

Practica de laboratorio

Introducción a CodeBlocks

1. Requerimientos

Para la Instalación en sistema operativo Windows:

- ✓ Se usaran las siguientes aplicaciones: CodeBlocks (escoger versión que se acomode a su sistema operativo), junto con MinGW (que incluye el compilador GCC / G++ y el depurador GDB de TDM-GCC).
- ✓ El sitio donde se descargara CodeBlocks es: <http://www.codeblocks.org/downloads>
- ✓ Escoger la descarga de código binarios (Download the binary release).
- ✓ Seleccionar la plataforma.

Downloads

There are different ways to download and install Code::Blocks on your computer:

- **Download the binary release**

This is the easy way for installing Code::Blocks. Download the setup file, run it on your computer and Code::Blocks will be installed, ready for you to work with it. Can't get any easier than that!

- **Download a nightly build:** There are also more recent so-called *nightly builds* available in the **forums**. Please note that we consider nightly builds to be *stable*, usually, unless stated otherwise.
- Other distributions usually follow provided by the **community** (big "Thank you!" for that!). If you want to provide some, make sure to announce in the forums such that we can put it on the official C::B homepage.

- **Download the source code**

If you feel comfortable building applications from source, then this is the recommend way to download Code::Blocks. Downloading the source code and building it yourself puts you in great control and also makes it easier for you to update to newer versions or, even better, create patches for bugs you may find and contributing them back to the community so everyone benefits.

- **Retrieve source code from SVN**

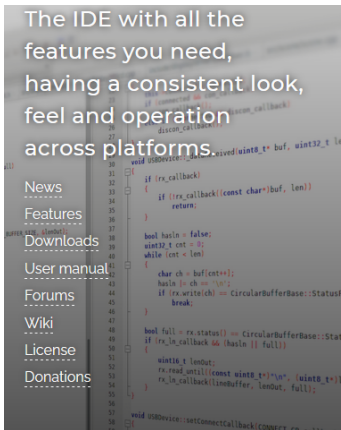
This option is the most flexible of all but requires a little bit more work to setup. It gives you that much more flexibility though because you get access to any bug-fixing we do [at the time we do it](#). No need to wait for the next stable release to benefit from bug-fixes!

2. Instalación de CodeBlocks con MinGW

2.1. Paso 1: Descarga de CodeBlocks con MinGW

Ir a la página de descarga de CodeBlocks, luego verificar versión de sistema operativo que usa su ordenador: si es de 32 bits o 64 bits.

- Esa información se puede ver en la ventana de información básica de equipo, dentro del panel de control de Windows.
- Si es de 64 bits, descargar la versión actual que incluye MinGW, por ejemplo: codeblocks-20.03mingw-setup.exe
- Si es de 32 bits, descargar la versión actual que incluye MinGW, por ejemplo: codeblocks-20.03mingw-32bit-setup.exe



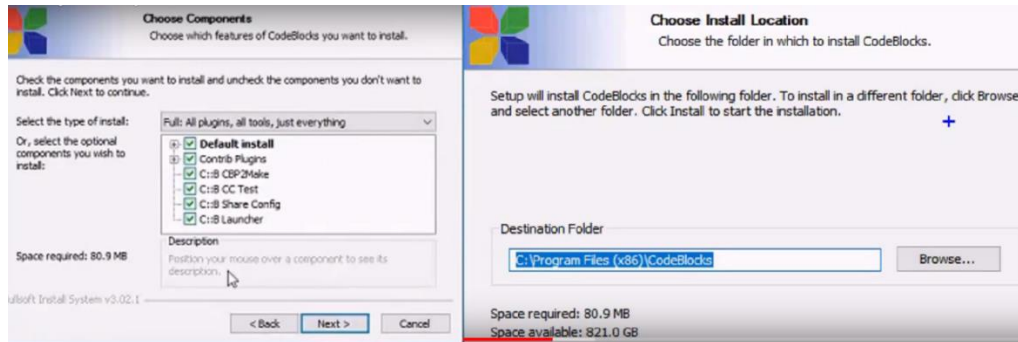
NOTE: The default builds are 64 bit (starting with release 20.03). We also provide 32bit builds for convenience.

