

## ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA Y SISTEMAS DE TELECOMUNICACIÓN

# PROGRAMACIÓN I

*Entorno de desarrollo para programar en C:*

*Compilador y VSCode*

# ÍNDICE

<b>1.- INTRODUCCIÓN .....</b>	<b>3</b>
<b>2.- INSTALACIÓN DE UN COMPILADOR DE LENGUAJE C .....</b>	<b>3</b>
2.1.- INSTALACIÓN EN ORDENADORES CON SISTEMA OPERATIVO WINDOWS .....	3
2.2.- INSTALACIÓN EN ORDENADORES MAC .....	5
2.3.- INSTALACIÓN EN ORDENADORES LINUX (UBUNTU) .....	5
<b>3.- AJUSTE DE LA VARIABLE PATH EN LOS ORDENADORES WINDOWS .....</b>	<b>6</b>
<b>4.- INSTALACIÓN DE VSCODE .....</b>	<b>8</b>
4.1.- INSTALACIÓN DEL EDITOR .....	8
4.2.- INSTALACIÓN DE LOS COMPLEMENTOS PARA VSCODE .....	9
4.3.- CONFIGURAR EL TERMINAL POR DEFECTO EN WINDOWS.....	10
4.4.- EL ENTORNO DE TRABAJO DE VSCODE .....	11
<b>5.- PRUEBA DE LA INSTALACIÓN.....</b>	<b>12</b>
5.1.- CONCEPTOS PREVIOS.....	12
5.2.- CREACIÓN DE LA ESTRUCTURA DE CARPETAS.....	13
5.3.- CREACIÓN DEL PROGRAMA <i>HOLA, MUNDO</i> .....	15
5.4.- COMPILAR EL PROGRAMA.....	16
5.5.- EJECUCIÓN DEL PROGRAMA .....	17
5.5.1.- Ejecución desde la consola integrada en VSCode.....	17
5.5.2.- Ejecución del depurador desde VSCode .....	18
5.5.3.- Agregar la configuración de depuración.....	20
5.5.4.- Ejecución en un terminal externo.....	20
5.5.5.- Ejecución desde el explorador de archivos del sistema operativo.....	21
<b>ANEXO 1.- DESINSTALACIÓN DE MINGW .....</b>	<b>23</b>
<b>ANEXO 2.- INTERPRETES DE INSTRUCCIONES.....</b>	<b>23</b>
A2.1.- EL TERMINAL INTEGRADO EN VSCODE .....	24

## 1.- Introducción

Para desarrollar programas en lenguaje C, vamos a utilizar dos herramientas: un compilador de C y el editor de texto Visual Studio Code (VSCode).

En los ordenadores de los laboratorios de la Escuela ya están disponibles las dos herramientas por lo que, cuando estés trabajando en ellos, no necesitarás instalar nada. En cambio, si vas a trabajar en tu ordenador personal, necesitarás instalar las dos cosas y realizar algunos ajustes.

El proceso de preparación de un ordenador para trabajar en C consta de las siguientes fases:

- Instalación del compilador de C.
- Ajuste de la variable *path* (solo en sistemas *Windows*).
- Instalación de *VSCode*.
- Instalación de complementos en *VSCode*.
- Prueba de la instalación.

Cada una de las fases se realiza de manera algo diferente según el sistema operativo que utilices. En los próximos apartados se va a explicar cómo poner a punto tu ordenador para implementar los programas en lenguaje C que desarrollaremos a lo largo del curso.

### NOTA: Desinstalación de MinGW

Si seguiste el curso 2023-24, es posible que tengas instalado el compilador MinGW, que es el que utilizábamos entonces. Puedes seguir utilizándolo, si lo deseas, pero si prefieres instalar el compilador que proponemos en este curso, es mejor que desinstales previamente MinGW. Para ello sigue las instrucciones del Anexo 1.

## 2.- Instalación de un compilador de lenguaje C

### 2.1.- Instalación en ordenadores con sistema operativo Windows

Instalaremos el conjunto de herramientas de compilación que ofrece MSYS2, entre las que se incluye el compilador GCC para Windows y el depurador GDB. El enlace al instalador lo puedes encontrar en la página:

<https://www.msys2.org/>

Tienes que descargar el instalador pulsando el botón que se indica en la siguiente Figura:

# MSYS2

## Software Distribution and Building Platform for Windows

MSYS2 is a collection of tools and libraries providing you with an easy-to-use environment for building, installing and running native Windows software.

It consists of a command line terminal called [mintty](#), bash, version control systems like git and subversion, tools like tar and awk and even build systems like autotools, all based on a modified version of [Cygwin](#). Despite some of these central parts being based on Cygwin, the main focus of MSYS2 is to provide a build environment for native Windows software and the Cygwin-using parts are kept at a minimum. MSYS2 provides up-to-date native builds for GCC, mingw-w64, CPython, CMake, Meson, OpenSSL, FFmpeg, Rust, Ruby, just to name a few.

To provide easy installation of packages and a way to keep them updated it features a package management system called [Pacman](#), which should be familiar to Arch Linux users. It brings many powerful features such as dependency resolution and simple complete system upgrades, as well as straight-forward and reproducible package building. Our package repository contains [more than 3200 pre-built packages](#) ready to install.

For more details see ['What is MSYS2?'](#) which also compares MSYS2 to other software distributions and development environments like [Cygwin](#), [WSL](#), [Chocolatey](#), [Scoop](#), ... and ['Who Is Using MSYS2?'](#) to see which projects are using MSYS2 and what for.

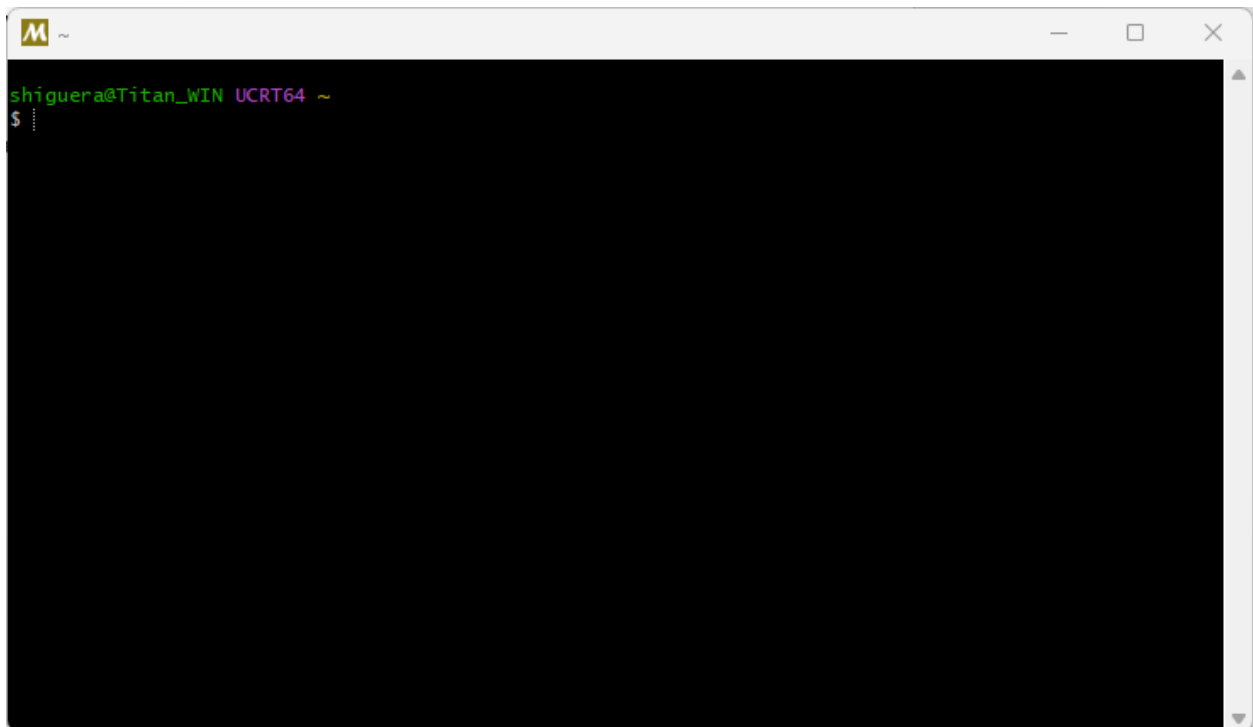
### Installation

1. Download the installer: [msys2-x86\\_64-20240727.exe](#)

**Enlace al instalador**

Una vez descargado el instalador, ejecútalo, aceptando las opciones por defecto que se te proponen para el directorio de instalación y para la carpeta de accesos directos.

Al finalizar la instalación acepta la opción de ejecutar MSYS2, con lo que se abrirá una ventana como la siguiente:



Esta es la consola de MSYS2. Para instalar los componentes del compilador, debes teclear la siguiente instrucción (tecléala exactamente o cópiala desde esta página y pégala en la consola de MSYS2):

```
pacman -S --needed base-devel mingw-w64-ucrt-x86_64-toolchain
```

Pulsa la tecla *INTRO* para ejecutar la orden y acepta las opciones por defecto que se te vayan proponiendo. La instalación pesa unos 170 MB, por lo que llevará cierto tiempo.

Tras terminar la instalación de todos los componentes, deberías poder ver en el explorador de archivos de *Windows* la carpeta `C:\msys64`, que es donde han sido integradas las herramientas del compilador.

Ahora, puedes cerrar la ventana de la consola y pasar al apartado que explica cómo ajustar la variable *path* del sistema operativo.

### NOTA: Abrir la consola de MSYS2

Podría suceder que necesitaras abrir la consola de MSYS2, por ejemplo, porque la cerraste o no indicaste que “se abriera al finalizar la instalación”. Si así fuera, tienes que ejecutar el programa `clang64.exe` que se encuentra en el directorio en el que se instaló MSYS2 (seguramente `C:\msys64`).

## 2.2.- Instalación en ordenadores Mac

En los ordenadores Mac uno de los posibles compiladores a instalar es *Clang*. Podría ser que ya lo tuvieses instalado. Para comprobarlo, abre un terminal y teclea la siguiente instrucción:

```
clang --version
```

Si el compilador ya está instalado, aparecerá un mensaje indicando el número de la versión disponible. Si no está instalado, lo puedes hacer tecleando en el terminal la siguiente instrucción:

```
xcode-select --install
```

Alternativamente, también puedes instalar en tu Mac el compilador *MinGW* siguiendo el procedimiento que se explica en el vídeo que hay colgado en el Moodle de la asignatura, dentro del apartado “*Recursos generales para la realización de las prácticas*”.

Puedes pasar desde aquí al apartado de instalación de *VSCode*.

