

INTRODUCCIÓN AL LENGUAJE C

ACERCA DE C

C es un lenguaje de alto nivel, con las grandes ventajas de racionalidad, estructuración y relativa facilidad de uso que ello supone; pero a su vez C nos permite también trabajar con bits, registros de CPU o posiciones de memoria, por lo que sin tener que utilizar el críptico y tedioso lenguaje máquina, con C podemos acceder directamente al hardware de nuestro ordenador.

Por ejemplo, una prueba tangible de lo mencionado son los sistemas operativos UNIX, Windows y OS/2 están escritos en C.

LOS ENTORNOS DE PROGRAMACIÓN EN C

Una vez que vaya dominando este lenguaje, podrá escribir, compilar y ejecutar programas C bajo múltiples entornos de programación y sistemas operativos. En realidad, cualquier editor ASCII sirve para escribir programas C y prácticamente todos los sistemas operativos poseen algún compilador C.

Sin embargo, el programar no consiste únicamente en escribir programas; éstos deben ser depurados de sus errores, ejecutados en situaciones controladas para comprobar su corrección, integrados en otros programas más complejos o, como condiciones sobre la memoria del ordenador donde se ejecutarán.

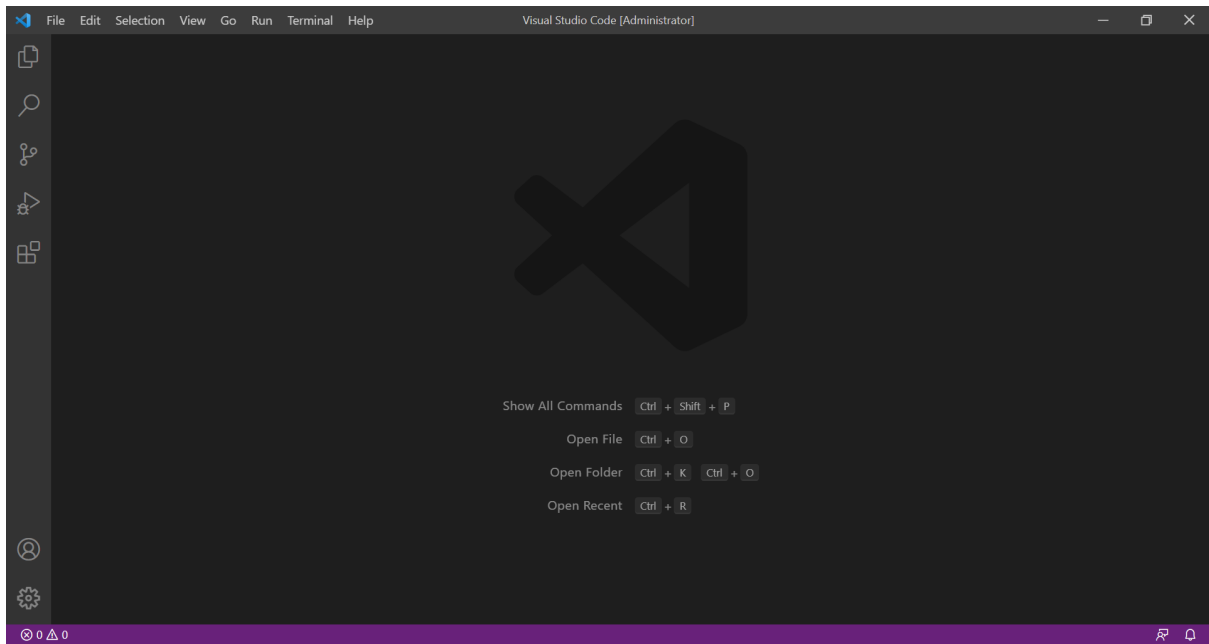
Es aquí donde se aprecia la importancia de poseer un buen entorno de programación. Ésta es una tarea creativa y estimulante, pero también difícil, trabajosa y sometida a criterios de productividad, sobre todo en la programación profesional. Un buen entorno no elimina estas circunstancias, pero sí proporciona herramientas que acentúan los aspectos positivos de la programación y minimiza los negativos.

La herramienta con la que trabajaremos es Visual Studio Code, uno de los más utilizados dentro de la programación profesional.