Microsoft Windows

File	Download from
codeblocks-20.03-setup.exe	FossHUB or Sourceforge.net
codeblocks-20.03-setup-nonadmin.exe	FossHUB or Sourceforge.net
codeblocks-20.03-nosetup.zip	FossHUB or Sourceforge.net
codeblocks-20.03mingw-setup.exe	FossHUB or Sourceforge.net
codeblocks-20.03mingw-nosetup.zip	FossHUB or Sourceforge.net
codeblocks-20.03-32bit-setup.exe	FossHUB or Sourceforge.net
codeblocks-20.03-32bit-setup-nonadmin.exe	FossHUB or Sourceforge.net
codeblocks-20.03-32bit-nosetup.zip	FossHUB or Sourceforge.net
codeblocks-20.03mingw-32bit-setup.exe	FossHUB or Sourceforge.net
codeblocks-20.03mingw-32bit-nosetup.zip	FossHUB or Sourceforge.net

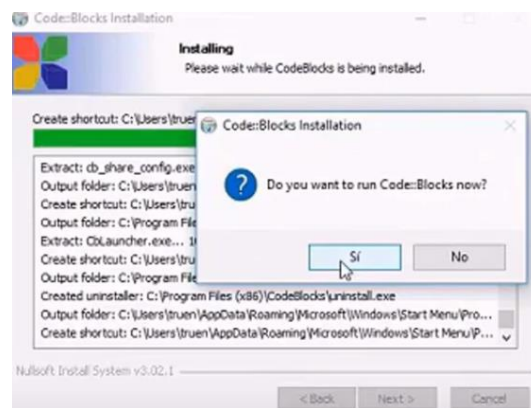
2.2. Paso 2: Instalación de CodeBlocks con MinGW

Una vez completada la descarga, pulsar el botón "Siguiente", aceptando la licencia, los componentes por defecto, la ruta del disco duro donde residirá el software.

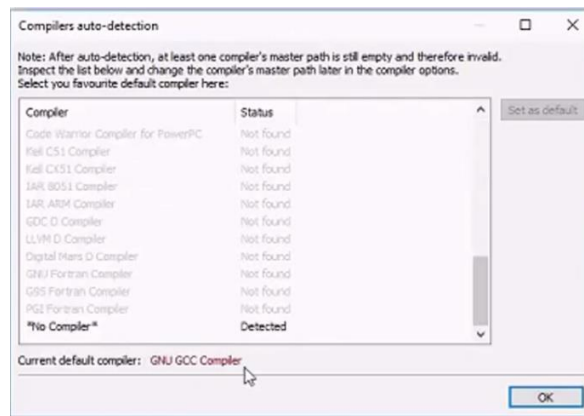


2.3. Paso 3: Selección de compilación asociada a C++

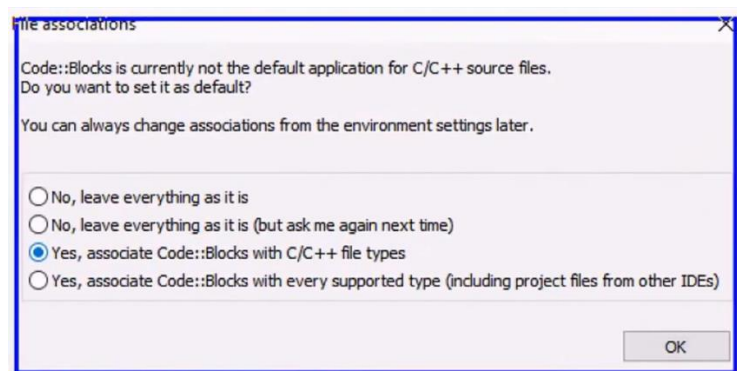
En algún momento nos preguntara si debemos correr ahora el CodeBlocks, le decimos que sí.



Antes de terminar la instalación de CodeBlocks, es probable de que salga una ventana preguntando quien va a ser el compilador, se recomienda dar por finalizado la instalación de CodeBlocks cuando este presente el botón de terminar.



Le damos ok, nos saldrá otra ventana preguntando quien se va asociar con nuestro CodeBlocks, le indicamos que lo asociaremos con archivos de extensión CPP, es decir archivos de tipo C++.