## 2.3.- Instalación en ordenadores Linux (Ubuntu)

En los ordenadores con sistema operativo Linux, en concreto los ordenadores con el sistema Ubuntu (el más habitual), abre un terminal y teclea la siguiente instrucción para comprobar si el compilador ya está instalado:

```
gcc --version
```

Si no está instalado, lo puedes hacer tecleando las siguientes instrucciones:

```
sudo apt-get update
sudo apt-get install build-essential gdb
```

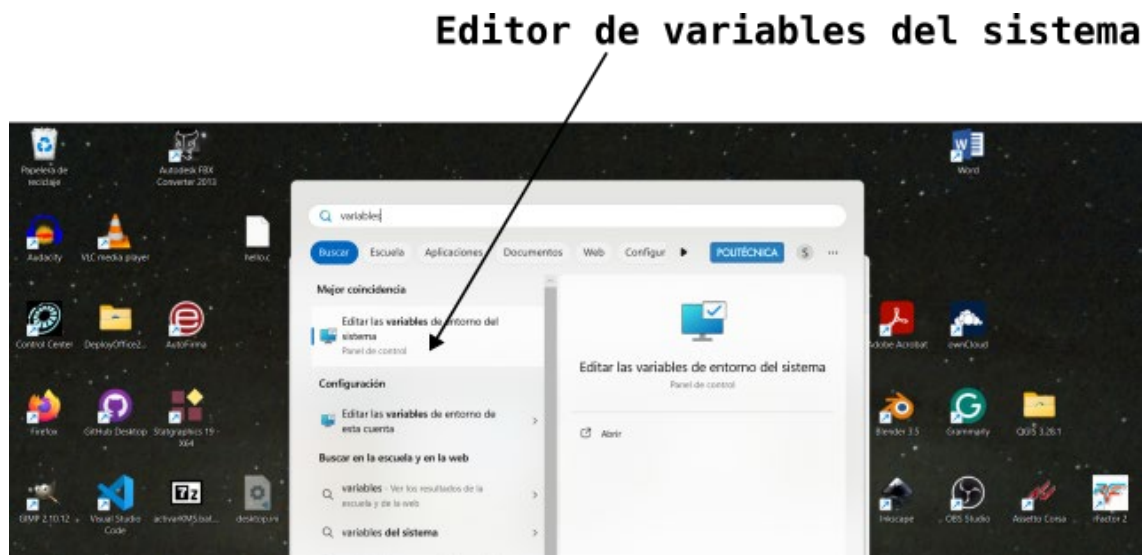
Observa que, además de las herramientas de compilación, se instala también el depurador *gdb*.

Ahora puedes pasar al apartado de instalación de *VSCode*.

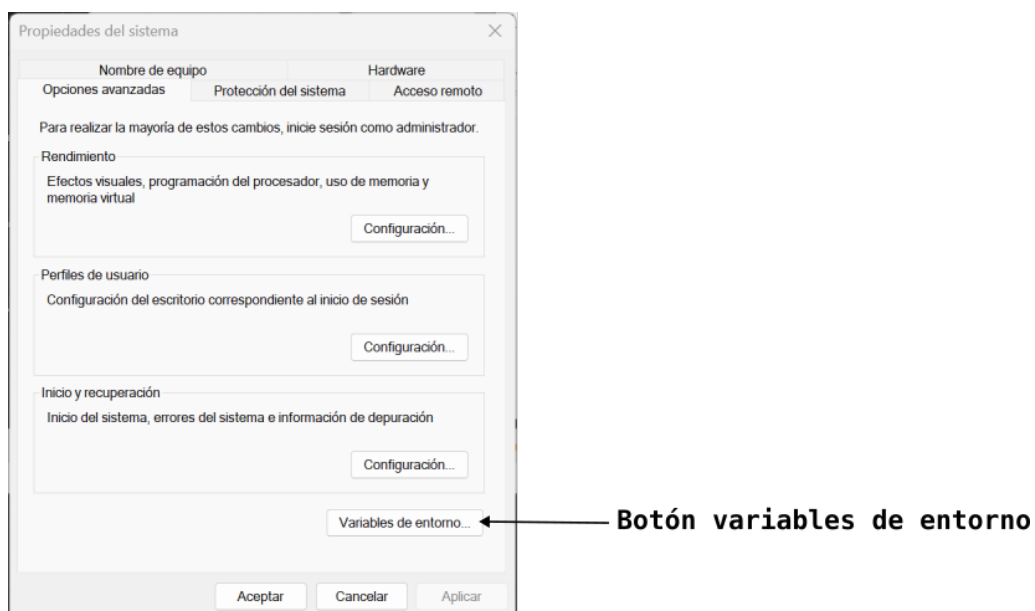
### 3.- Ajuste de la variable *path* en los ordenadores Windows

Este ajuste sólo hay que hacerlo en el caso de los ordenadores *Windows*. Consiste en guardar el directorio de instalación del compilador en la variable *path* del sistema operativo, para que el sistema operativo sea capaz de encontrarlo cuando sea necesario.

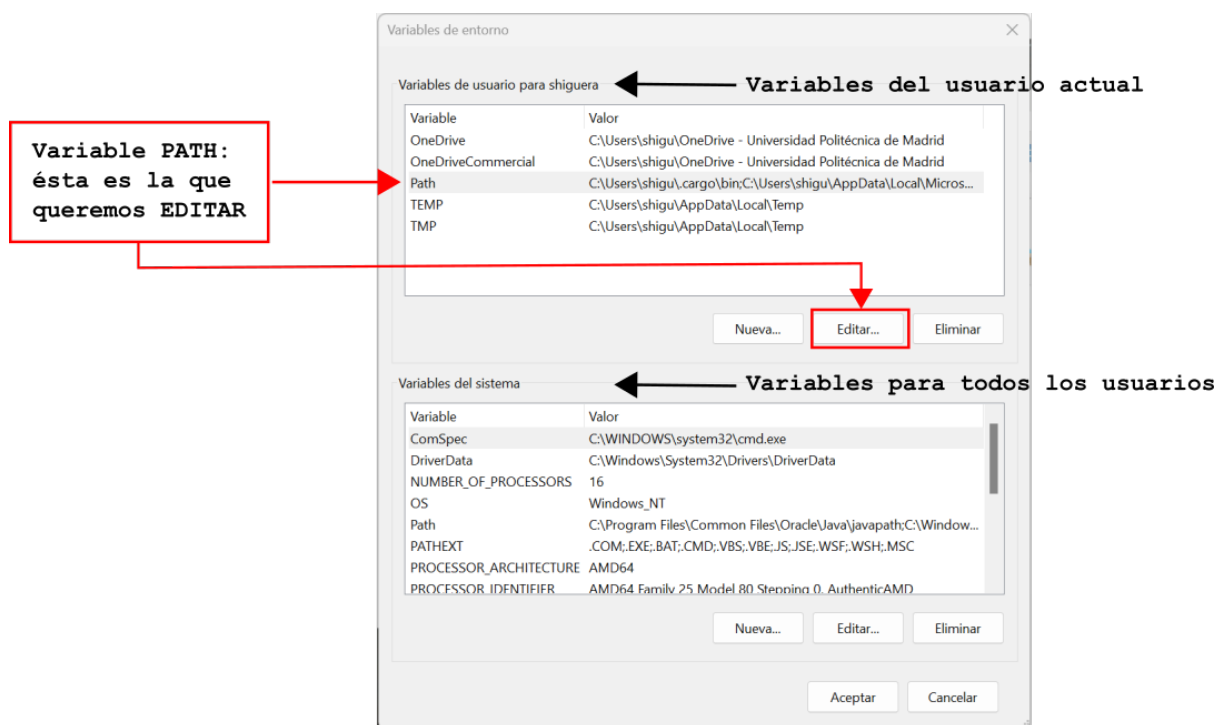
Para ajustar la variable *path*, debes acceder al diálogo “*Editar las variables de entorno del sistema*”. Puedes buscar el diálogo escribiendo “*variables*” en el buscador, como se ha hecho en la siguiente figura:



Al abrir el editor de variables, se mostrará un diálogo como el de la siguiente figura, donde debemos pulsar el botón *Variables de entorno*:



Este botón abrirá un nuevo diálogo, que se muestra en la siguiente figura, y que es en el que estamos interesados:



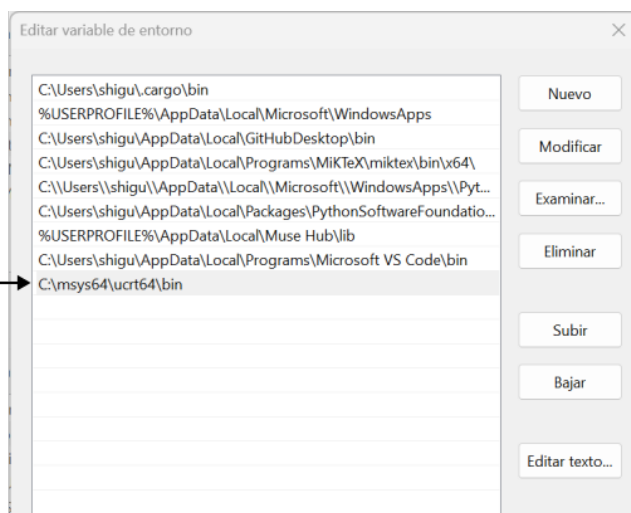
### NOTA: Acerca de la variable *PATH*

Observa que Windows dispone de dos conjuntos de variables: uno para el usuario actual, en la parte superior, y otro, en la parte inferior del diálogo, con variables globales que se aplican a todos los usuarios del ordenador. Nosotros debemos modificar la variable *path* del usuario actual.

**¡Muy importante!: la variable *path* se debe editar, no hay que eliminarla. Si la eliminas, es posible que tengas que reinstalar Windows.**

Selecciona la variable *path* del usuario y pulsa EDITAR. Al hacerlo se nos mostrará un nuevo diálogo que contendrá las distintas rutas de directorios que hay guardadas en dicha variable. Son las rutas que usa *Windows* para buscar los programas cuando se solicita su ejecución. Según la instalación de cada uno, esas rutas son diferentes. En la siguiente figura, se muestra el diálogo que aparece en un determinado ordenador:

Variable que hay que añadir:  
**C:\msys64\ucrt64\bin**



Hay que pulsar el botón *Nuevo*, para añadir una nueva ruta y teclear la dirección del directorio `ucrt64\bin` de nuestra instalación del compilador. Si durante la instalación aceptaste las opciones por defecto que se ofrecían, la ruta que hay que añadir es la siguiente:

**C:\msys64\ucrt64\bin**

Una vez añadida dicha ruta, cierra los diálogos que se han abierto pulsando el botón *ACEPTAR*. A partir de ese momento, el compilador debería estar correctamente instalado en nuestro sistema *Windows*. Para comprobarlo, abre el terminal del sistema (consola) y teclea la siguiente instrucción:

**gcc --version**

El terminal debería mostrar un mensaje con información de la versión del compilador instalada, similar al de la siguiente figura:

Si no es así, vuelve a repetir cuidadosamente los pasos de la instalación desde el principio.