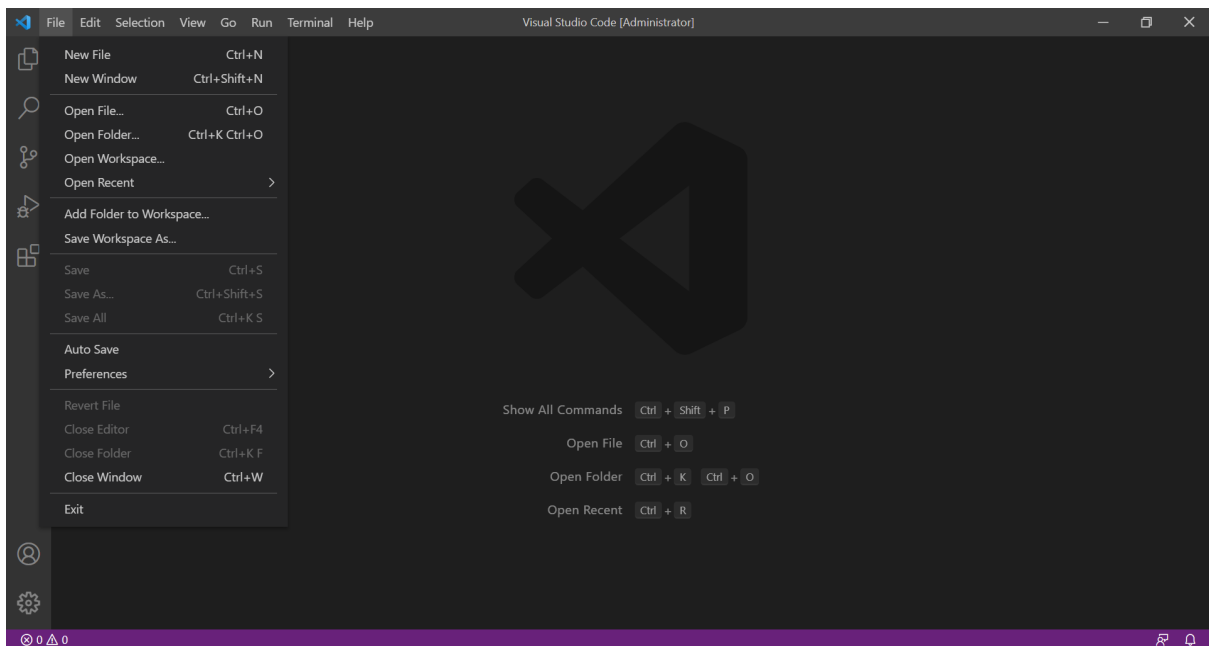
EDICIÓN DE PROGRAMAS

Si usted acaba de arrancar Visual Studio Code por primera vez, seguramente la pantalla luzca como la figura 1.