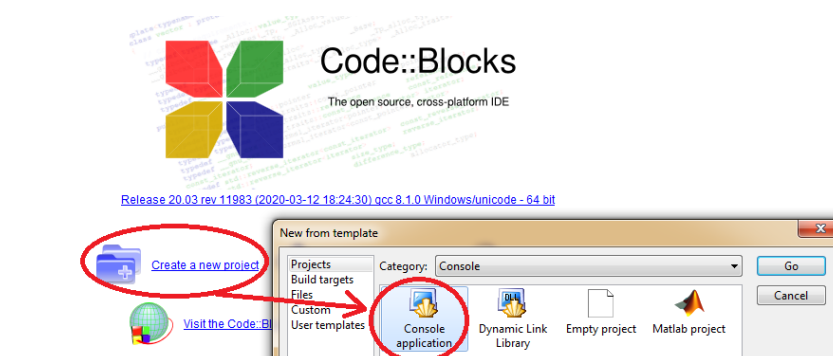


Esto suele pasar cuando tenemos más de una ID instalada, es decir, si también contamos con NetBeans, Eclipse, etc.

2.4. Paso 4: Crear nuevo proyecto

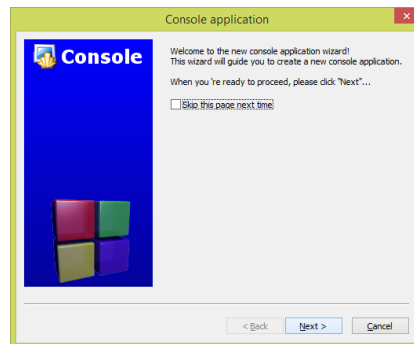
Luego de mostrar la plataforma de CodeBlocks, procedemos a crear un nuevo proyecto:

- Seleccionar FILE -> NEW -> PROJECT, o desde la opción que se muestra en la pantalla inicial: "Create a new Project".
- Seleccionar como plantilla de proyecto la opción "Console Application", ubicado específicamente desde la Categoría "Console".
- Presionar botón "Go".

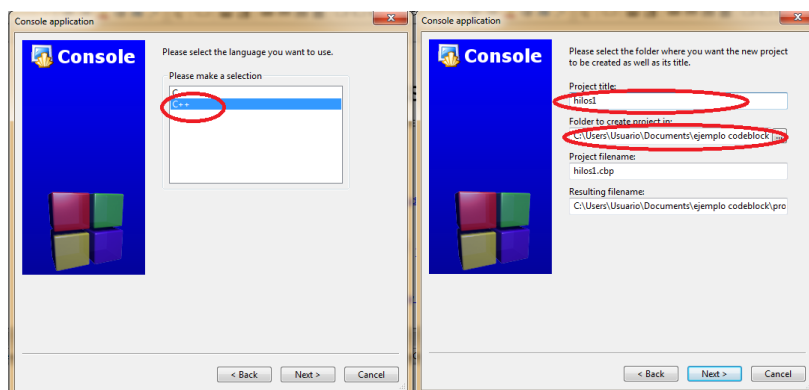


De forma opcional usamos el menú desplegable de categoría para ayudarnos a buscar la opción "Console Application"

Nos aparecerá una ventana de asistente de aplicación en consola donde se configura el proyecto, presionar el botón “Next”.



Les damos “Next” a las opciones: el lenguaje de programación a emplear es C++, especificar el nombre del proyecto y la ubicación del directorio del proyecto en el disco duro.

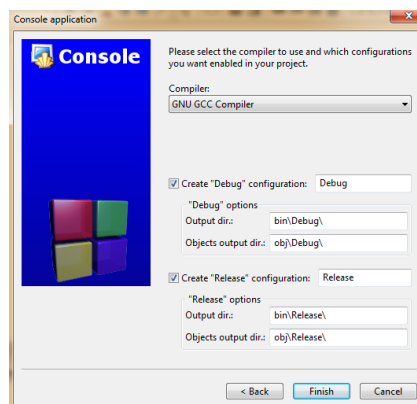


La extensión CBP es especial para proyectos de C++.

A continuación saldrá una ventana donde se debe de seleccionar el compilador del proyecto y los perfiles:

- Debug: Proceso de desarrollo de software.
- Release: Opciones del compilador de mayor optimización, eliminando información de depuración terminado la compilación.
- GNU GCC: Compilador provisto por defecto por CodeBlocks.

