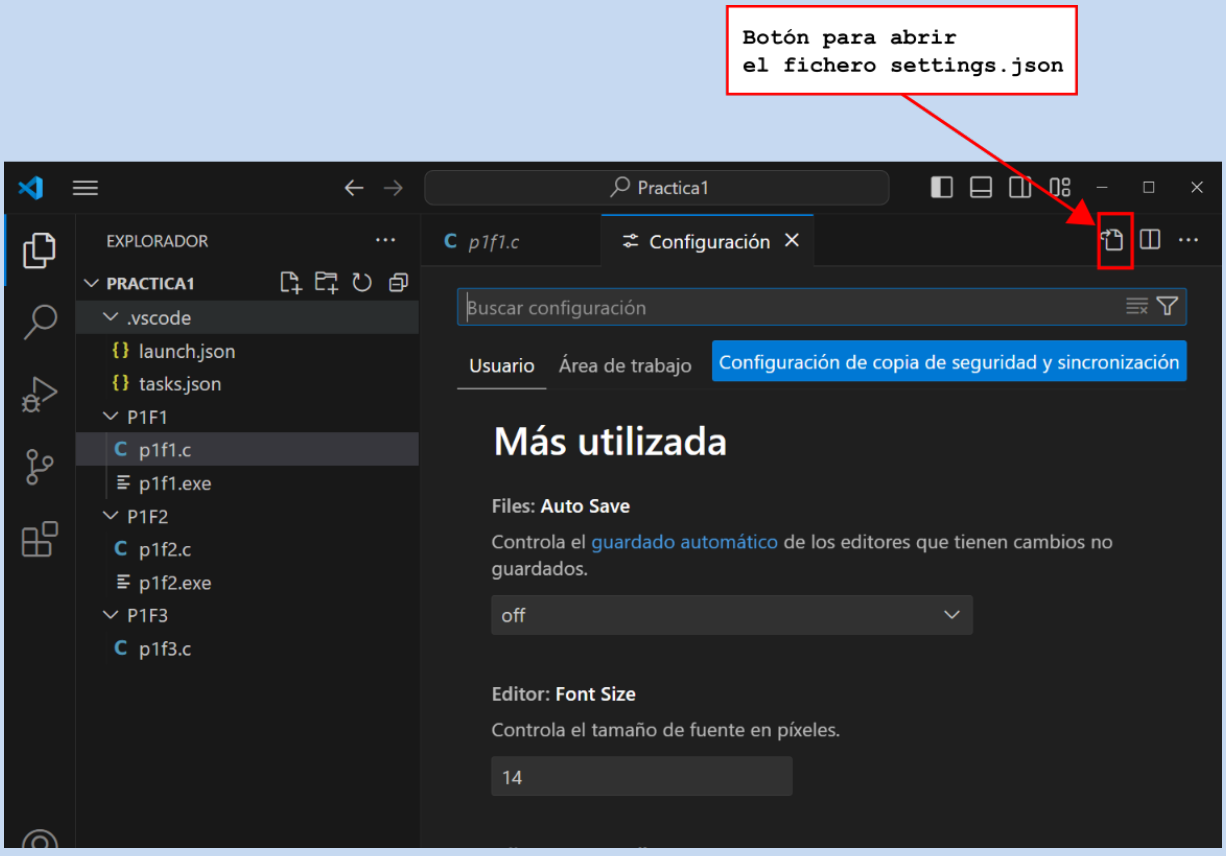
que la página de códigos activa es la 65001 y nuestro programa *hello* visualizará correctamente los caracteres especiales al ser ejecutado. La siguiente figura recoge el resultado:



Esta forma de proceder es muy útil y puede ser conveniente tener siempre configurado el terminal interno para que utilice la página de códigos 65001. El problema es que no nos sirve cuando queremos probar los programas en el terminal externo.