## **NOTA: Abrir el terminal**

Recuerda que, para abrir el terminal del sistema, debes teclear en el buscador *cmd* o *terminal*, con lo que aparecerá el programa que Windows llama “*Símbolo del Sistema*”. Este terminal se usará frecuentemente, por lo que es recomendable que lo ancles a la barra de tareas para poder acceder más fácilmente.

## **4.- Instalación de VSCode**

### **4.1.- Instalación del editor**

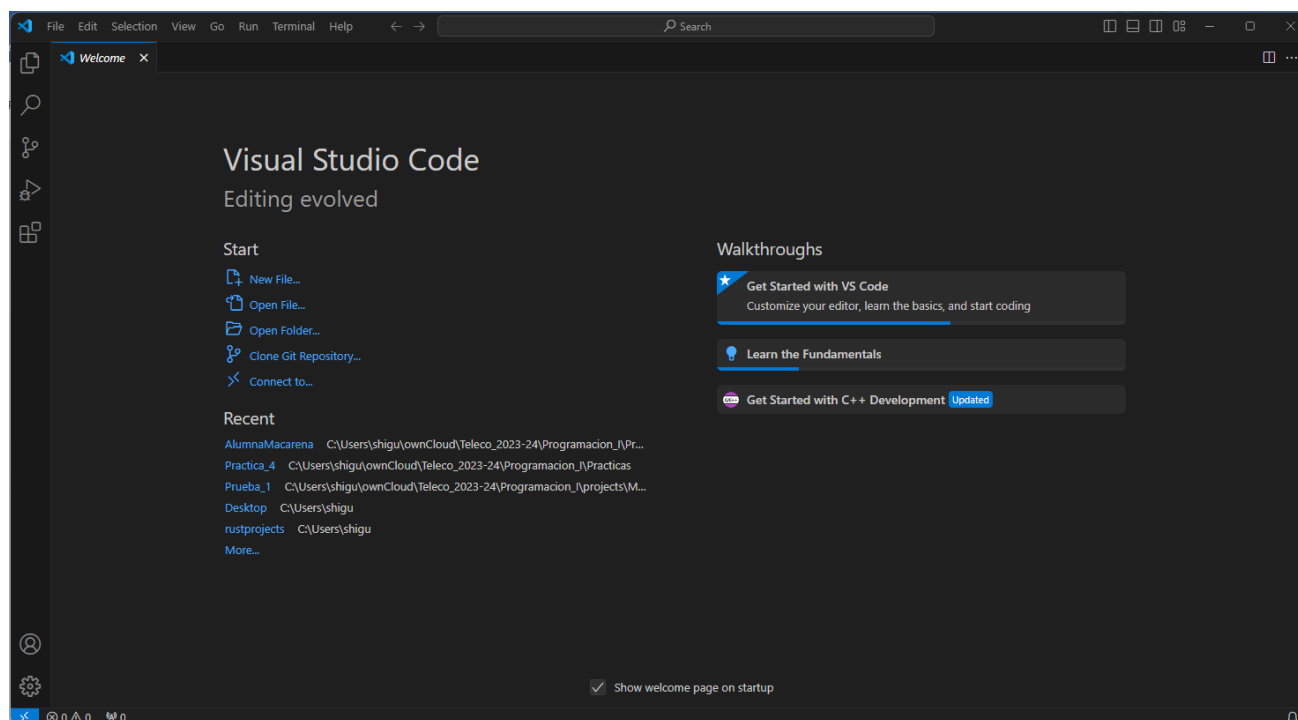
Para instalar *VSCode*, debes ir a la página Web del programa y descargar la versión adecuada a tu sistema operativo. Normalmente, al entrar en la página se nos ofrece directamente la versión más conveniente. La dirección Web es la siguiente:

<https://code.visualstudio.com/>



Descarga y ejecuta el instalador, aceptando las opciones por defecto que se ofrecen. La única excepción a esta regla es que conviene marcar la opción “*Crear un acceso directo en el escritorio*”. Si no lo has hecho, no te preocupes, podrás añadir el acceso más adelante.

El último paso de la instalación nos ofrece ejecutar *VSCode*. Podemos aceptar y aprovechar para anclar el programa a la barra de tareas, pues, como sucede con el terminal, lo usaremos frecuentemente. Al ejecutar *VSCode*, deberías ver una ventana similar a la siguiente:



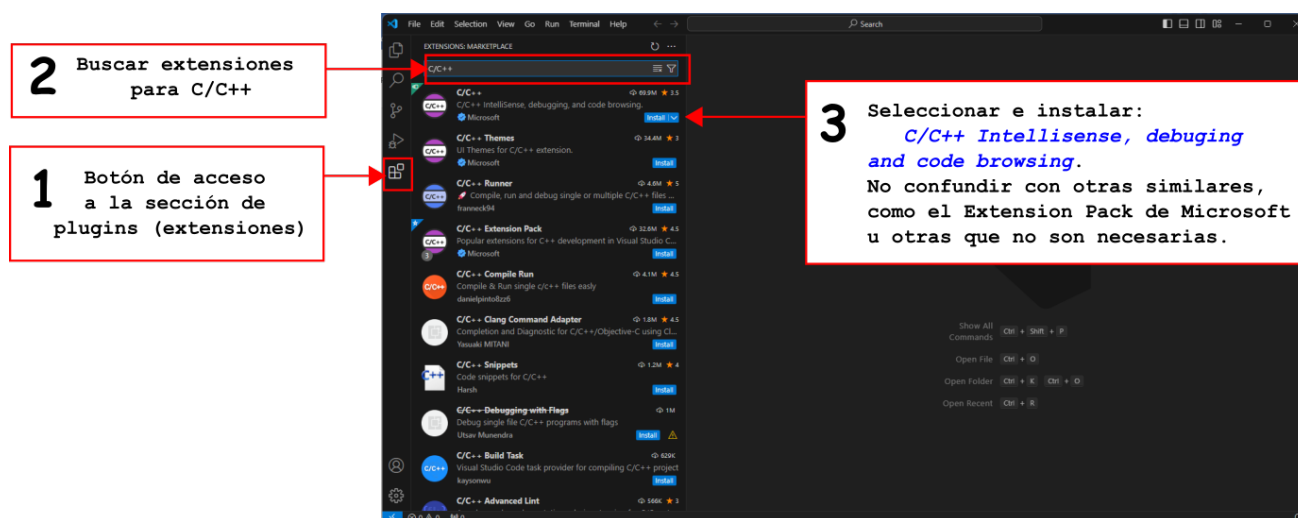
El editor nos muestra una pestaña de bienvenida (*Welcome*). Siempre que abramos *VSCode* se presentará esta pestaña, salvo que lo desactivemos en el *checking* que parece en la parte inferior de la misma.

## 4.2.- Instalación de los complementos para VSCode

*VSCode* es un editor que permite escribir programas en cualquier lenguaje. Ofrece algunas extensiones (*plugins*) que facilitan la tarea de programar en cada lenguaje concreto.

Para programar en lenguaje C vamos a añadir algunos complementos que facilitarán la escritura, la compilación, la depuración y la ejecución de los programas.

En primer lugar, vamos a instalar la extensión para programar en C. Para ello, pulsa sobre el botón correspondiente a las extensiones que aparece en la barra izquierda de la ventana de *VSCode*, como se indica en la siguiente figura:



Debes instalar la extensión llamada “*Intellisense, debugging and Code Browsing*”. No la confundas con otras similares que no es necesario instalar y que pueden crear confusión cuando se está aprendiendo a programar.

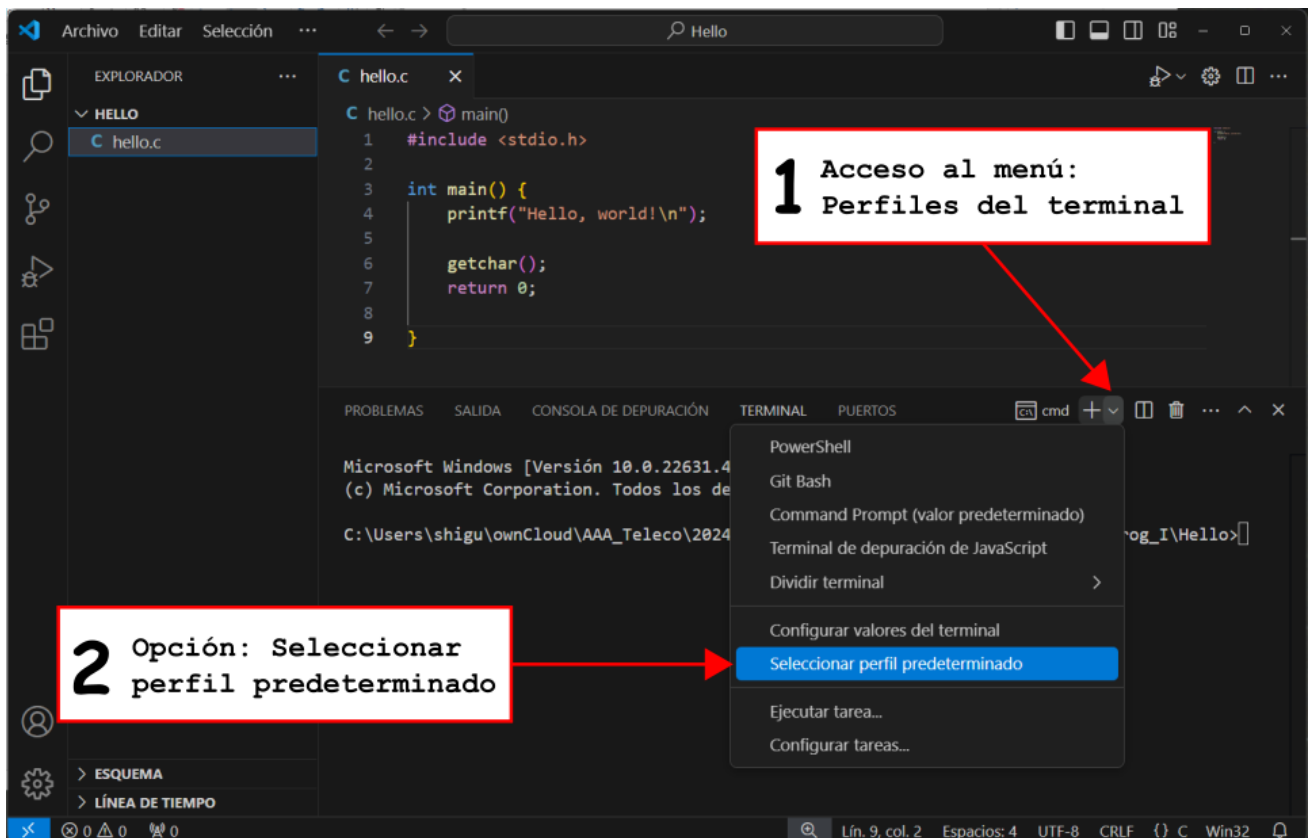
### NOTA: VSCode en español

Es posible que, por defecto, VSCode tenga todos los menús y opciones en inglés. Si no te sientes cómodo trabajando en inglés y prefieres que VSCode utilice español, puedes instalar la extensión “*Spanish Language Pack for Visual Studio Code*”. Para ello, abre la sección de Extensiones, busca dicha extensión y pulsa sobre el botón instalar.