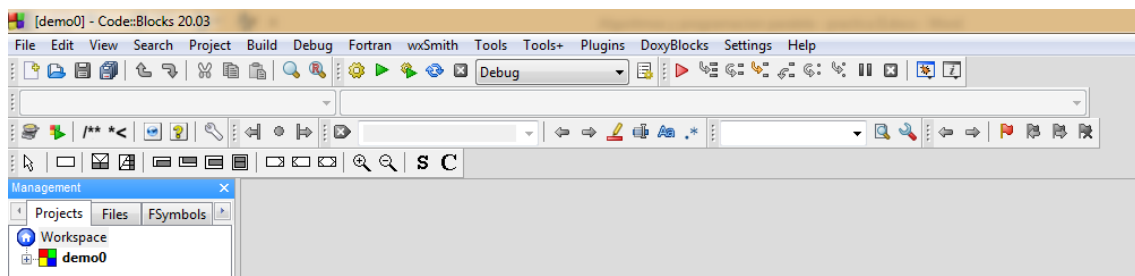
Dejar las opciones por defecto y presionar el botón “Finish”.



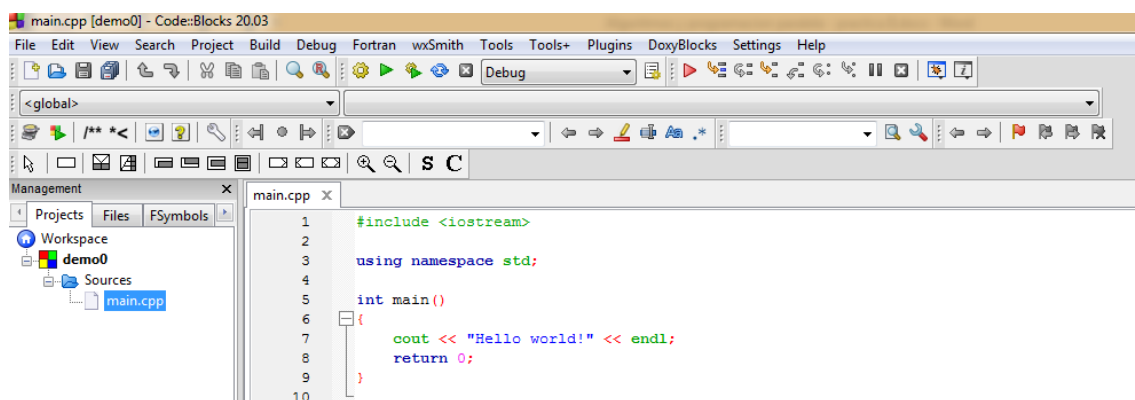
Las compilaciones son generadas gracias a las rutas de variable de entorno de Windows que se encuentran en la carpeta “/bin”.

2.5. Paso 5: Abriendo archivo CPP

Inmediatamente genera un WorkSpace o entorno de trabajo dentro de la pestaña “Projects” de la venta de administrador “Management”.



Hacer doble click sobre Workspace, seguido del nombre de proyecto, luego doble click sobre la carpeta “Sources”, allí se encuentra el archivo “main.cpp”.

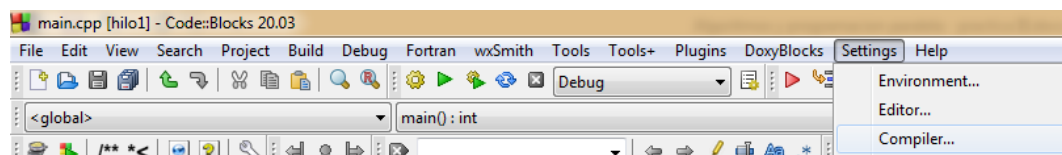


Por defecto ya contiene código con una demostración de prueba.