NOTA: Abrir el fichero de configuraciones

Puedes abrir el fichero *settings.json* directamente desde el botón que se ofrece en la parte superior de la ventana de configuración, abierta con *Archivo -> Preferencias -> Configuración*, tal como se muestra en la siguiente figura:



2.3.- Personalización del terminal de Windows

Se podría hacer algo parecido para personalizar el arranque del terminal externo, el de Windows. En este caso habría que crear un fichero *.bat* que fuera el que arrancara el terminal y que debería tener el siguiente contenido:

```
C:\Windows\System32\cmd.exe /K chcp 65001
```

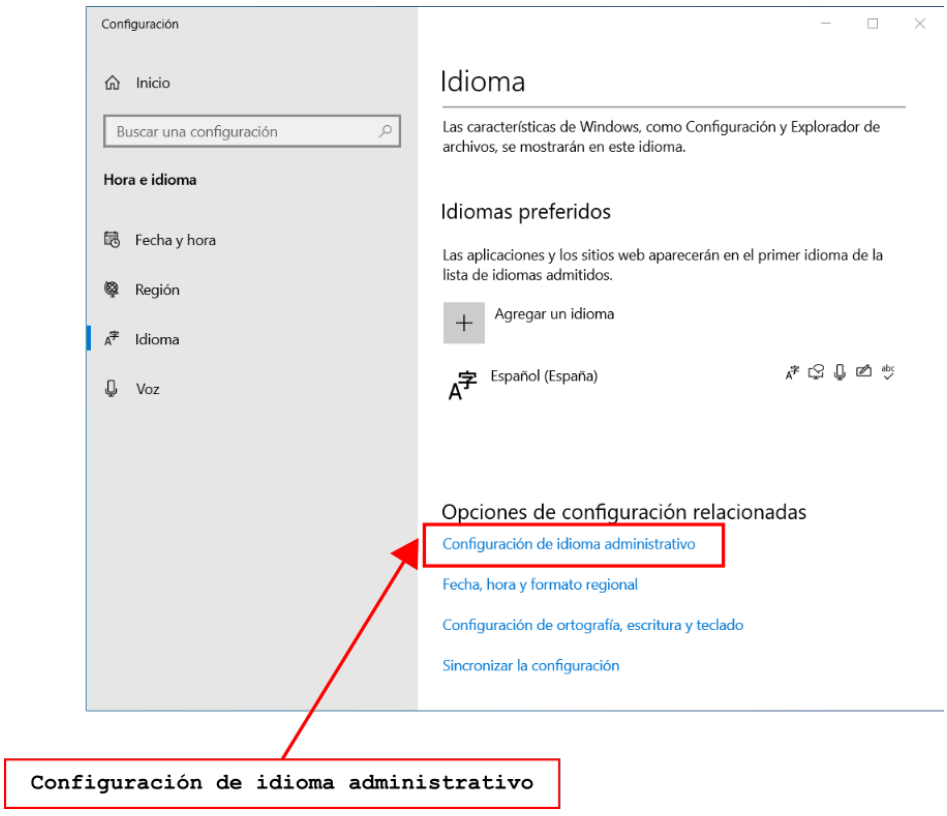
Se observa una única instrucción en la que se llama a *cmd.exe* y se le pasa como parámetro el comando que queremos ejecutar al arrancar. Podríamos grabar el fichero como *cmd.bat,* en el escritorio por ejemplo, e indicar a VSCode que lo llame para ejecutar el terminal externo. Para ello, hay que modificar la propiedad *Terminal->External->Windows Exec* de la configuración de VSCode.

Pero es más útil configurar Windows de manera global, como se explica en el siguiente apartado.

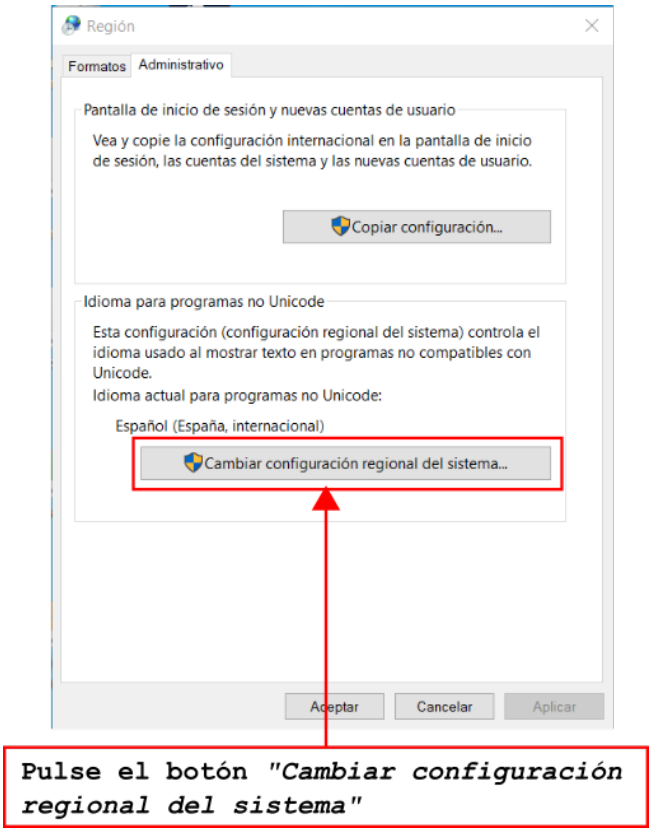
2.4.- Configuración global de Windows

Sin duda, esta es la solución más interesante y la que recomendamos que lleves a cabo. Las últimas versiones de Windows ofrecen una opción en la configuración de idioma que permite utilizar UTF-8 en el terminal. El proceso que debes seguir es el siguiente:

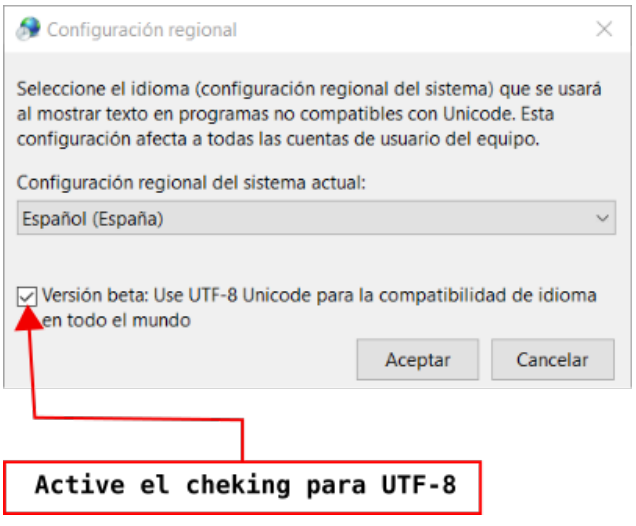
1. Buscar el diálogo *Configuración regional* y, en la opción correspondiente al *Idioma*, seleccionar el ítem *Configuración de idioma administrativo*. La siguiente figura corresponde a Windows 10, pero en Windows 11 es muy parecido:



2. Se abrirá un diálogo en el que debemos pulsar sobre el botón "*Cambiar configuración regional del sistema*", como se indica en la siguiente figura:



3. Esto abrirá un nuevo diálogo en el que hay que activar el checking "*Versión beta: Use UTF-8 Unicode para la compatibilidad de idioma en todo el mundo*", como indica la figura siguiente:



Al pulsar en el botón *Aceptar*, el sistema indicará que debes reiniciar el ordenador para que los cambios surtan efecto. Reinícialo y, a partir de ese momento, las ventanas de terminal utilizarán la página de códigos 65001. Esto será así para cualquier terminal que utilice el *cmd.exe*, tanto el terminal de Windows como los terminales integrados en VSCode.

3.- Forzar la visualización de UTF-8 desde el programa

Cuando los programas que hagamos se vayan a ejecutar en un ordenador distinto del nuestro, sobre el que no podemos actuar previamente, no sabremos a priori en qué página de códigos está configurada la consola. Por ello, debemos utilizar instrucciones dentro de nuestro programa que fuercen al sistema operativo a utilizar UTF-8. Vamos a explicar dos maneras de conseguirlo:

1. Mediante la función *system*.
2. Mediante las funciones *SetConsole*.

3.1.- Utilizando la función *system*

La función `system()` de la librería *windows.h* permite pasar órdenes al sistema operativo (Windows, en este caso). Recuerda la instrucción `system("pause")` que utilizamos en los programas para detener la ejecución, a la espera de que se pulse una tecla. Lo que estamos haciendo es indicar al sistema operativo que ejecute la instrucción *pause*, que es una instrucción del terminal de Windows, no de C (para comprobarlo, prueba a teclear la instrucción *pause* en la consola).

De la misma forma, podemos pasar al sistema operativo la instrucción *chcp 65001* que, como se ha explicado en el Apartado 2.1, fuerza a la consola a utilizar la codificación UTF-8. El ejemplo inicial modificado quedaría de la siguiente forma:

```
#include <stdio.h>
#include <windows.h>

int main() {
   system("chcp 65001");

   printf("¡Hola, mundo, qué año llevamos!\n\n");

   system("pause");
   return 0;
}
```

Prueba el código anterior y comprueba que la consola selecciona la página de códigos 65001 antes de imprimir el mensaje, con lo que los caracteres especiales se muestran correctamente. De esta forma, tendremos la garantía de que nuestro programa se ejecutará bien en cualquier ordenador Windows.