## 4.3.- Configurar el terminal por defecto en Windows

VSCode ofrece varias posibilidades para el terminal integrado. En concreto, en el sistema operativo Windows, se ofrecen dos terminales: *PowerShell* y *CommandPrompt*. Es posible que tengas configurado por defecto la utilización de *PowerShell*, pero en este curso preferimos utilizar el terminal *CommandPrompt*. Se puede configurar VSCode para que utilice por defecto el terminal *CommandPrompt*; tienes que seguir los siguientes pasos:

1. Abre el terminal con la opción de menú *Ver -> Terminal*, o utilizando el atajo de teclado **CTRL+ñ**.
2. Abre el menú *Perfil de inicio* del terminal y selecciona la opción “*Seleccionar perfil predeterminado*”. En la siguiente figura se muestra cómo abrir el menú mencionado.



3. En la parte superior de la ventana se te ofrecerá el terminal que quieres asociar al perfil predeterminado: Selecciona *Command Prompt*. A partir de ese momento, cuando abras el terminal integrado en el editor, se mostrará una ventana de comandos con la misma funcionalidad que la de Windows.

#### 4.4.- El entorno de trabajo de VSCode

Los editores de texto que se utilizan para programar, como VSCode, ofrecen numerosas ayudas pensadas específicamente para los desarrolladores. Es por ello que frecuentemente se denominan *Entornos de Desarrollo Integrado* o IDE (*Integrated Development Environment*).

No hay que confundirlos con los *procesadores de texto*, como Word, que se utilizan para escribir documentos, no programas. La principal diferencia es que los editores de texto trabajan con ficheros de *texto plano*. Texto plano quiere decir que los ficheros solo tienen texto. En los procesadores de texto, como Word, los ficheros tienen, además del texto, otros códigos que indican el tipo de fuente, el formato de los párrafos y otras características del documento que se está escribiendo.

Cuando se abre un fichero de programa en VSCode, observarás que también hay diferentes colores para diferentes tipos de instrucciones. Estos colores no están definidos en el fichero del programa, forman parte de las ayudas que ofrece el editor. El editor es capaz de distinguir distintos tipos de instrucciones y mostrarlas en diferentes colores, para facilitar la programación, pero el fichero del programa sólo tiene el texto plano.

VSCode es un editor de texto pensado para escribir programas en cualquier lenguaje. Se trata de un entorno muy completo que ofrece numerosas ayudas a la programación. Conviene aprender a

utilizarlo de manera eficiente, pues ello redundará en un aumento de la productividad y una mayor precisión al codificar los programas.

Puedes consultar la siguiente página web, para comprender un poco mejor el entorno de trabajo:

<https://code.visualstudio.com/docs/getstarted/userinterface>

## 5.- Prueba de la instalación

Vamos a desarrollar el típico programa “*Hola, Mundo*”, para comprobar que hemos hecho correctamente la instalación del compilador y de VSCode. Pero, antes, vamos a definir algunos términos habituales en programación que nos van a hacer falta.

### 5.1.- Conceptos previos

**Programa:** se suele llamar así a los archivos ejecutables, aunque a veces también se llama así al propio código fuente del programa. En Windows, los archivos ejecutables tienen la extensión `.exe`. En Mac o en Linux no necesitan ninguna extensión especial.

Los programas `.exe` son el resultado de *compilar* uno o más ficheros fuente. Cuando se programa en lenguaje C, los ficheros fuente tienen la extensión “`.c`”. La mayoría de los programas que haremos a lo largo del curso constan de un solo fichero “`.c`” que, al compilarlo, dará lugar al programa ejecutable correspondiente. A medida que los programas van siendo más complicados, empiezan a necesitar varios ficheros fuente “`.c`”, ficheros de cabeceras “`.h`” e incluso ficheros de otros tipos. El programa será el resultado de compilar de manera ordenada todos esos ficheros para obtener el ejecutable correspondiente. Surge así el concepto de *proyecto*.

**Proyecto:** se suele denominar así al conjunto de ficheros que se necesitan para que, al compilarlos, se obtenga el ejecutable correspondiente. En VSCode, todos los ficheros de un proyecto están en una misma carpeta que es la denominada *carpeta del proyecto*.

**Construir el proyecto (*build*):** se denomina así al conjunto de operaciones que hay que hacer con los archivos de un proyecto para obtener el programa ejecutable. También se le suele llamar *compilar el programa*, aunque, en realidad, además de compilar se tienen que hacer otras operaciones, por ejemplo, *enlazar*.

#### NOTA: Acerca de los nombres de archivos y carpetas

Los nombres que utilices para los archivos de los programas o para las carpetas de los proyectos solo deben utilizar letras del alfabeto inglés, en mayúsculas o minúsculas, y el guión bajo.

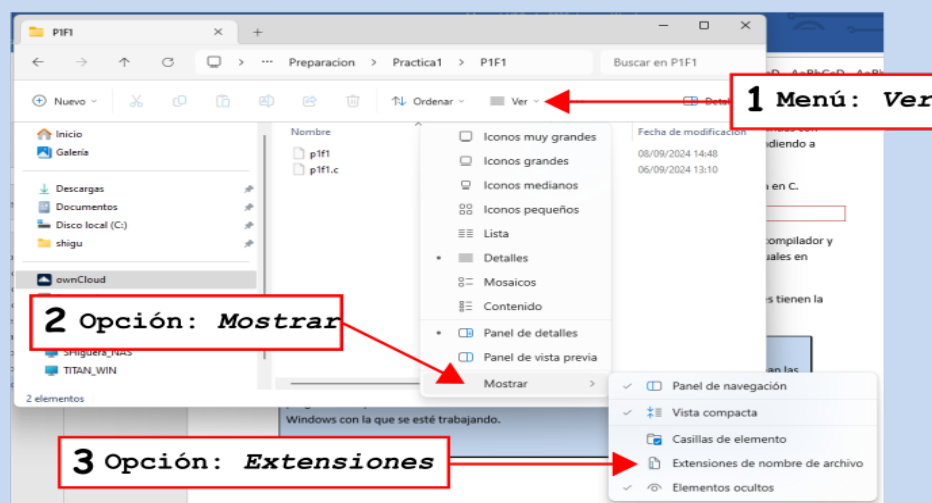
**¡NO UTILICES LETRAS COMO LA Ñ, ESPACIOS, LETRAS ACENTUADAS U OTROS CARACTERES ESPECIALES PARA LOS NOMBRES DE ARCHIVOS O CARPETAS, PUES PUEDE DAR LUGAR A ERRORES!**

## NOTA: Extensión de los nombres de archivos en Windows

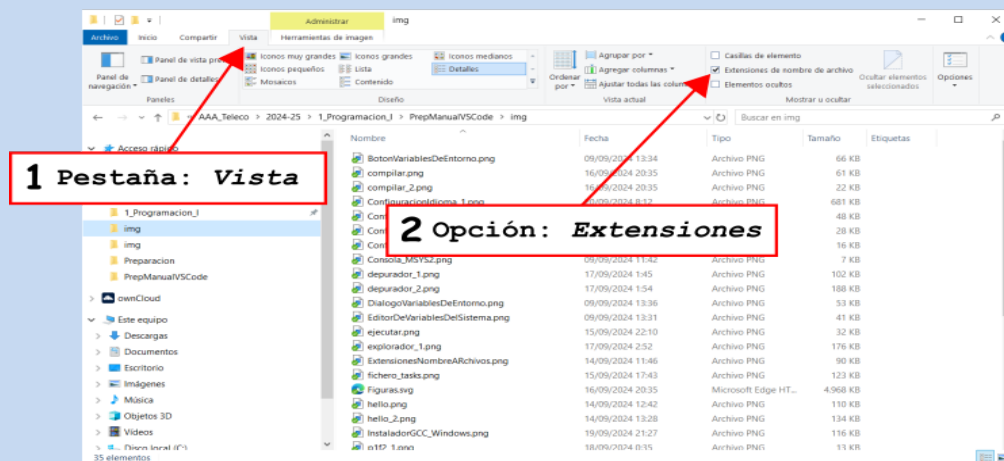
El explorador de archivos de Windows establece por defecto la opción de que no se vean las extensiones.

Esta opción es muy incómoda cuando se está programando y conviene desactivarla. La forma de hacerlo es diferente según la versión de Windows. En general, es una de las opciones que ofrece Windows como propiedades de la visualización de las carpetas en el explorador. En Windows 11, una forma de hacer que se vean las extensiones de los archivos es en la opción del menú del explorador **Ver -> Mostrar -> Extensiones de nombre de archivo**. En Windows 10 es parecido. Las siguientes figuras pueden guiarte sobre cómo hacerlo.

### EXPLORADOR DE ARCHIVOS EN WINDOWS 11



### EXPLORADOR DE ARCHIVOS EN WINDOWS 10



## 5.2.- Creación de la estructura de carpetas

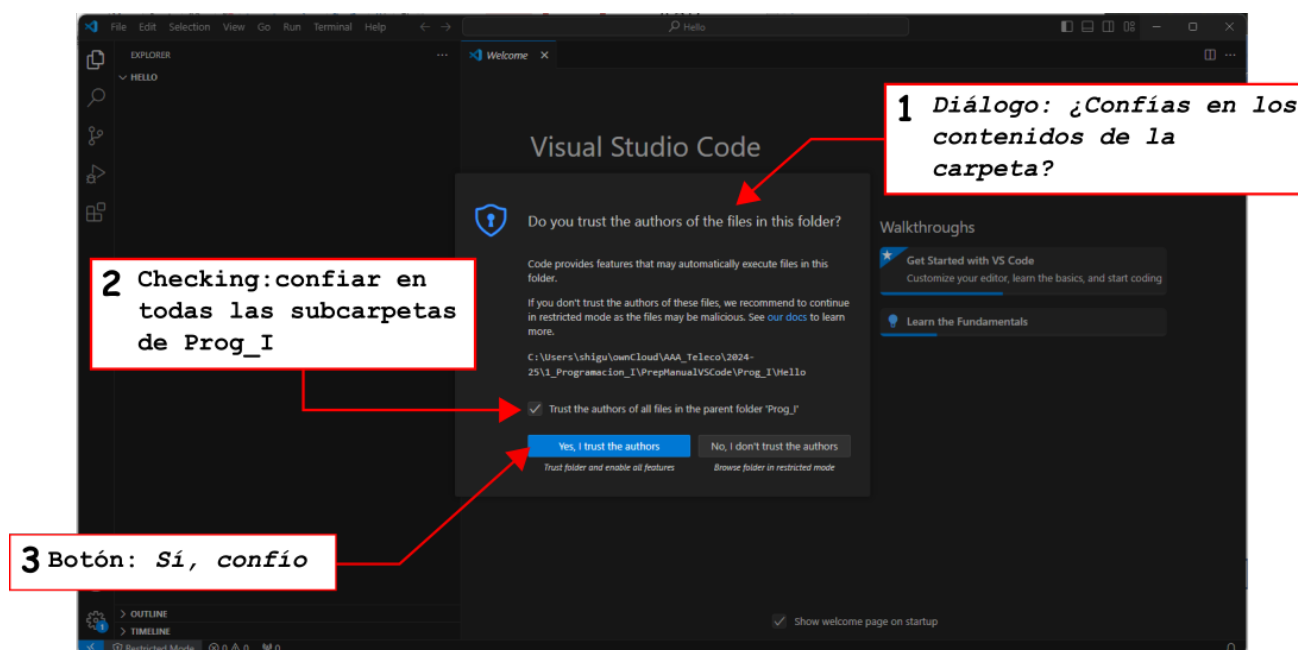
Lo normal es agrupar todos los ficheros de un proyecto dentro de una misma carpeta (directorio) del ordenador.