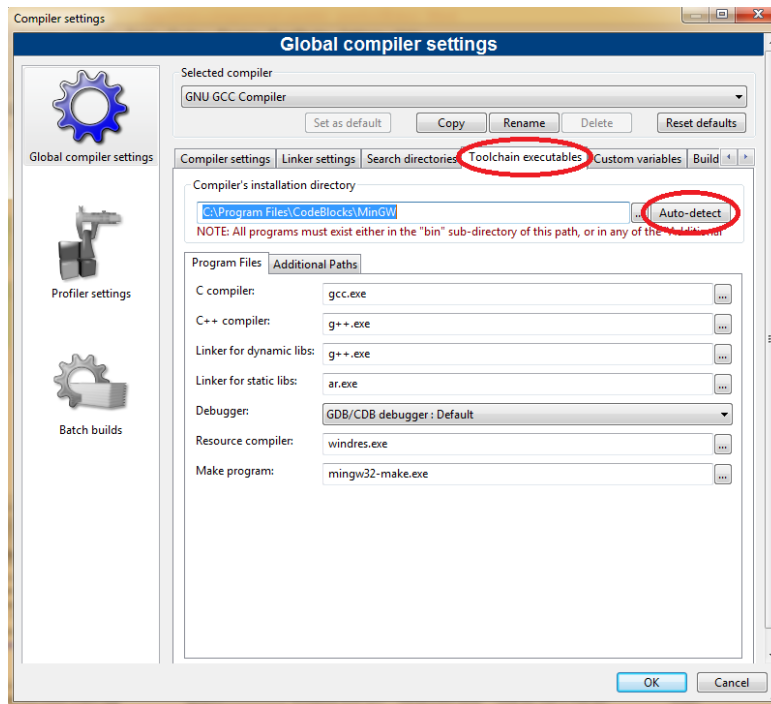
2.6. Paso 6: Verificar ruta de MinGW

Se verifica la ubicación del MinGW, que es donde contiene los paquetes y librerías necesarios para la compilación del programa.

Para ello, dentro de la ventana de entorno de CodeBlocks, ir a menú “Setting... -> Compiler ...”

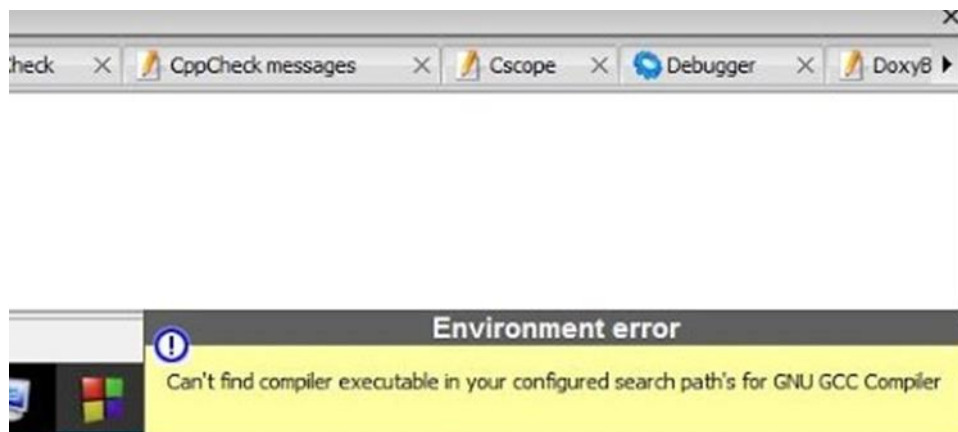


Nos dirigimos a la pestaña “Toolchain executables”, y presionamos el botón de auto detectar, para buscar la ubicación de la carpeta MinGW.

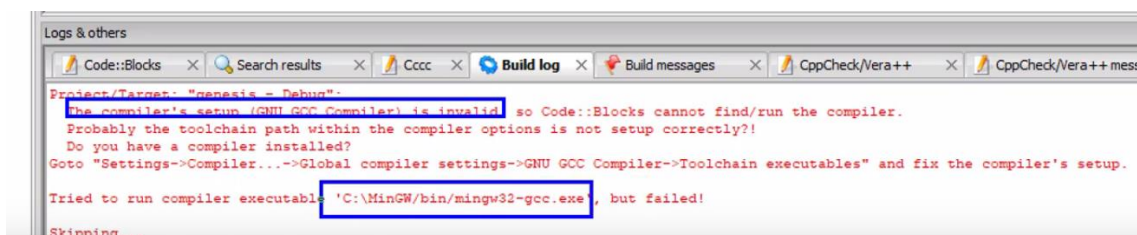


En caso de que no lo auto detecte, ingresar manualmente la ubicación de la carpeta MinGW y cambiar los siguientes valores: gcc.exe y g++.exe, como se mostró en la imagen. Luego, finalizar los cambios.
