Laboratorio de Computación I

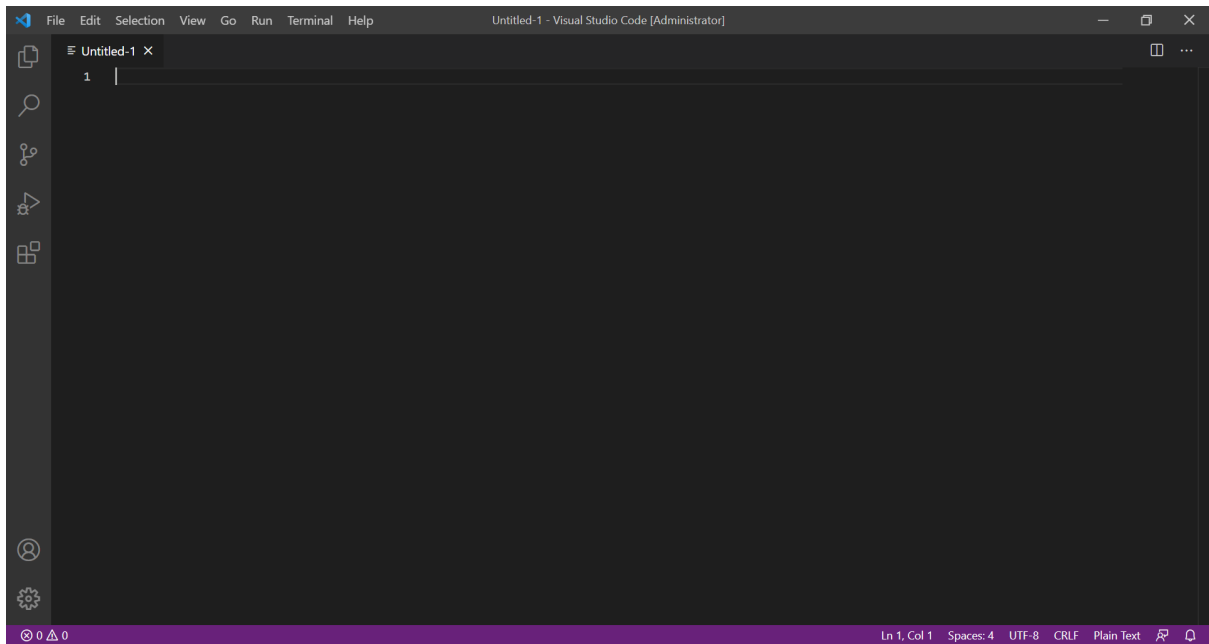


Para comenzar a trabajar, presione File que aparece a la izquierda de la barra de menús, aparecerán las siguientes opciones como se muestra en figura 2.

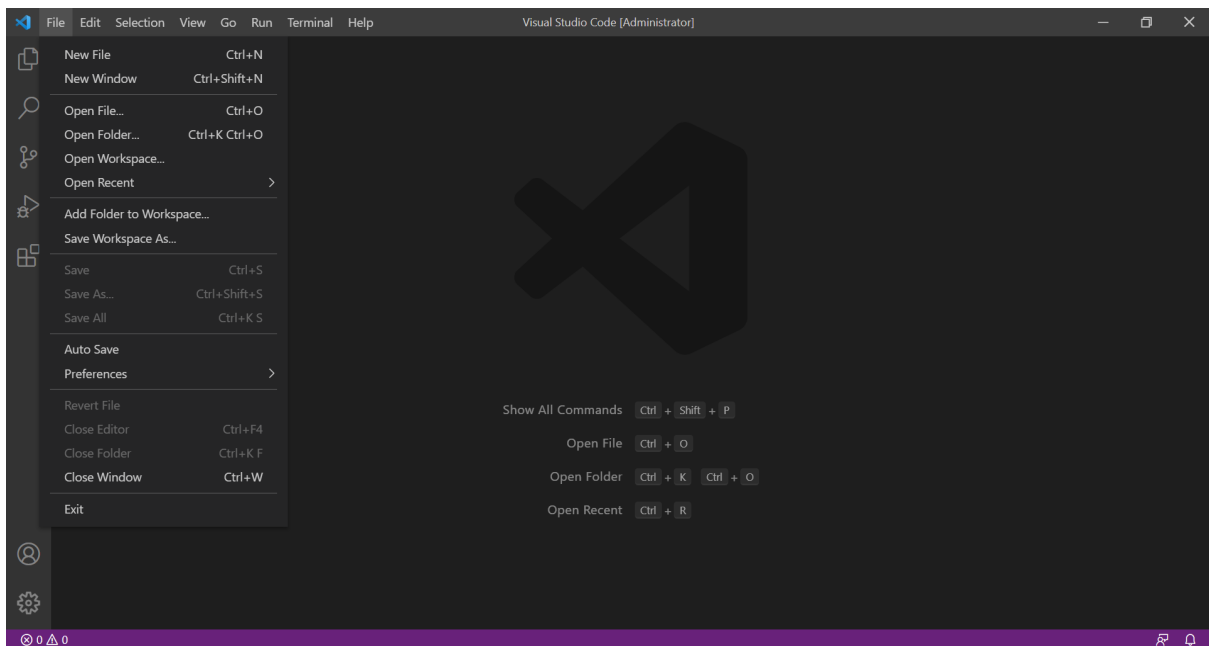


Presione New File y verá cómo aparece su documento en blanco como muestra la figura 3.

