

Contenido

- 2. Primeros pasos
 - 2.1 Primer programa
 - 2.2 Ejecutando nuestro programa
 - 2.3 Posibles errores

2. PRIMEROS PASOS.

Nuestras primeras acciones serán aprender un poco sobre la interfaz que **CodeBlocks** nos presenta. Ayudándonos mediante un código sencillo y mejor conocido como el “**Hola mundo**”. Este código es uno de los más sencillos que podemos hacer, ya que no requiere demasiadas instrucciones, solo las necesarias.

Será nuestra primera versión de un código en lenguaje C/C++. Aunque en esta sección no explicaremos tan detalladamente este famoso código, nos ayudará a familiarizarnos con algunos tipos de instrucciones que podemos utilizar en el lenguaje.

Es recomendable que si el lector es un nuevo en este tema, lea detenidamente en las partes donde crea que debe hacerlo. Siempre podemos volver a leer un párrafo para comprenderlo mejor. Dedicar un tiempo para tratar de entender que es lo que sucede, también es una buena sugerencia.

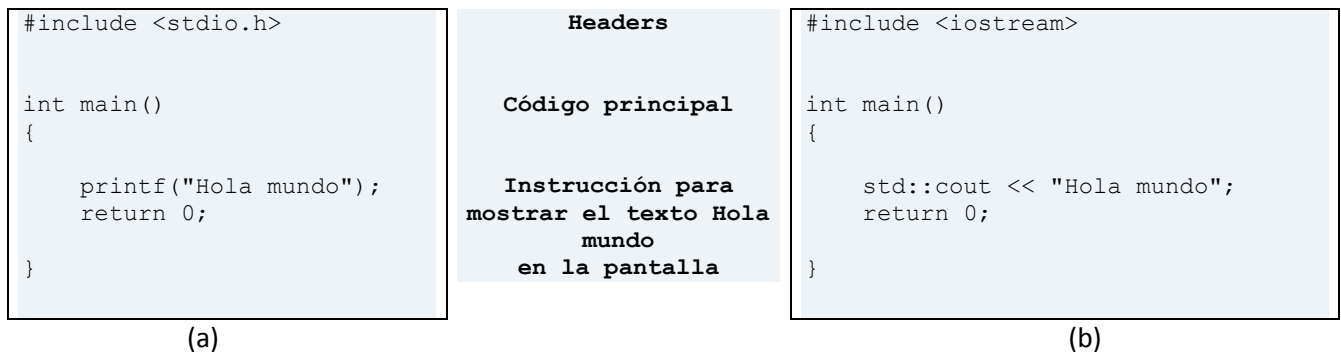
2.1 Primer programa.

Un programa o código en lenguaje C/C++ es un conjunto de instrucciones, en **formato de texto**, dadas en secuencia para realizar una tarea específica. Para comenzar, podemos hacer notar algunas partes principales en un código:

Headers	<i>Sección que permite decirle al compilador que utilerías o herramientas vamos a utilizar, por ejemplo las más usadas son, utilerías para cálculos matemáticos, para mostrar información en la pantalla y leer datos desde el teclado.</i>
Variables globales	<i>En secciones posteriores, detallaremos el significado este término.</i>
Declaración de funciones	<i>Es el lugar donde podemos indicar las rutinas (creadas por nosotros mismos) que usaremos o que podemos usar en nuestro código.</i>
Código principal	<i>En esta sección incluiremos todo aquello que necesitamos que realice nuestro código.</i>
Definición de funciones	<i>Aquí escribimos explícitamente que hace cada rutina que se haya especificado en la sección Declaración de funciones.</i>

La razón por lo cual no definimos de manera completa los términos anteriores, es para no abordar con demasiada anticipación algunos temas que serán vistos con más detalles y ejemplos en el

futuro. Por ahora nos concentraremos en la sección **headers** y **código principal**. El **código 2.1** muestra el ejemplo de unos códigos más sencillo y conocidos usando el lenguaje C y C++. En él, se identifican algunas de las secciones antes mencionadas.



Código 2.1. (a) Código que corresponde a un código escrito en lenguaje C.
(b) Código escrito en lenguaje C++.