Puede ser conveniente también agrupar todas las carpetas de todos los proyectos dentro de un mismo directorio que denominaremos raíz. Os proponemos crear en el ordenador una carpeta llamada `Prog_I`, dentro de la cual iremos generando a lo largo del curso carpetas independientes para cada proyecto que hagamos.

En los ordenadores del laboratorio, cada alumno tiene asignada una unidad de disco llamada “I:”. Dentro de ella, es donde el alumno debe crear su carpeta `I:\Prog_I`. Si en tu ordenador utilizas Windows, lo normal es que el disco se llame “C:” y, por tanto, la carpeta que se creará será: `C:\Prog_I`. En ordenadores Mac o Linux se puede crear el directorio `Prog_I` dentro de cualquiera de las carpetas de usuario.

En VSCode, cada proyecto tiene asignada una carpeta del ordenador y, para trabajar en él, hay que abrir la carpeta correspondiente con la opción de menú `Fichero -> Abrir carpeta`.

La primera vez que se va a trabajar en un proyecto, será necesario crear la carpeta correspondiente. Se puede hacer directamente desde VSCode o bien desde el explorador de archivos de Windows. Crea un directorio llamado `Hello` dentro de la carpeta `Prog_I` y abre esta última con VSCode. Deberías ver algo parecido a lo que muestra la siguiente figura:

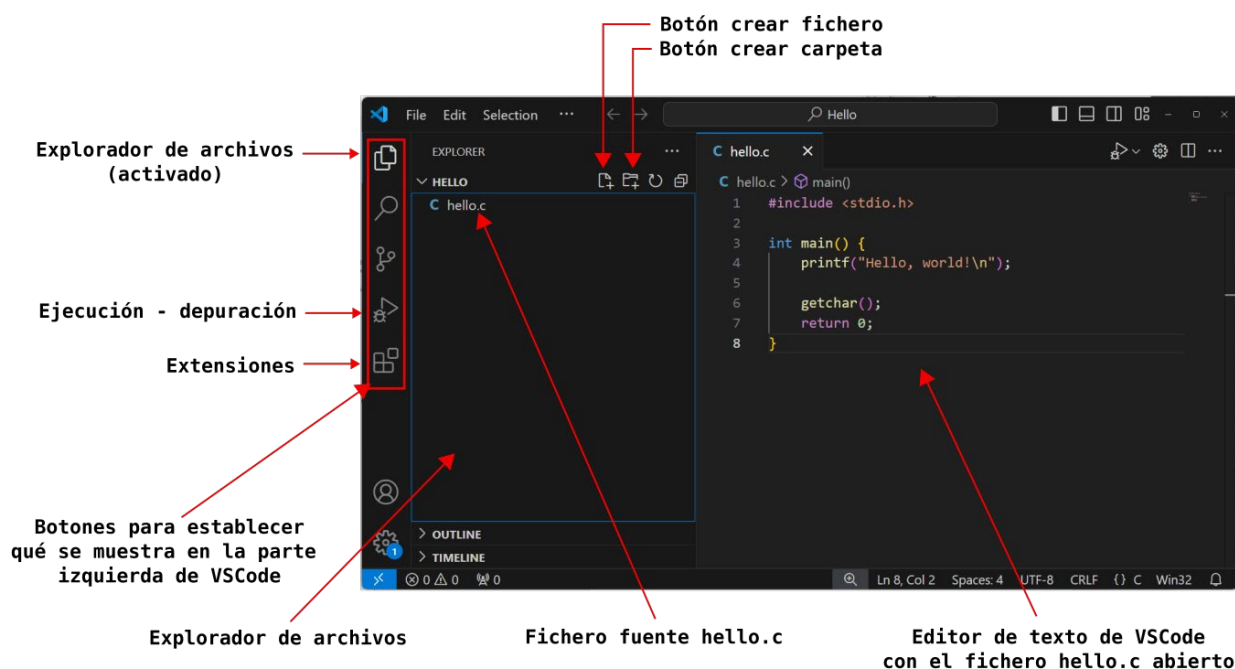


La primera vez que abrimos una carpeta en VSCode se nos ofrece un diálogo para confiar en los contenidos de dicha carpeta. También hay un *checking* para establecer que se confía en todas las subcarpetas a partir de ese momento. Conviene activarlo (observa la figura anterior).

Una vez establecido que confiamos en la carpeta, veremos en el editor la pestaña *Welcome*. Como ya se ha dicho, la visualización de esta pestaña se puede desactivar en el *checking* situado en la parte inferior de la ventana. En cualquier caso, debemos cerrar dicha pestaña con lo que tendremos el editor preparado para hacer nuestro primer programa.

La parte izquierda de VSCode es el explorador de archivos, aunque los botones de la barra izquierda permiten también acceder a las extensiones o el entorno de ejecución, como ya se ha visto durante la instalación. Dicho

explorador tiene unos botones en la parte superior que permiten crear ficheros y carpetas, como se muestra en la siguiente figura:



### 5.3.- Creación del programa *Hola mundo*

Dentro de la carpeta `Hello` del proyecto, crea un fichero llamado `hello.c`. Utilizando el editor, teclea el código del programa. Si tu ordenador usa Windows, teclea el siguiente código dentro del fichero:

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

int main() {
    printf("Hello, world!\n");
    system("pause");
    return 0;
}
```

Si tu ordenador no utiliza Windows, puedes teclear el siguiente código alternativo.

```
#include <stdio.h>

int main() {
    printf("Hello, world!\n");
    printf("Pulse INTRO para finalizar\n");
    getchar();
    return 0;
}
```

Por el momento no te preocupes de comprender el contenido del programa anterior, lo irás aprendiendo durante el curso.

Una vez que hayas tecleado el código del programa tienes que grabar el fichero en disco con la opción de menú **Fichero -> Guardar** (*File -> Save*), aunque conviene que te acostumbres a usar el atajo **CTRL+S**.

## NOTA: Atajos de teclado

Cuando se está programando, la utilización de los atajos de teclado incrementa mucho la productividad y es muy conveniente acostumbrarse a utilizarlos. Suelen consistir en la pulsación de dos o tres teclas simultáneamente. Es habitual utilizar combinaciones de la tecla Control (CTRL), de la tecla Mayúsculas (SHIFT) o de la tecla ALT junto con otra tecla.

En las líneas anteriores, cuando se escribe CTRL+F5, por ejemplo, se refiere a que hay que pulsar la tecla Control y, sin soltarla, pulsar la tecla F5.

No obstante, los atajos de teclado pueden ser diferentes de unos sistemas operativos a otros, o incluso en el mismo sistema operativo, según el idioma en el que se tenga configurado el teclado. Lo normal es que cada opción de menú indique a su derecha el atajo de teclado que le corresponde.

También es posible configurar los atajos de teclado de manera personalizada. En el siguiente enlace se puede acceder a la información que ofrece VSCode en relación con los atajos de teclado, e incluso descargar una chuleta con los más habituales para los distintos sistemas operativos:

<https://code.visualstudio.com/docs/getstarted/keybindings>

## 5.4.- Compilar el programa

Como ya sabes, para poder ejecutar el programa en el ordenador, hay que traducirlo a código máquina. Este proceso es el que se denomina *compilar el programa*.

En nuestro caso, partimos del código fuente del programa en el fichero *hello.c* y lo compilaremos para obtener el programa ejecutable. En Windows, el ejecutable se llamará *hello.exe*; en Mac y Linux se llamará simplemente *hello*.

Para compilar el programa hay que utilizar la opción de menú `Terminal -> Ejecutar tarea de compilación`, cuyo atajo de teclado es CTRL+SHIFT+B.

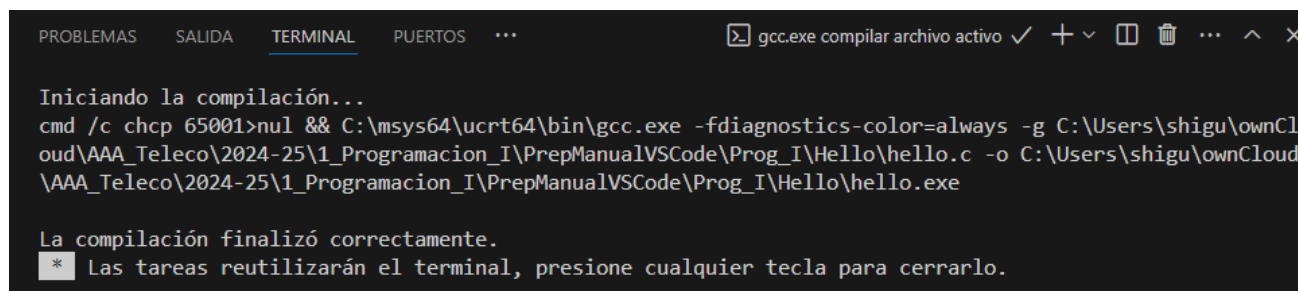
La primera vez que realicemos la compilación es probable que VSCode nos ofrezca elegir el compilador que queremos utilizar. La figura siguiente muestra VSCode ofreciendo opciones para la elección del compilador. Tenemos que seleccionar `gcc.exe`.



Una vez que seleccionemos el compilador `gcc`, se abrirá la consola en la parte inferior de la ventana y nos mostrará los mensajes generados durante la compilación. Si hay errores, se nos presentarán mensajes indicando en qué líneas del código fuente se producen los errores. Si no hay errores, el resultado será similar



al de la siguiente figura, donde puedes ver el mensaje “La compilación finalizó correctamente”, mostrado en el terminal integrado en el IDE.