3.2.- Utilizar las funciones *setConsole*

La librería *windows.h* proporciona una pareja de funciones que permiten ajustar las entradas por teclado o las salidas por terminal a la codificación UTF-8. Son las siguientes:

- `SetConsoleCP(X)`: esta función establece la tabla de códigos para las entradas por teclado. Si sustituimos `X` por 65001, o bien por la constante `CP_UTF8` predefinida, que contienen el mismo valor, conseguiremos activar la codificación UTF-8 para la entrada por teclado. Esto sería: `SetConsoleCP(CP_UTF8)`.

- `SetConsoleOutputCP(X)`: esta función establece la tabla de códigos a usar por el terminal de salida. Aquí también debemos sustituir `X` por la constante `CP_UTF8` o su valor asociado 65001. Esto sería: `SetConsoleOutputCP(CP_UTF8)`.

Utilizando estas dos funciones, el programa de pruebas quedaría de la siguiente forma:

```
#include <stdio.h>
#include <windows.h>

int main() {
   SetConsoleCP(65001);
   SetConsoleOutputCP(65001);

   printf("¡Hola, mundo, qué año llevamos!\n\n");

   system("pause");
   return 0;
}
```

Prueba a ejecutar esta nueva versión del programa de pruebas y comprobarás que los caracteres especiales se muestran de manera correcta. En realidad, en este programa no sería necesario ajustar las entradas y se podría omitir la llamada a la función `SetConsoleCP()`.

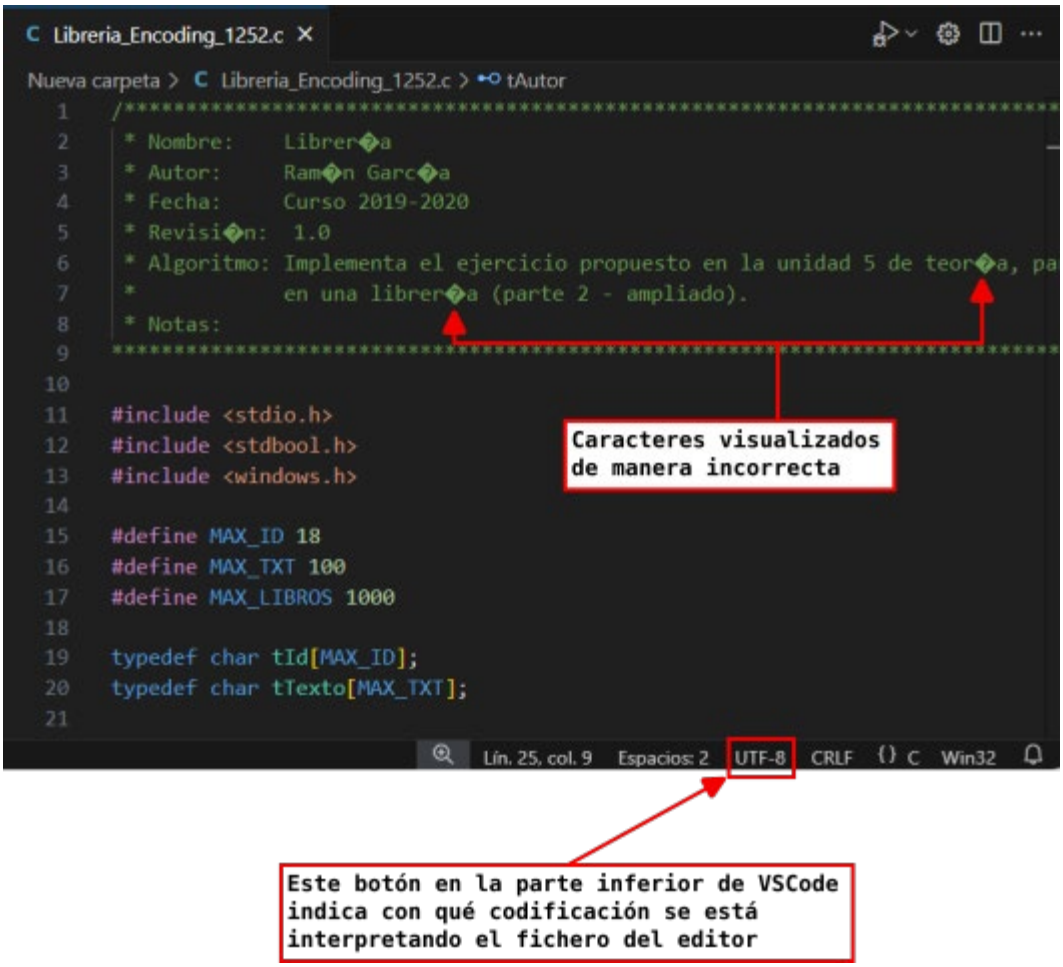
4.- Codificación de caracteres de los ficheros de texto