Sin embargo, si aún persiste que no encuentra el compilador:



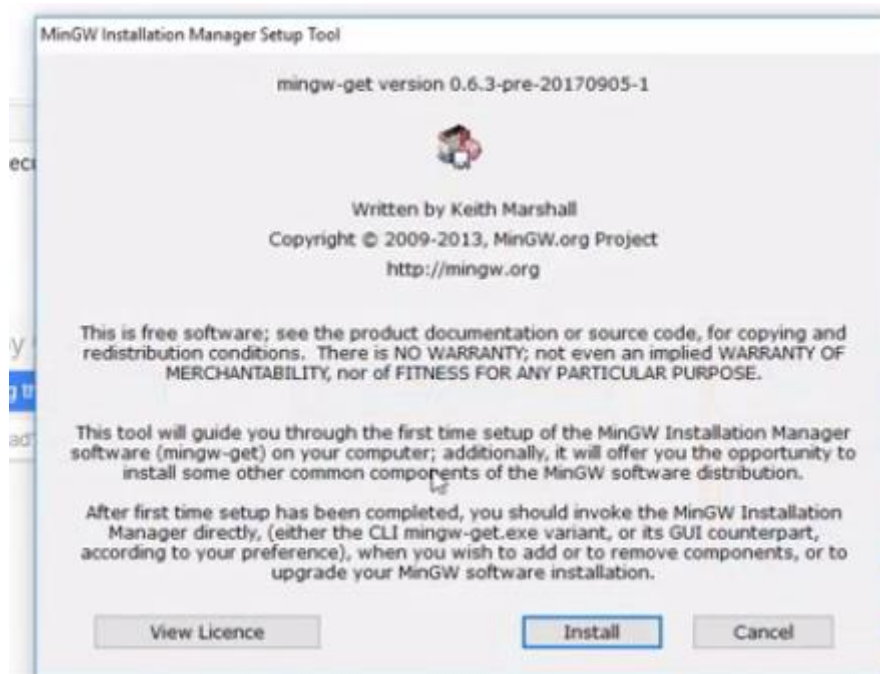
Significa que no está reconociendo el compilador de MinGW que ha venido instalado, debido a que si se intenta ejecutar el archivo, saldrá el mensaje de error:



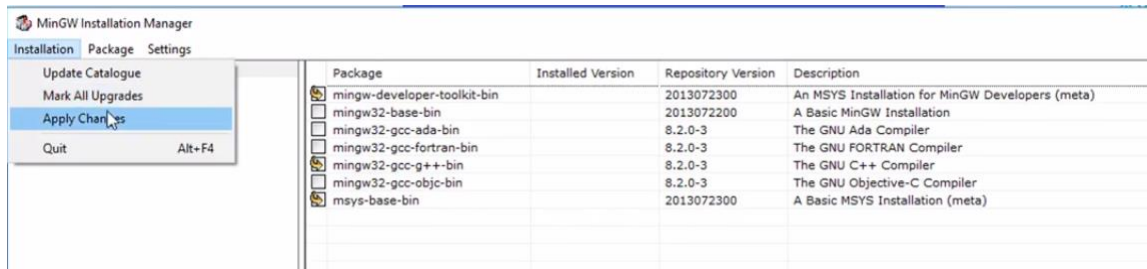
Se recomienda instalar manualmente MinGW y escoger versión de sistema operativo (es probable que redirija hacia otro sitio de descarga), mediante el siguiente enlace: <http://mingw-w64.org/doku.php/download>

	Version	Host	GCC / Mingw-w64 Version	Languages	Additional Software in Package Manager
 Arch Linux		Arch Linux	8.2.0/5.0.4	Ada, C, C++, Fortran, Obj-C, Obj-C++	305, full list: Show
 Cygwin	Rolling	Windows 	5.4.0/5.0.2	Ada, C, C++, Fortran, Obj-C	5 (bzip2, libgrypt, libgpg-error, minizip, xz, zlib)
 Debian	Debian 7 (Wheezy)		4.6.3/2.0.3	Ada, C, C++, Fortran, Obj-C, Obj-C++, OCaml	2 (gdb, nsis)
	Debian 8 (Jessie)		4.9.1/3.2.0		
	Debian 9 (Stretch)		6.3.0/5.0.0		9 (gdb, libassuan, libgrypt, libgpg-error, libksba, libnph, nsis, win-iconv, zlib)
	Debian 10 (Buster)		8.3.0/6.0.0		
 Fedora	Fedora 19		4.8.1/?	Ada, C, C++, Fortran, Obj-C, Obj-C++	149, full list: Show
 MacPorts	Rolling	macOS 	8.2.0/5.0.4	C, C++, Fortran, Obj-C, Obj-C++	1 (nsis)
 MingW-w64-builds	Rolling	Windows 	7.2.0/5.0.3	C, C++, Fortran	4 (gdb, libiconf, python, zlib)
 Msys2	Rolling	Windows 	9.2.0/trunk	Ada, C, C++, Fortran, Obj-C, Obj-C++, OCaml	many
 Ubuntu	12.04 Precise Pangolin		4.6.3/2.0.1	Ada, C, C++, Fortran, Obj-C, Obj-C++, OCaml	2 (nsis, gdb)
	14.04 Trusty Tahr		4.8.2/3.1.0		
	14.10 Utopic Unicorn		4.9.1/3.1.0		
	15.04 Vivid Vervet		4.9.2/3.2.0		
	15.10 Wily Werewolf		4.9.2/4.0.2		3 (nsis, gdb, zlib)
	16.04 Xenial Xerus		5.3.1/4.0.4		
 Win-Builds	1.5	Windows 		C, C++	91, full list: Show
		Linux 	4.8.3/3.3.0		