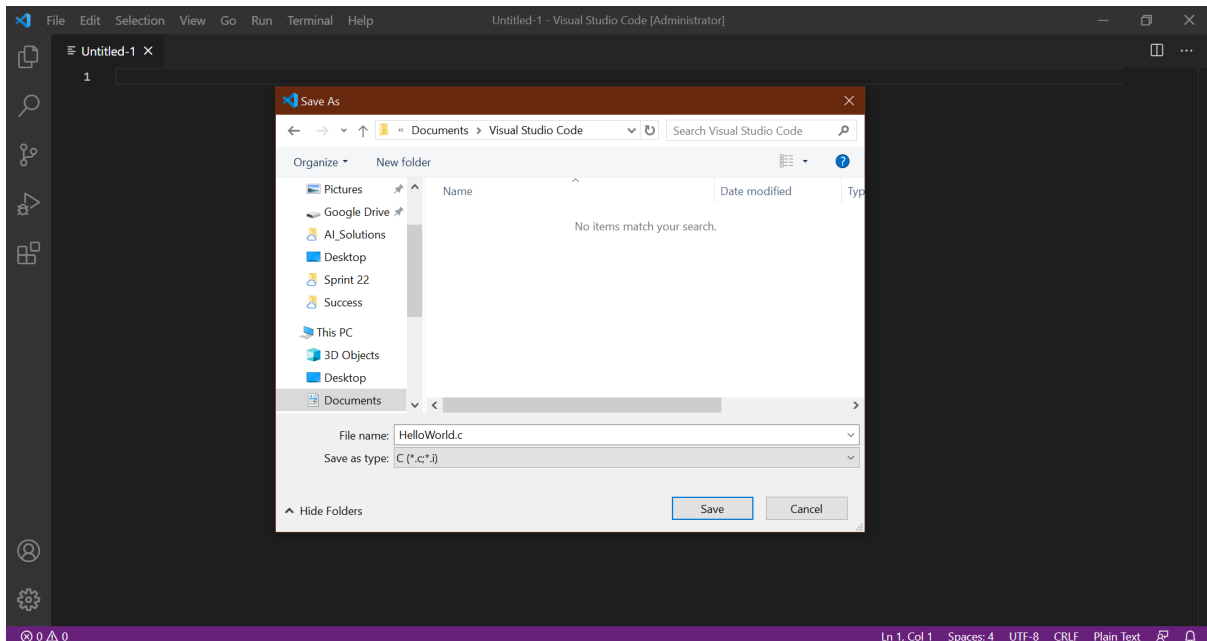
Laboratorio de Computación I



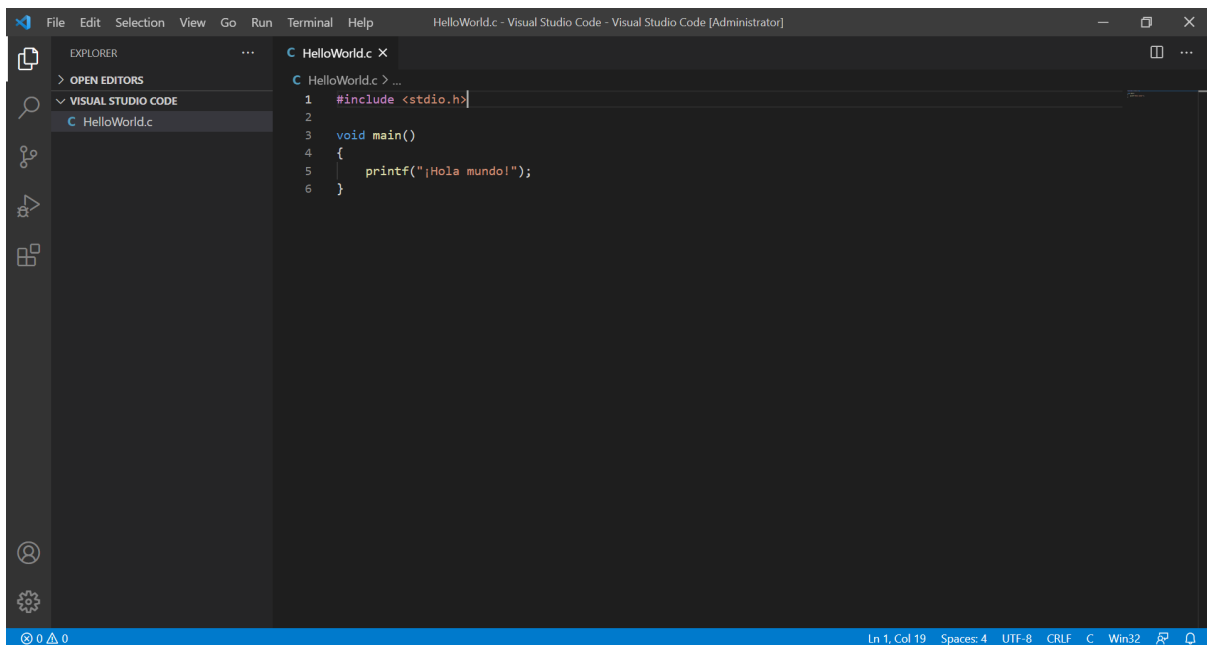
Presione nuevamente en File y luego en Save As para nombrar el archivo y elegir su ubicación en el disco duro, como se indica en la figura 4.