Los ficheros de texto plano, entre los que se encuentran los programas que escribimos en C, también están escritos en el disco con alguno de los sistemas de codificación existentes. VSCode utiliza UTF-8 por defecto. Windows también está adoptando la codificación UTF-8, pero, hasta hace unos años, utilizaba otros sistemas, y es habitual encontrarse con ficheros codificados con otras tablas de códigos. Una de las más comunes es la CP_1252.

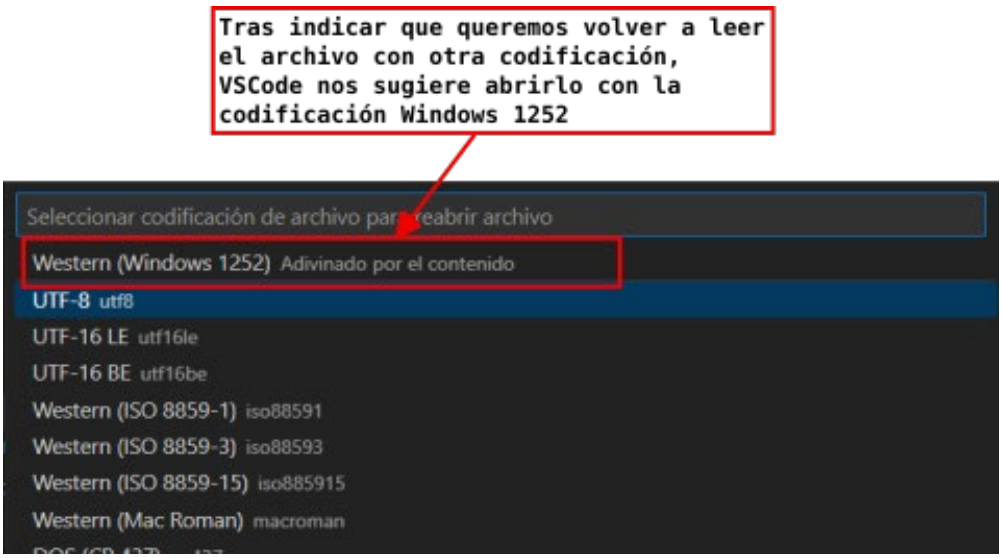
No es fácil saber con qué codificación está guardado un fichero, aunque VSCode intenta detectarla a partir de ciertas pistas que puede proporcionar el contenido del mismo.

En la parte inferior de la ventana de VSCode se indica con qué codificación se está leyendo el fichero que se muestra en pantalla. Por defecto, VSCode utiliza la codificación UTF-8 para interpretar los ficheros de texto que se presentan en el editor. Si se visualiza con codificación UTF-8 un fichero que está escrito con otra codificación, los caracteres especiales no se verán correctamente.