Como podemos observar, ambos códigos tienen la misma estructura, lo que debemos observar son los cambios en los **headers** y la **instrucción** para mostrar algo en pantalla. En el futuro veremos que, para nuestro propósito, esta sección y este tipo de instrucciones en particular, son la mayor diferencia entre ambos lenguajes. Escribamos entonces, nuestro primer código utilizando **CodeBlocks**. En la pantalla principal, podemos dirigirnos al menú **File -> New -> Empty File**, para crear un archivo vacío y comenzar a escribir nuestras instrucciones en este (ver **imagen 2.1**).

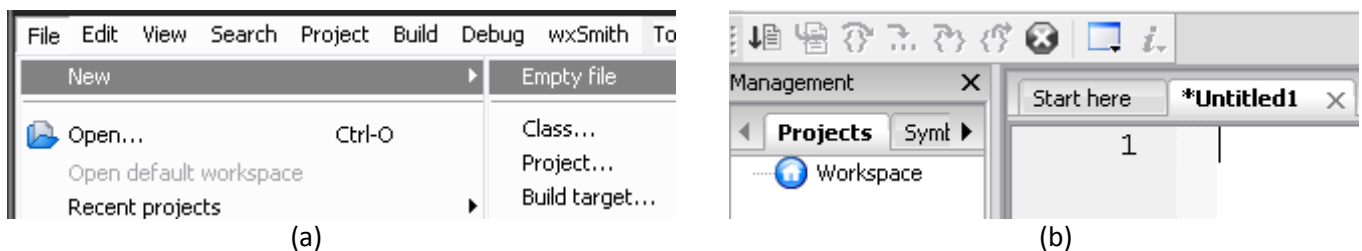


Imagen 2.1. (a) Elegiremos la opción **Empty File** cada vez que necesitemos escribir un código nuevo.
(b) Cuando se selecciona esta opción en la **sección de codificación** aparece una pestaña con el título **"*untitled1"**, que significa que es un nuevo archivo sin nombre.

Sin embargo, es recomendable que antes de comenzar a escribir cualquier instrucción, vayamos a la opción **File -> Save file** para guardar el archivo. Como en la mayoría de las aplicaciones de Windows, se abrirá un cuadro de diálogo para buscar la ruta donde se guardara el archivo nuevo. Una vez elegida la ruta, escribamos el nombre nuestro archivo. **Es importante asegurarse de que la extensión que debemos ponerle a nuestro archivo sea "cpp"**. En la **imagen 2.2**, mostramos que hemos guardado nuestro archivo como **main.cpp**.



Imagen 2.2. Sección inferior del cuadro de dialogo que muestra **CodeBlocks** para guardar un archivo.

Una vez elegido el nombre, el nombre en la nueva pestaña (recordar la **imagen 2.1.a**) cambiara por el que hayamos elegido. La razón por la cual guardamos el archivo antes y agregamos la extensión **cpp**, es para que **CodeBlocks** identifique que queremos escribir un código en C/C++ y nos permite ver colores dependiendo de qué escribamos (ver **imagen 2.3**).

```

1  #include <stdio.h>
2
3  int main()
4  {
5      printf("Hola mundo");
6      return 0;
7  }
```

(a)

```


1  #include <iostream>
2
3  int main()
4  {
5      std::cout << "Hola mundo";
6      return 0;
7  }
```

(b)

Imagen 2.3. Códigos con detalles en color para ambos lenguajes.

Es importante decidir en estos momentos, cuál será el tipo de código que escribiremos, esto es, decidir si escribir códigos en C o C++. En nuestra particular opinión, es mejor escribir los códigos en lenguaje C, pero como hemos dicho antes, la sintaxis es muy similar.

Una vez decidido el lenguaje, lo siguiente será escribir todo el texto correspondiente al **código 2.1**. Es recomendable escribir el texto para practicar nuestra escritura en el teclado y **no utilizar el famoso copy-paste**. También es importante mencionar, que debemos escribir los códigos tal cual los exponemos, usando letras minúsculas, ya que si usamos mayúsculas o las intercalamos entonces no funcionará.