Elija la ubicación deseada, defina el nombre del archivo y tenga en cuenta seleccionar la extensión .c, como se muestra a continuación en la figura 5.



Ahora usted se encuentra en condiciones de editar su primer programa. La pantalla se indica en la figura 6.



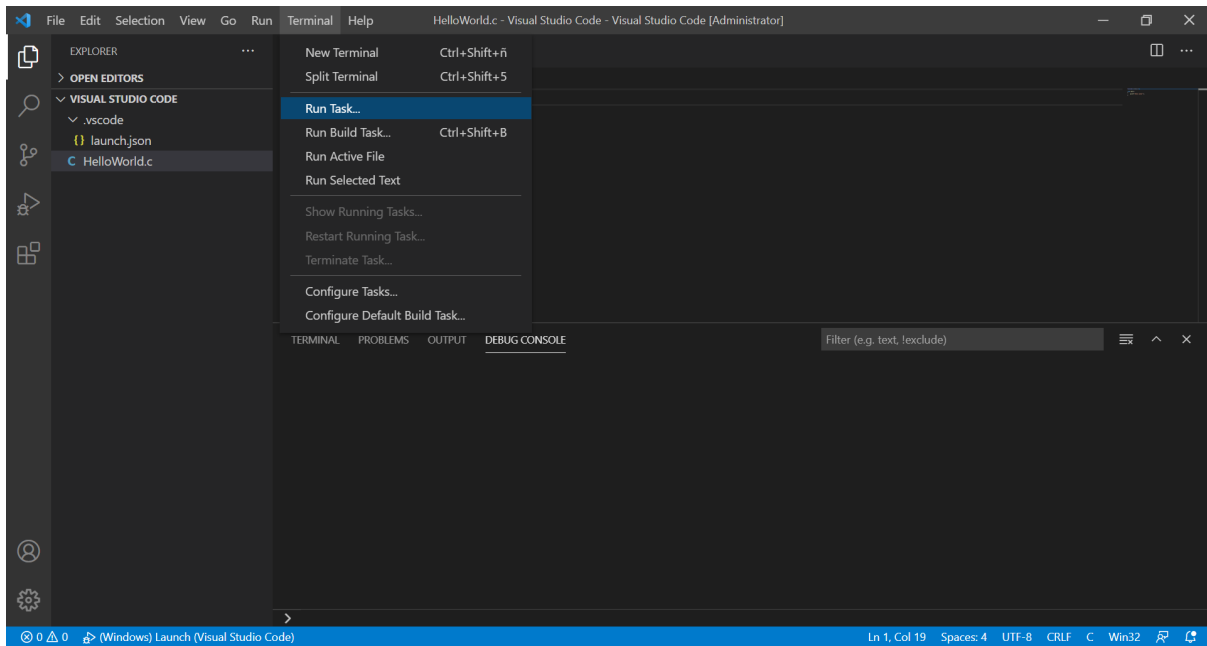
Recuerde guardar periódicamente los cambios a medida que va trabajando.