```
PROBLEMAS  SALIDA  TERMINAL  PUERTOS  ...  gcc.exe compilar archivo activo  +  -  [ ]  [X]  ...  ^  X

Iniciando la compilación...
cmd /c chcp 65001>nul && C:\msys64\ucrt64\bin\gcc.exe -fdiagnostics-color=always -g C:\Users\shigu\ownCloud\AAA_Teleco\2024-25\1_Programacion_I\PrepManualVSCode\Prog_I\Hello\hello.c -o C:\Users\shigu\ownCloud\AAA_Teleco\2024-25\1_Programacion_I\PrepManualVSCode\Prog_I\Hello\hello.exe

La compilación finalizó correctamente.
* Las tareas reutilizarán el terminal, presione cualquier tecla para cerrarlo.
```

Si la compilación se realiza correctamente, en el explorador de archivos de VSCode podrás ver que ahora está también el ejecutable *hello.exe* (en Windows) o simplemente *hello* (en otros sistemas operativos).

### NOTA: El archivo a compilar debe tener el foco

Para compilar un fichero, dicho fichero tiene que estar abierto en el editor de texto de VSCode. Además, hay que pinchar con el ratón dentro del editor, para ponerlo en foco. Es entonces cuando se puede utilizar el atajo CTRL+SHIFT+B para realizar la tarea de compilación (o, alternativamente, seleccionar la opción de menú *Terminal -> Ejecutar tarea de compilación*)

## 5.5.- Ejecución del programa

Hay distintas formas de ejecutar el programa que acabamos de crear. Vamos a explicar algunas de ellas.

### 5.5.1.- Ejecución desde la consola integrada en VSCode

Observa el terminal bajo el editor: Debajo del mensaje “La compilación finalizó correctamente” indica que hay que pulsar una tecla para cerrar el terminal de compilación; Pincha con el ratón en el terminal, para ponerlo en foco y, a continuación, pulsa alguna tecla para cerrar los mensajes de compilación.

Deberías ver el terminal *Ventana de comandos (Command prompt)*, indicando el directorio actual y con el cursor esperando a que tecleemos alguna orden. La instrucción que hay que teclear es el nombre del programa que queremos ejecutar: *hello* (en Windows, para ejecutar un fichero *.exe*, no es necesario teclear la extensión, basta con teclear el nombre, al igual que en el resto de sistemas operativos). Si el *prompt* no está dentro de la carpeta del proyecto, tendrás que meterte previamente en su interior usando la instrucción *cd NombreCarpeta*. Si todo va bien, deberías ver un resultado similar al de las siguientes figuras.

La primera figura corresponde a la ejecución de la primera versión del programa, la versión para Windows que utiliza la instrucción *system("pause")*. El programa imprime el mensaje "*Hello, world!*" y queda esperando a que se pulse una tecla para finalizar.

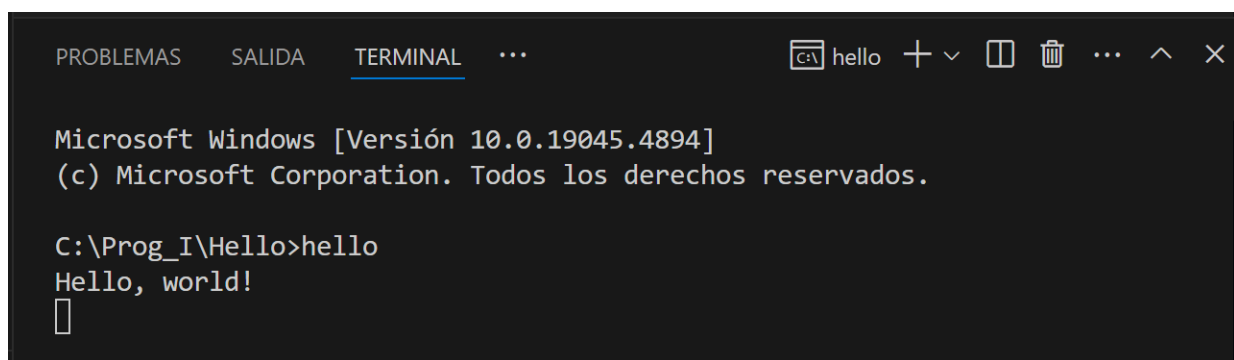


```
TERMINAL ... c:\ hello + - [ ] [ ] ... ^ X

C:\Prog_I\Hello>hello
Hello, world!
Presione una tecla para continuar . . .

Lín. 10, col. 14  Espacios: 4  UTF-8  CRLF  { } C  Win32
```

Si has utilizado la segunda versión de código, que sirve para cualquier sistema operativo (incluido Windows), el resultado será similar al de la siguiente figura:



```
PROBLEMAS  SALIDA  TERMINAL ... c:\ hello + - [ ] [ ] ... ^ X

Microsoft Windows [Versión 10.0.19045.4894]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Prog_I\Hello>hello
Hello, world!
█
```

En este caso, el programa imprime en pantalla “Hello, world!”, y, como resultado de la instrucción `getchar()`, queda esperando hasta que pulsemos la tecla INTRO. Una vez pulsada dicha tecla, el programa termina su ejecución, el terminal muestra el directorio actual, y el cursor queda esperando a que tecleemos alguna instrucción.

La razón de no haber utilizado directamente esta segunda versión para cualquier sistema operativo, habiendo presentado una alternativa para Windows, es que en este sistema operativo podemos tener problemas cuando durante el programa solicitamos al usuario números enteros, ya lo estudiarás detenidamente en clase. Como en este programa no se realiza ese tipo de petición podemos usar cualquiera de las dos versiones expuestas.

### 5.5.2.- Ejecución del depurador desde VSCode

Otra alternativa para ejecutar el programa es lanzar el depurador desde las opciones de menú de VSCode. Para ello, tienes que utilizar la opción de menú `Ejecutar -> Iniciar depuración (Run -> Debug)`, cuyo atajo de teclado es `F5`, alternativamente a la opción `Ejecutar -> Ejecutar sin depuración (Run -> Run without debugging)`, cuyo atajo de teclado es `CTRL+F5`, y que nos llevaría al mismo estado del apartado anterior.

En cualquier caso, en ambas opciones, el resultado será el mismo: se ejecuta el programa, compilando previamente si fuera necesario, y se muestra el resultado. La diferencia es que, en nuestro caso (con depuración), el programa se detiene antes de finalizar, y en la parte superior de la ventana de VSCode se

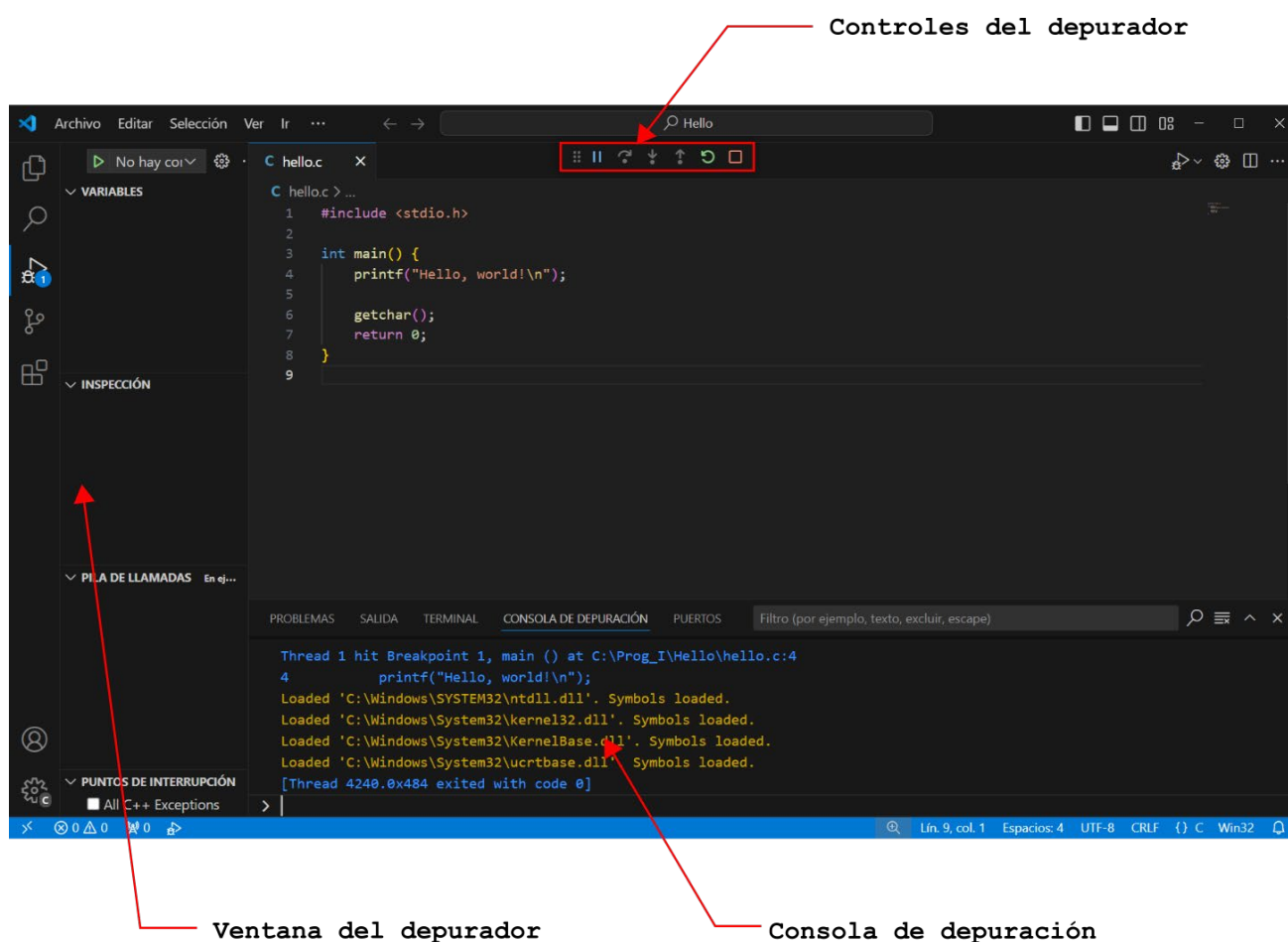
mostrarán los controles del depurador. Por el momento, no vamos a utilizar el depurador. En la Práctica 1 implementaremos un ejemplo para explicar cómo usarlo.

Como sucedía al compilar, la primera vez que intentemos ejecutar un programa mediante esta opción, se nos pedirá que seleccionemos el depurador que queremos utilizar. Tenemos que seleccionar la opción indicada como C/C++ (GDB/LLDB) y, a continuación, elegir GCC.

Sucedarán varias cosas:

1. El terminal que se muestra en la parte inferior de VSCode cambia al denominado *Consola de depuración*.
2. En la parte izquierda de VSCode, en vez de “el explorador de archivos”, se muestra la ventana del depurador.
3. En la parte superior de VSCode aparecen los controles del depurador.

La siguiente Figura muestra el estado en que queda VSCode tras ordenar ejecutar el programa en el modo de depuración:



En este momento, el programa está detenido en la instrucción `getchar()`, esperando que pulsemos la tecla `INTRO` para finalizar. Para hacerlo, debemos seleccionar la pestaña “Terminal de la consola”, picar en ella para ponerla en foco y pulsar la tecla `INTRO`. Con ello, el programa finalizará su ejecución.

Tras finalizar la ejecución del programa, se debería mostrar en la parte izquierda de VSCode el Explorador de archivos (si no es así, actívalo tú mismo con el botón correspondiente). Observa que ahora hay una nueva

carpeta colgando del directorio raíz llamada `“.vscode”`: si la abres, puedes ver que dentro hay un fichero llamado `tasks.json` (se trata de un fichero creado por VSCode y que contiene la configuración necesaria para lanzar el proceso de ejecutar el depurador).

### 5.5.3.- Agregar la configuración de depuración

---

En el apartado anterior hemos visto que, cuando se lanza la ejecución del programa desde las opciones del menú *Ejecutar* de VSCode, se crea el fichero `tasks.json`. Es posible y preferible crear una configuración de depuración más completa para nuestros programas.

Para ello, abre el fichero del programa en el editor de texto y pincha con el botón derecho del ratón. Aparecerá un menú flotante en el que debes elegir la opción *Agregar configuración de depuración*. Como resultado, dentro de la carpeta `“.vscode”` se crearán dos ficheros: `tasks.json` y `launch.json`.

El fichero `tasks.json` es el mismo que se generó en el apartado anterior y proporciona la forma de compilar el programa. El fichero `launch.json` permite personalizar la forma en la que se ejecuta el programa.

Puedes abrir ambos ficheros en el editor y estudiar su contenido. Puede que no comprendas la mayoría de las instrucciones, pero te darán algunas pistas de cómo hace su trabajo VSCode.

Una vez que se han creado los ficheros para un programa, cada vez que utilices las opciones del menú *Ejecutar*, VSCode los utilizará para compilar y ejecutar cualquier programa que generes dentro de un proyecto que cuelgue de la carpeta raíz.

### 5.5.4.- Ejecución en un terminal externo

---

Ahora, vamos a ejecutar el programa *“Hola, Mundo”* desde las opciones del menú de VSCode, pero, en vez de ejecutarlo en el terminal integrado de VSCode, vamos a utilizar un terminal externo (en Windows, su consola).

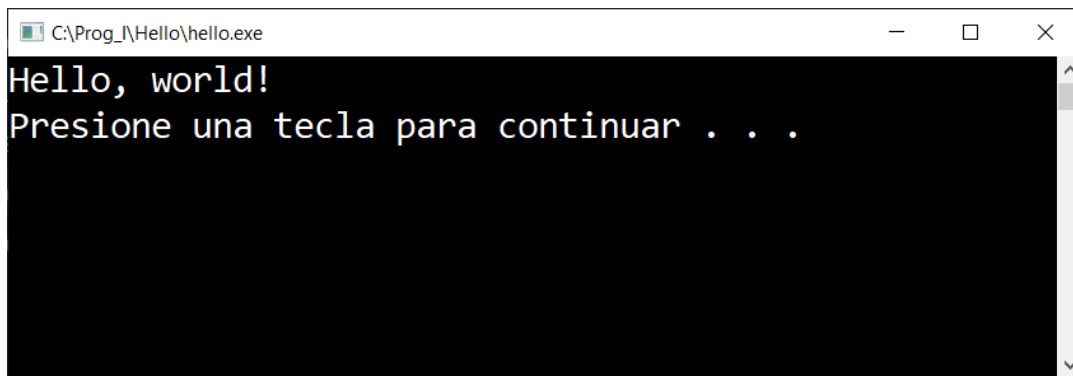
Para ello, abre el fichero `launch.json` en el editor de texto. Observa que hay una línea que dice lo siguiente:

```
"externalConsole": false,
```

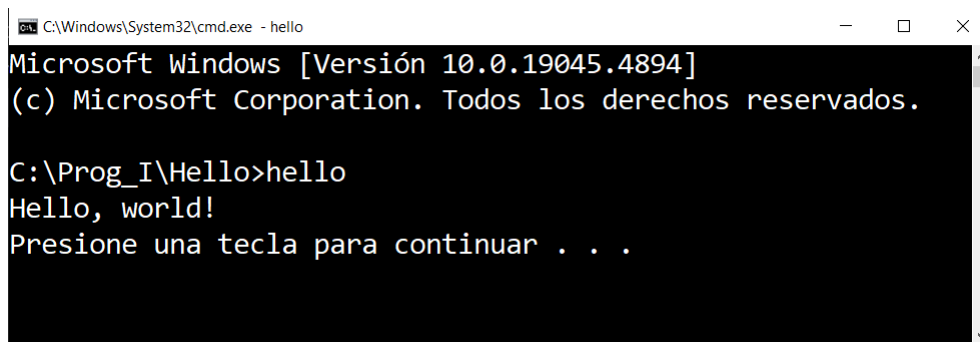
Sustituye la palabra `false` por `true` y guarda el fichero con `CTRL+S`. ¡Ojo, no borres la coma que hay al final de la línea!

Abre el programa en el editor y ejecútalo sin depuración, utilizando `CTRL+F5` (alternativamente puedes utilizar la opción de menú *Ejecutar* -> *Ejecutar sin depuración*).

Observa que ahora la ejecución del programa se lleva a cabo en un terminal externo, no en el terminal integrado en VSCode. Igual que en las ocasiones anteriores, el programa imprimirá el mensaje *“Hello, world!”* y quedará detenido esperando la pulsación de una tecla o de `INTRO` para finalizar y devolver el control a VSCode. Si lo pruebas, el resultado debería ser similar al de la siguiente figura, que muestra la salida en la versión de código para Windows:



Otra forma de abrir un terminal externo es, situados dentro del editor de texto de VSCode, pulsar la combinación de teclas `CTRL+SHIFT+C`. Este atajo de teclado abrirá el terminal con el directorio raíz del proyecto seleccionado. Una vez abierto el terminal, podemos teclear el nombre del programa para ejecutarlo, como se hace en la siguiente figura:



A partir de ahora, cuando quieras ejecutar tus programas, puedes elegir entre ejecutarlos en el terminal integrado de VSCode o en un terminal externo. Según la circunstancia y el momento del desarrollo del programa, puede ser más interesante una u otra opción. En cualquier caso, ya sabes cómo hacerlo: simplemente hay que cambiar el valor de la línea `externalConsole` del fichero `launch.json`.

### NOTA: Directorio de trabajo

Cuando, estando en el terminal, tecleamos el nombre de un programa para que se ejecute, es necesario que el programa esté en el directorio de trabajo del terminal.

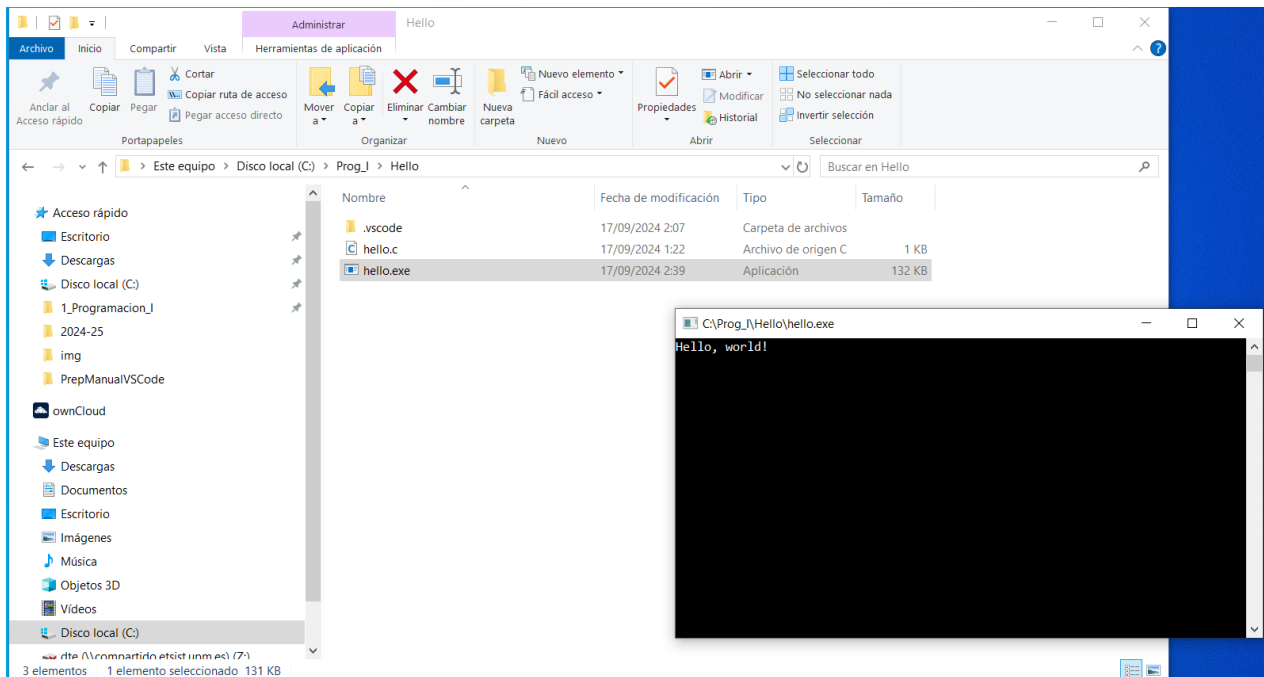
Los métodos que se han indicado para abrir el terminal integrado de VSCode o el terminal externo, normalmente nos dejarán situados en el directorio raíz del proyecto. Si el programa que queremos ejecutar está en otro directorio, habrá que situarse en él antes de poder ejecutarlo tecleando su nombre.

Esto se puede conseguir mediante instrucciones `cd` (*change directory*), como se explica en el manual de uso de la consola que tenéis a vuestra disposición en el Moodle de la asignatura.

### 5.5.5.- Ejecución desde el explorador de archivos del sistema operativo

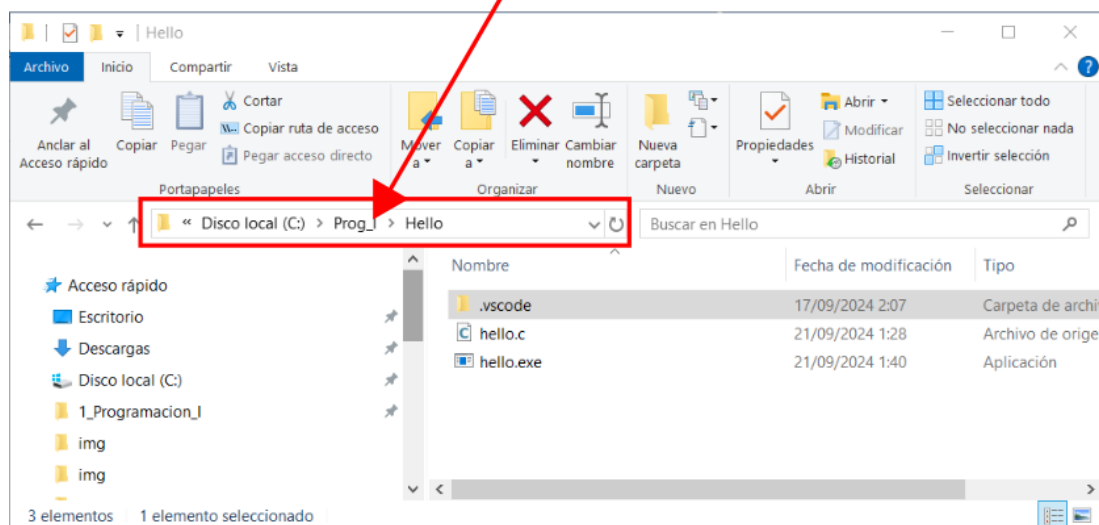
Una vez que compilemos nuestros programas y obtengamos los ejecutables, también los podemos ejecutar desde el explorador de archivos del sistema operativo. Para ello, abre el explorador de archivos, navega hasta la carpeta en la que está el ejecutable y haz doble click con el ratón sobre él. El resultado debería ser similar

al del apartado anterior: se abrirá un terminal, se imprimirá “Hello, world!”, y se quedará esperando a que pulsemos una tecla o INTRO para finalizar. La siguiente figura muestra el resultado:



El explorador de archivos de Windows nos ofrece la posibilidad de abrir un terminal posicionado en un directorio concreto, para poder ejecutar los programas que se encuentren en ese directorio, tecleando su nombre. Para ello, hay que teclear la palabra `cmd` en la barra de ruta del explorador. El resultado será que obtendremos un terminal abierto en el directorio que tuviera seleccionado en ese momento el explorador. La siguiente figura muestra dónde teclear `cmd`:

**Teclea `cmd` aquí y obtendras un terminal abierto en este directorio**



## ANEXO 1.- Desinstalación de MinGW

Si seguiste el curso 2023-24, es posible que tengas instalado el compilador MinGW, que es el que utilizábamos entonces. Puedes seguir utilizándolo, si lo deseas, pero si prefieres instalar el compilador que proponemos en este curso, es mejor que desinstales previamente MinGW. Para ello sigue la siguiente secuencia:

1. Borra completamente la carpeta donde está instalado MinGW. En el caso de que lo hayas instalado en un ordenador Windows, es probable que dicha carpeta sea `C:\MinGW`.
2. Si estás en un sistema Windows, tienes que borrar también las dos rutas de directorios que añadíamos a la variable *path* del sistema: `C:\MinGW\bin` y `C:\MinGW\msys\1.0\bin`  
Para borrar estas rutas, edita la variable *path* como se explica en el apartado 3 de este manual y elimina los dos directorios mencionados.

**¡¡Ten cuidado de no borrar la variable *path* completa!! Sólo hay que editarla.**

## ANEXO 2.- intérpretes de instrucciones

El sistema operativo es una colección de programas que permiten al usuario interactuar con todos los dispositivos que componen un ordenador.

Originalmente, la interacción se hacía a través de un programa que se denominaba la consola, también llamada terminal. Genéricamente, a estos programas se les llama *Command-Line Interface* (CLI) o también el *Shell* del ordenador. En español, es habitual llamar a estos programas: el *terminal*, la *consola* o el *intérprete de comandos* o *instrucciones*. En inglés, *Command Prompts* o *Shells*.

En los años 80 del siglo pasado, la empresa Apple Inc. desarrolló el *Graphic User Interface* (GUI) para su popular ordenador Macintosh. En poco tiempo, todos los sistemas operativos incorporaron sus propios GUIs. En concreto, Microsoft presentó la primera versión del sistema operativo Windows.