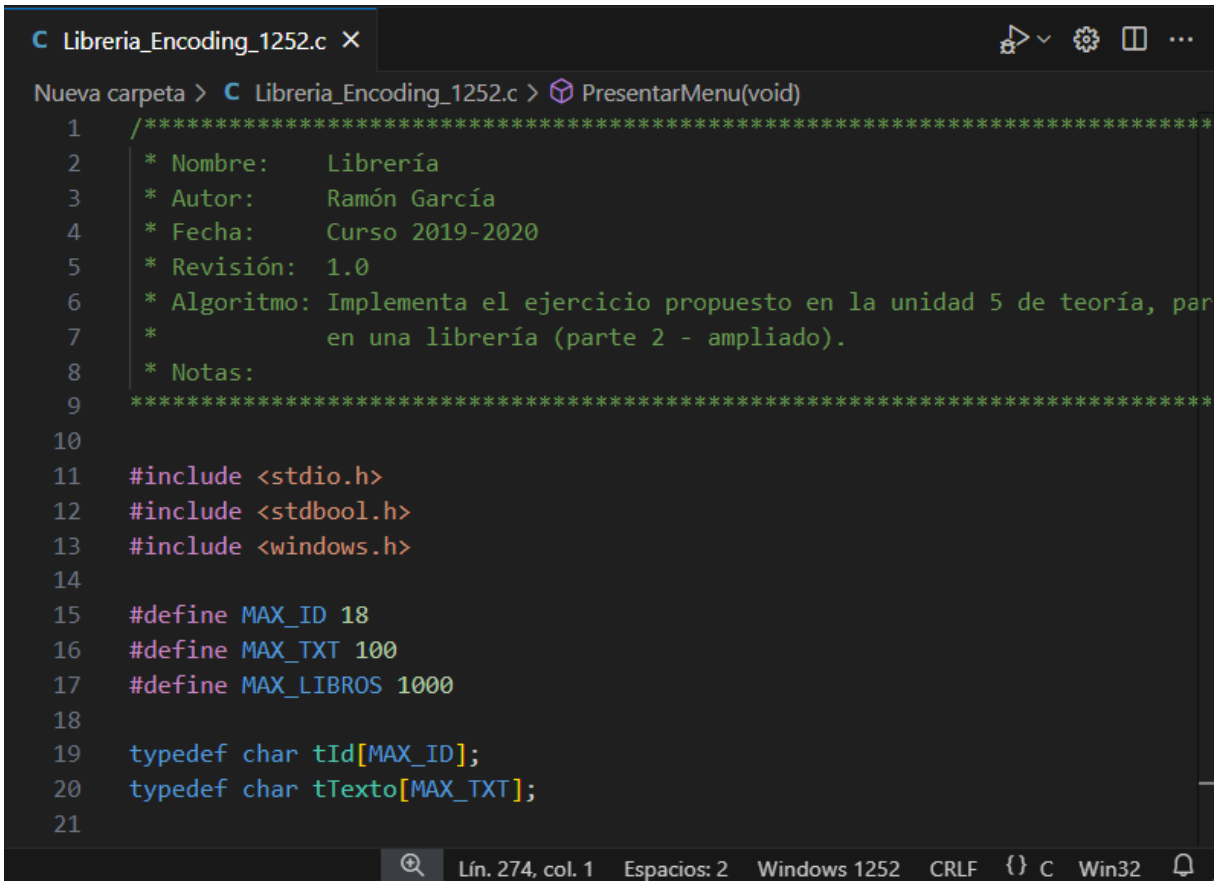
En la siguiente figura se muestra una parte de un programa C en el editor de VSCode. El fichero se ha leído e interpretado con codificación UTF-8. Observa que las letras acentuadas y otros caracteres del idioma español no se muestran correctamente, sino que aparece un rombo con una interrogación en su lugar. VSCode no está siendo capaz de interpretar adecuadamente los caracteres especiales, pues los intenta interpretar con UTF-8 y, en realidad, el fichero está escrito en el disco con la codificación Windows 1252 (CP_1252). Esta es una situación muy habitual.



Para visualizar correctamente los caracteres especiales, hay que volver a leer el fichero del disco utilizando la codificación en la que se escribió: pulsa el botón de la parte inferior de VSCode en el que se indica la codificación, el que está enmarcado en la figura anterior. VSCode nos ofrecerá volver a leer el fichero con otra codificación o escribirlo en el disco con otra codificación. Si elegimos la opción *Volver a cargar con Encoding*, VSCode nos propondrá leerlo con la codificación más adecuada, como se ve en la siguiente figura, aunque también podemos elegir la que nosotros deseemos:



En la siguiente figura se muestra el fichero tras abrirlo con la codificación Windows 1252.



Observa que ahora los caracteres especiales se muestran correctamente en pantalla y que, en la parte inferior de la ventana de VSCode, la codificación ha cambiado a Windows 1252.

Una vez que hemos leído correctamente el fichero, podemos decidir guardarlo con otra codificación. Normalmente es preferible tener los ficheros guardados con codificación UTF-8. Para ello, sólo tenemos que volver a pulsar el botón de la codificación y elegir la opción *Guardar con encoding*, seleccionando la página de códigos con la que queramos guardar el fichero, en este caso, UTF-8.

A partir de ese momento, el fichero quedará escrito en el disco en codificación UTF-8 y, cuando lo abramos en modo UTF-8, los caracteres especiales se visualizarán correctamente.

NOTA: Tabla de códigos de los ficheros

Tienes que ser consciente de que la tabla de códigos que establezcas en VSCode para abrir un fichero, es la que se usará para compilarlo, y con ella tendrás que emparejar la de la consola. Así, es una buena práctica, como ya se ha indicado en este documento, disponer de todos los ficheros en UTF-8 y configurar la consola con esta misma tabla de códigos.