COMPILACIÓN Y EJECUCIÓN DE PROGRAMAS

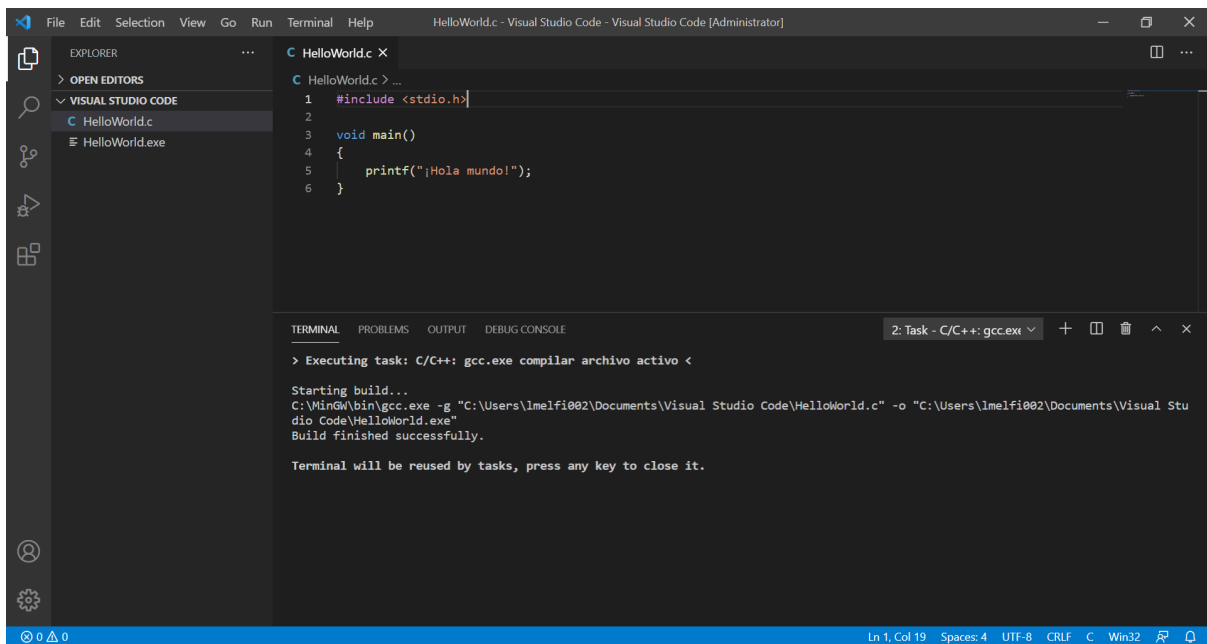
Una vez que hayamos completado la edición de un programa, nuestro deseo será probablemente ejecutarlo. Para ello es preciso compilarlo previamente para construir su módulo objeto y a continuación enlazarlo con las bibliotecas estándar de C.

Para compilar un programa presionamos Terminal y elegimos la opción Run Task Como muestra la figura 7.

Laboratorio de Computación I

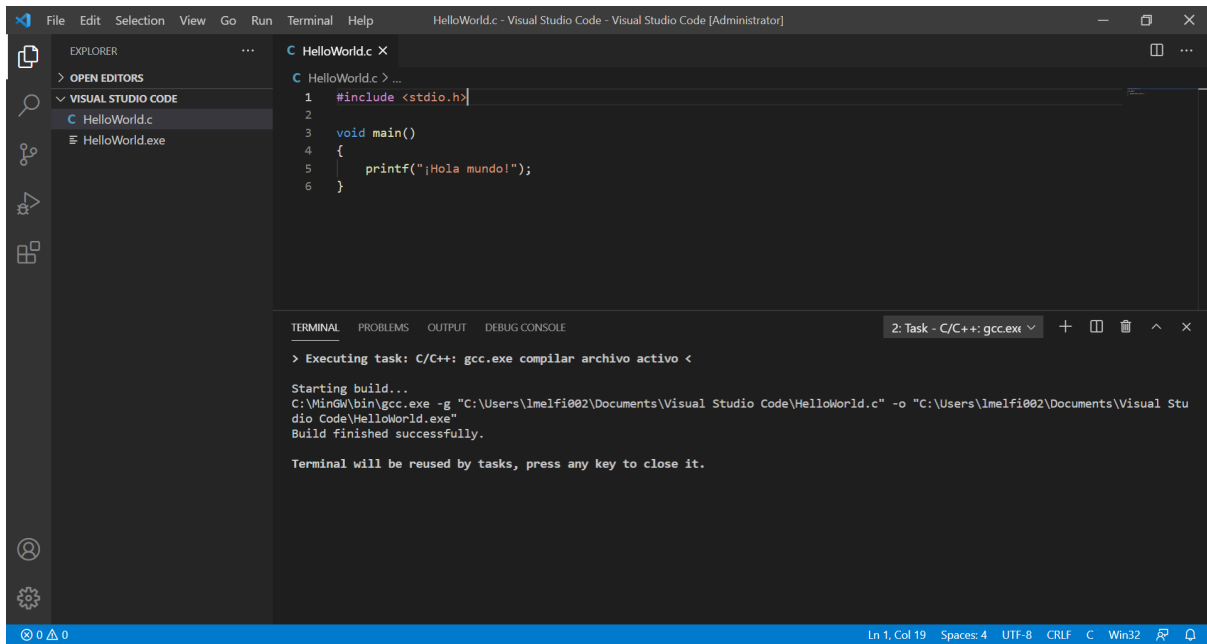


Una vez compilado obtendremos el ejecutable de nuestro programa como se indica en la figura 8.