En los GUI, los directorios de los discos se representaban por la imagen de una carpeta. Por ello, a día de hoy, las agrupaciones de archivos en los discos se denominan carpetas o directorios, de manera indistinta.

A pesar de que muchas tareas del ordenador se pueden realizar a través del GUI, sigue habiendo tareas que son más fáciles de realizar utilizando la consola. Incluso, hay tareas que sólo se pueden realizar utilizando la consola.

Todos los sistemas operativos ofrecen una o más consolas para interactuar con él. Además, estas consolas disponen de un lenguaje de programación que permite automatizar las tareas que se quieren realizar. A los programas escritos en dicho lenguaje se les denomina *batch scripts*. En Windows, los programas escritos para la consola *Command Prompt* tienen la extensión *.bat*. En Linux o Mac no necesitan ninguna extensión concreta, aunque es habitual que tengan la extensión *.sh*.

En Linux y Mac, las consolas más populares son *Bash* y *Z-shell*. En Windows es el denominado *Command Prompt*, aunque las últimas versiones de Windows ofrecen también la consola *Power Shell*, un terminal basado en su versión equivalente de Linux y que ofrece funcionalidades más avanzadas.

Cuando se está programando, es imprescindible recurrir a la consola frecuentemente, por lo que es conveniente aprender a utilizarla. De hecho, una vez que te acostumbras a usarla, verás que hay muchas tareas que son más fáciles de hacer con la consola y que tu productividad se incrementa notablemente.

A lo largo del curso utilizaremos frecuentemente la consola. En el caso de Windows, utilizaremos el *Command Prompt*. En el Moodle de la asignatura dispones de un pequeño manual de uso con las instrucciones más frecuentes. En Linux o Mac puedes utilizar cualquiera de las consolas que ofrezcan.

## A2.1.- El terminal integrado en VSCode

VSCode también ofrece una consola integrada. Entre las posibles opciones se encuentran las propias del sistema operativo. En el caso de Windows, en el apartado 4.3, se ha explicado como configurar VSCode para que utilice por defecto la consola *Command Prompt*.

Puedes abrir el terminal integrado y ejecutar una serie de instrucciones. Se dice que estás en una *sesión de terminal*. Puedes cerrar el terminal y luego volver a abrirlo. Los valores de la sesión se conservan y podrás ver los mensajes que hubiera en el terminal, tal y como los dejaste cuando lo cerraste. También puedes abrir nuevas sesiones de terminal y mantener varias sesiones abiertas al mismo tiempo.

Para abrir el área de terminales de la parte inferior de VSCode puedes usar la opción de menú `Ver -> Terminal`. Ahí podrás ver las sesiones de terminal que tengas abiertas. Para crear una nueva sesión de terminal, puedes usar la opción de menú `Terminal -> Nuevo terminal` o los controles que se ofrecen en el área de terminales.

Si cierras el área de terminales, no se destruyen las sesiones que tengas activas, solo se dejan de visualizar en la parte inferior de la ventana. Para destruir una sesión de terminal, tienes que utilizar los controles que ofrece el área de terminales.

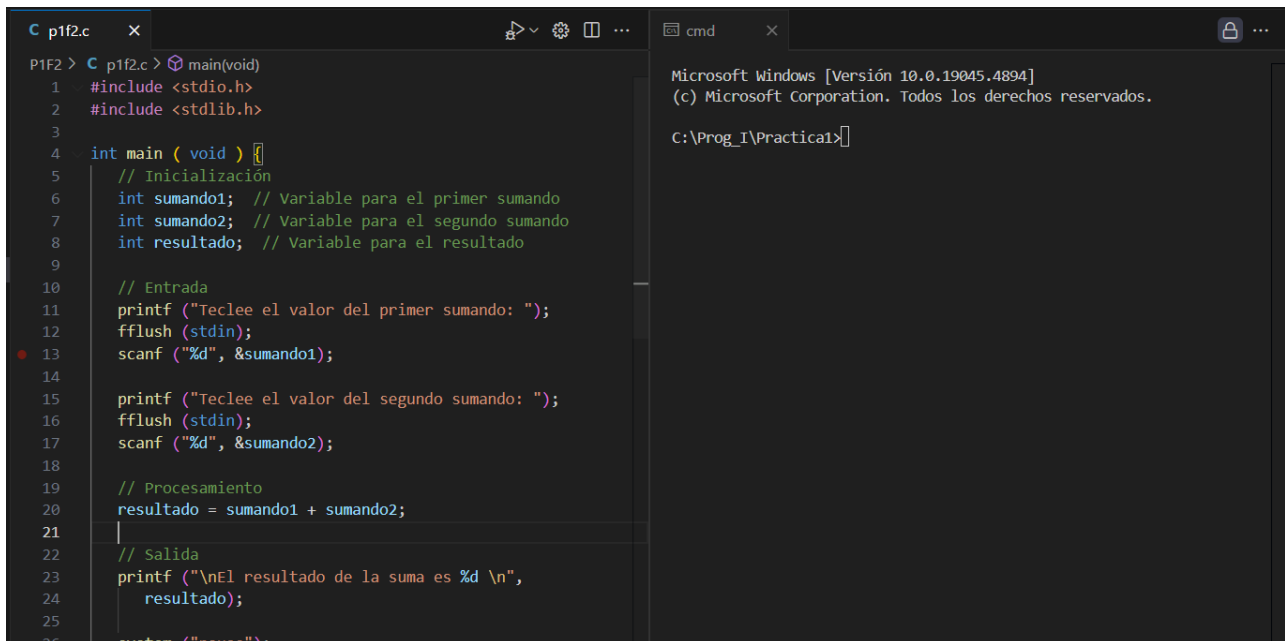
La figura siguiente muestra el área de terminales de VSCode, con dos sesiones abiertas e indicación de los controles para abrir y cerrar.





En VSCode, cuando nos encontramos en el editor de texto (pinchando con el ratón en el editor, éste adquiere el foco de las entradas de teclado), puedes abrir el área de terminales utilizando el atajo de teclado que, en Windows y con teclados en español, es **CTRL+Ñ**, en Linux **CTRL+`** y en Mac **^`**. Para crear nuevas sesiones se utilizan los mismos atajos de teclado, pero añadiendo la tecla **SHIFT**.

También es útil visualizar una sesión de terminal en la zona del editor de texto. Para ello, pincha y arrastra el terminal correspondiente desde la zona de sesiones activas hasta la zona de pestañas del editor. La Figura siguiente muestra la zona del editor de VSCode con dos paneles, uno para el código fuente y otro para el terminal.



The image shows the Visual Studio Code interface with two panels. The left panel displays a C program named `p1f2.c` with the following code:

```
P1F2 > C p1f2.c > main(void)
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main ( void ) {
5      // Inicialización
6      int sumando1; // Variable para el primer sumando
7      int sumando2; // Variable para el segundo sumando
8      int resultado; // Variable para el resultado
9
10     // Entrada
11     printf ("Teclee el valor del primer sumando: ");
12     fflush (stdin);
13     scanf ("%d", &sumando1);
14
15     printf ("Teclee el valor del segundo sumando: ");
16     fflush (stdin);
17     scanf ("%d", &sumando2);
18
19     // Procesamiento
20     resultado = sumando1 + sumando2;
21
22     // Salida
23     printf ("\nEl resultado de la suma es %d \n",
24             resultado);
25 }
```

The right panel shows a terminal window titled `cmd` with the following text:

```
Microsoft Windows [Versión 10.0.19045.4894]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Prog_I\Practica1>
```