Hecho lo anterior, debemos indicarle a **CodeBlocks** que deseamos ver el resultado final. Para conseguirlo, nos dirigimos al menú **Build -> Build and Run** (o buscar el icono  o presionar **F9**) para que **CodeBlocks** le diga al compilador (GNU GCC Compiler) que hemos escrito un programa y que necesitamos que realice una *comprobación* a nuestro código, para ver si esta correcto. Si hemos escrito el **código 2.1** correctamente, debe aparecernos en la **ventana de mensajes** (ver **imagen 2.4.a**).


Lo que hemos hecho al utilizar la opción **Build -> Build and Run** es conocido comúnmente en programación, como **compilar** nuestro código. La acción compilar consiste en indicarle a un compilador (en este caso GNU GCC Compiler) que un código necesita revisión. La revisión es el proceso que realiza el compilador para checar la sintaxis y encontrar posibles errores en nuestro código; por ejemplo, que escribimos mal una instrucción o nos faltaron algunos símbolos. Si hemos cometido algún tipo de error, al utilizar **Build -> Build and Run**, la **ventana de mensajes** nos lo hará saber, aunque no siempre es fácil entender cuál es el error, pero con la práctica será más fácil interpretar los mensajes y corregir nuestros errores.

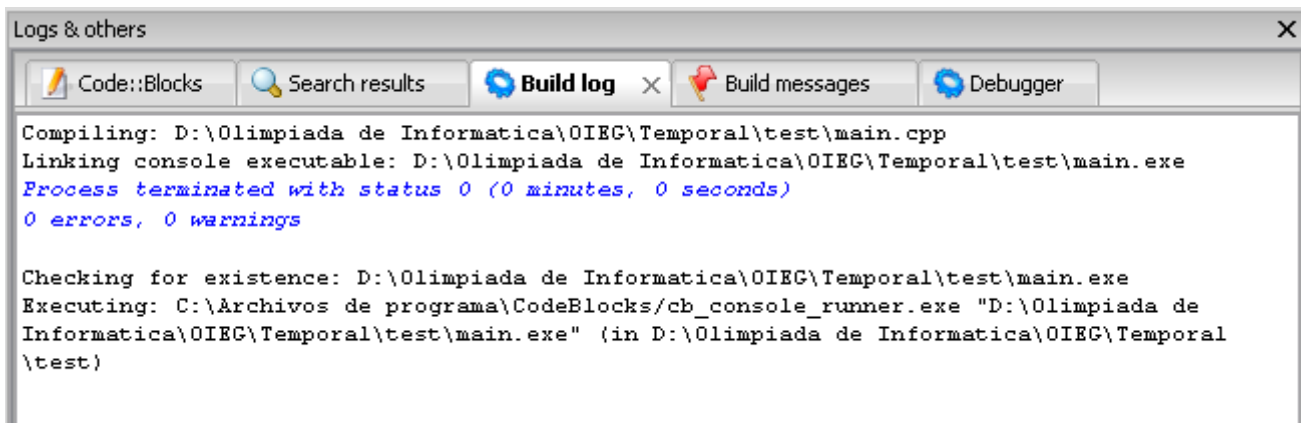
Ejercicio 2.1: Realiza una modificación o elimina algún símbolo para que observes como es que CodeBlocks te avisa que algo anda mal, recuerda regresarlo a su estado normal.

CodeBlocks es un editor que permite los comandos de control de texto que la mayoría de los editores: copiar, pegar, cortar, deshacer acciones y rehacer acciones, pruébalas.

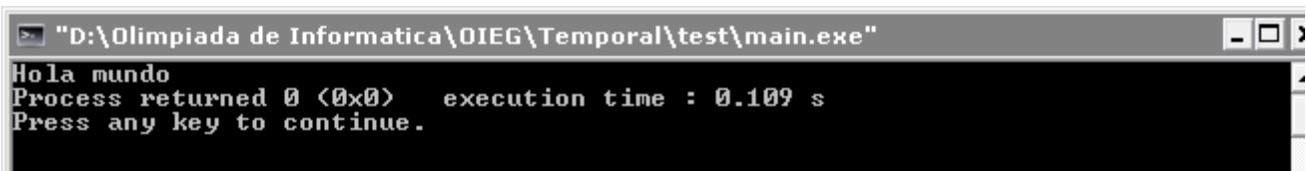
2.2 Ejecutando nuestro programa.