EL PRIMER PROGRAMA

El primer ejemplo de programa en C, es uno que imprime el mensaje "¡Hola mundo!". Pese a sus pocas líneas contiene los elementos esenciales de todo programa C, como se muestra en la figura 9.



The screenshot shows the Visual Studio Code interface. The Explorer panel on the left shows the file structure with 'HelloWorld.c' and 'HelloWorld.exe'. The main editor displays the code for 'HelloWorld.c':

```
1 #include <stdio.h>
2
3 void main()
4 {
5     printf("¡Hola mundo!");
6 }
```

The TERMINAL panel at the bottom shows the execution of the task 'C/C++: gcc.exe compiler archivo activo <'. The output indicates a successful build:

```
> Executing task: C/C++: gcc.exe compiler archivo activo <
Starting build...
C:\WinGW\bin\gcc.exe -g "C:\Users\lmelfi002\Documents\Visual Studio Code\HelloWorld.c" -o "C:\Users\lmelfi002\Documents\Visual Studio Code\HelloWorld.exe"
Build finished successfully.
Terminal will be reused by tasks, press any key to close it.
```

Las funciones de entrada/salida son aquellas que transportan datos entre el programa y la entrada y salida estándar (teclado y pantalla respectivamente). La entrada/salida no forma parte del lenguaje C, sino que se proporciona a través de una biblioteca de funciones. La biblioteca o librería donde se encuentran estas funciones es `<stdio.h>`. Siempre que queramos utilizar funciones de entrada/salida, deberemos incluir al comienzo de nuestro programa la línea:

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

A continuación aparece la siguiente línea:

```
void main()
```

Fijémonos primero en **main**, que es el nombre de una función. Una función es un cierto conjunto autónomo de líneas de código, diseñadas generalmente para realizar una tarea concreta. Posteriormente estudiaremos a fondo el uso de funciones en C, pero le recordamos que las funciones suelen recibir argumentos, esto es, datos a procesar, y que, como fruto de este proceso, también devuelven frecuentemente otros datos. La línea anterior especifica que **main** no devuelve nada paréntesis (este es el efecto de la palabra **void**, que podríamos traducir aquí como “vacío”) y que tampoco recibe ningún argumento: estos deberían estar entre los dos paréntesis que siguen a **main**, como vemos, no hay nada

entre ellos. En el caso de **main**, y en el de cualquier otra función C, su cuerpo está delimitado entre las llaves { y }, donde se especifica cuál va hacer su tarea concreta.

Nuestro **main** contiene una única línea:

```
printf("¡Hola mundo!");
```

La función de nombre **printf** tiene como objetivo mostrar en pantalla cierta información, que en nuestro caso viene dada como cadenas de caracteres, pero que puede ser distinta en otras ocasiones como tendremos oportunidad de ver más adelante. Aquí printf recibe como argumento lo que hay entre sus paréntesis, la cadena de caracteres "¡Hola mundo!" donde las comillas son justamente una indicación al compilador de que lo que encierran es precisamente una cadena.

TIPOS DE DATOS EN C

Todo lenguaje posee un cierto conjunto de datos básicos predefinidos: vienen a ser los ladrillos que utilizaremos en el trabajo con datos. Pero, estos datos predefinidos son a menudo insuficientes para abordar tareas de programación medianamente complejas. Todos los lenguajes de programación ofrecen diversos mecanismos de construcción de datos compuestos a partir de los predefinidos.

En concreto, C nos permite combinarlos en **tablas** y **estructuras** (*arrays* y *structs* en C, respectivamente).

A continuación, se detalla los tipos de datos básicos:

DATO	DESCRIPCIÓN	TAMAÑO	RANGO
int	Número entero	2 bytes	-32.768 a 32.767
char	Carácter	1 byte	-128 a 127
float	Número en punto flotante	4 bytes	3.4×10^{-38} a 3.4×10^{38}
double	Número en punto flotante	8 bytes	1.7×10^{-308} a 1.7×10^{308}

Ejemplo 1: definimos una variable de tipo entero con nombre numero.

```
int numero;
```

Ejemplo 2: se definen las variables a, b y c como números de tipo flotante.

```
float a,b,c;
```

VARIANTES DE LOS TIPOS CLÁSICOS

A continuación se detallan las variantes de los tipos clásicos de datos:

DATO	DESCRIPCIÓN	TAMAÑO	RANGO
short	Número entero	2 bytes	-32.768 a 32.767
long	Número entero	4 bytes	-2.147.483.648 a 2.147.483.647
unsigned	Número entero positivo	2 bytes	0 a 65.535
unsigned short	Número entero positivo	2 bytes	0 a 65.535
unsigned long	Número entero positivo	4 bytes	0 a 4.294.967.295
unsigned char	Carácter positivo (ASCII)	1 byte	0 a 255
long double	Número en punto flotante	10 bytes	$3.4 \times 10^{-4.932}$ a $3.4 \times 10^{4.932}$

Por ejemplo, pueden usarse:

```
short int a;
long int b;
```

o abreviarse a:

```
short a;
long b;
```

Otros ejemplos de este estilo los tenemos en las definiciones:

```
signed long c;
unsigned short d;
signed e;
unsigned f;
```

donde en las dos últimas líneas se entiende que estamos definiendo datos **int** con signo y sin signo respectivamente.

Por su parte, las letras U y L situada la final de las constantes:

1234L

2345U

3456UL

identifican a **1234** como un **long int**, a **2345** como **unsigned** y a **3456** como **unsigned long**.

EXPRESIONES ARITMÉTICAS

En una expresión aritmética aparecen una serie de constantes o variables numéricas, esto es de los tipos int, float, double o sus variantes, junto con los denominados operadores aritméticos. Seguido se detallan los operadores aritméticos de C:

OPERADOR	OPERACIÓN
+	Suma
-	Resta
*	Multiplicación
/	División
%	Resto de división
-(monario)	Cambio de signo
++	Incremento
--	Decremento

La preferencia y asociatividad de los operadores aritméticos es la siguiente:

PREFERENCIA	OPERADOR	ASOCIATIVIDAD
1	++, --, - (monario)	Derecha a izquierda
2	*, /, %	Izquierda a derecha
3	+, -	Izquierda a derecha

USO DE PARÉNTESIS

C usa dos mecanismos para ordenar la ejecución de una expresión aritmética: la **precedencia** de operadores, y la **asociatividad** entre operadores de igual precedencia. Por ejemplo, $*$, $/$ y $\%$ tienen mayor precedencia que $+$ y $-$. Así, en la expresión $12/3/4*2+1*2$, como en cualquier otra expresión en C, se ejecutan antes los operadores de mayor precedencia; por tanto, antes de ejecutarse la suma, se han de ejecutar los cocientes y los productos. El orden es de izquierda a derecha, por ejemplo en la expresión $12/3/4*2$, primero se divide el 12 por 3; su resultado, 4, se divide entonces por 4, dando 1, que se multiplica finalmente por 2. En otras palabras, usando paréntesis, dicha expresión equivale a $((12/3)/4)*2$. Si se desea cambiar la precedencia de la operación, se utilizan paréntesis, por ejemplo $(12/3)/(4*2)$ divide 4 por 8 que serían los resultados de 12 dividido 3 y 4 multiplicado por 2 respectivamente.

EXPRESIONES RELACIONALES Y LÓGICAS

Las otras dos familias son las expresiones **relacionales** y las expresiones **lógicas**. Los operadores de relación (o relacionales) son aquellos empleados para comparar dos valores numéricos, es decir, los correspondientes a las preguntas tales, como “¿A es mayor que B? o ¿C es igual a D?”. Los símbolos con que estos operadores se representan son muy similares a los usados habitualmente en matemáticas. Una expresión relacional es una expresión en las que otras expresiones aritméticas se combinan mediante operadores relacionales. Veamos un ejemplo concreto, $(12/3/4*2+1*2) >= (5*3*2)$ en la que estamos preguntando si la expresión de la izquierda es mayor o igual que la transcrita a la derecha. El resultado es que no, ya que el valor de la izquierda es 4, que es obviamente menor que el de la derecha 30. Claramente, el resultado no es un dato aritmético, es un dato lógico. Observe que un dato lógico toma dos valores, **Verdadero** o **Falso**. En C podemos llamarlos True y False respectivamente. También podríamos representarlos con 1 y 0 respectivamente.

	OPERADOR	OPERACIÓN
RELACIONALES	$>=, <=$	Mayor o igual, Menor o igual
	$>, <$	Mayor, Menor
	$==$	Igual
	$!=$	Distinto

LÓGICAS	&&	Y lógico
		O lógico
	!	Negación lógica

El uso de paréntesis es similar al descrito en la sección anterior.