La **imagen 2.4.b** muestra una ventana con fondo negro que aparece cuando el proceso de compilación ha terminado y no hay error alguno. La pantalla negra es nuestro **programa ejecutándose**, esto quiere decir que el compilador ha creado un archivo con extensión **exe** (archivo que puede ejecutarse en Windows), que realizará las **mismas acciones** que le hemos dado, en cada ocasión que lo ejecutemos. Este archivo se crea en la misma carpeta donde guardamos nuestro código al principio.

Si al usar **Build -> Build and Run** todo salió bien, podemos usar la opción **Build -> Run** o **Ctrl + F10** o el botón  para ejecutar nuestro programa nuevamente, sin pasar por el proceso de compilado. Esto quiere decir, **que ejecutaremos el archivo exe que se ha generado desde la última vez que compilamos correctamente.**



(a)



(b)

Imagen 2.4. (a) Captura de pantalla después de **presionar F9**. Nos indica que todo es correcto y ahora está listo para probar el código.
(b) Si todo es correcto, al momento de **presionar F9** una ventana con fondo negro aparecerá con la palabra **Hola mundo**. Además de indicar el tiempo en segundos que necesitó nuestro programa para realizar las acciones pedidas.

Debemos exhibir una diferencia muy importante entre ejecutar nuestro programa usando **CodeBlocks** y ejecutarlo desde el **explorador de Windows**, como cuando ejecutamos cualquier otro programa o tratamos de abrir un archivo, por ejemplo Word, WMP, Firefox o Google Chrome (ver **imagen 5**). Cuando utilizamos la primera forma, al final de la pantalla negra aparece el mensaje **“Press any key to continue”** como puede observarse en la **imagen 2.4.b**; este mensaje significa que debemos presionar cualquier tecla (una letra, ENTER, espacio, etc) para cerrar la ventana y continuar con nuestro código. Este mensaje es utilizado por **CodeBlocks** para dar una pequeña **pausa** a nuestro programa y poder observar algunos resultados.



Imagen 2.5. Parte de la ventana del **explorador de Windows**. El archivo **main.cpp** fue guardado y se han generado 2 archivos **main.exe** y **main.o**. El archivo **main.exe** es el archivo que podemos ejecutar, dando un **doble clic** sobre él.

Lo anterior puede tener poco sentido, pero si optamos por la segunda forma de ejecutar nuestro programa, veremos el sentido que tiene la **pausa** anterior. Si ejecutamos el archivo **main.exe**, veremos una pantalla negra aparecer y como en cuestión de milisegundos, ésta desaparece. Podemos ser muy cuidadosos y ver que en esos pequeños milisegundos de pantalla negra, se puede apreciar **SOLO LA FRASE Hola mundo**, más no el mensaje extra de la **imagen 2.4.b**.

A pesar de haber logrado que nuestro primer código funcionara, hace falta detenernos un poco para analizar qué es lo que hemos escrito, es decir, analizar y definir todas y cada unas de las instrucciones que en primera instancia, podría parecernos poco familiares. Por otro lado, hemos logrado que interactúe nuestro código en texto con el compilador, lo cual es un avance bastante considerable ya que hemos logrado que a partir de instrucciones “básicas”, creamos un código que realiza lo que le pedimos.

En la siguiente sección iniciaremos explicando los detalles acerca del **código 2.1**, utilizando además para introducir los conceptos básicos del lenguaje C/C++.

2.3 Posibles errores

- Recuerda guardar el archivo con extensión **.c** o **.cpp**, es importante que el IDE identifique este tipo de archivos. El detalle de color, también puede ayudar.
- Debes tener cuidado como escribes las instrucciones. El tipo de letra natural para C/C++ son las minúsculas. La mayoría o casi todas las instrucciones deben estar en minúsculas. Usar mayúsculas es instrucciones que ya existen como **printf**, **void**, **int**, **main**, **return**, puede ocasionar errores.
- Además de escribir correctamente las instrucciones, no debes olvidar los símbolos. En el **código 2.1** algunos de los símbolos son: las **llaves** “{” y “}”, **paréntesis** “(” y “)”, **punto y coma** “;”.
- Trata de no mezclar instrucciones del lenguaje C con C++.
- Asegúrate de escribirlo los códigos correctamente.