Se procede a instalar MinGW, marcado el botón “Siguiente” por defecto en ubicación de ruta

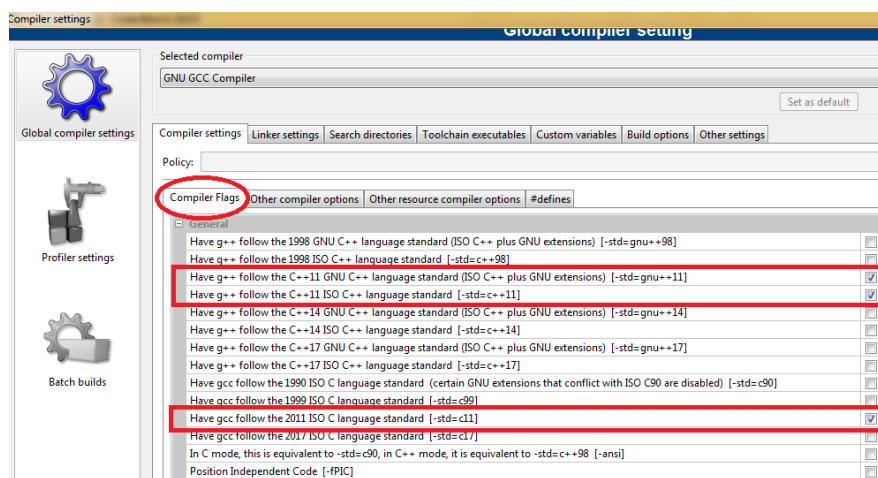


Luego, aparecerá la venta de administración de versión de MinGW, donde se marca los componentes que se van a necesitar para compilar (siendo más relevante el depurador GCC / G++), luego aplicar los cambios dentro de menú de instalación.



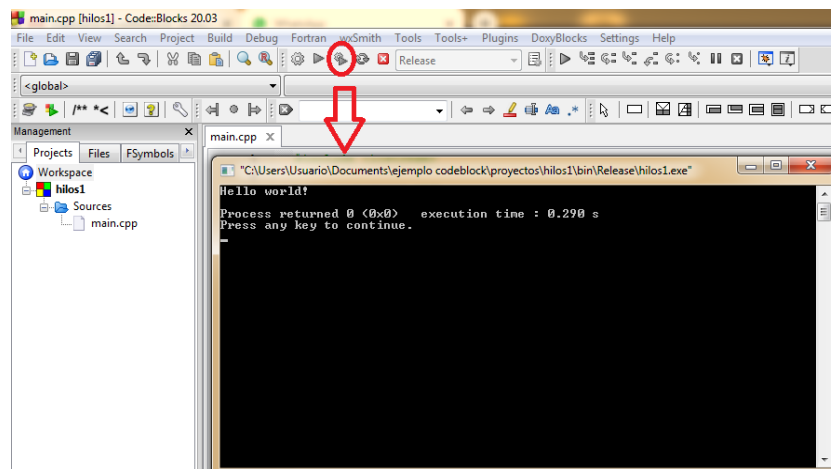
2.7. Paso 7: Seleccionar la orientación de compilación

Para terminar la verificación, volver a ir al menú “Setting... -> Compiler ...”, y seleccionar la pestaña “Compiler Flag”, dependiendo de las clases y funciones que se van a usar, seleccionar el ISO y lenguaje estándar apropiado.



2.8. Paso 8: Compilar Proyecto

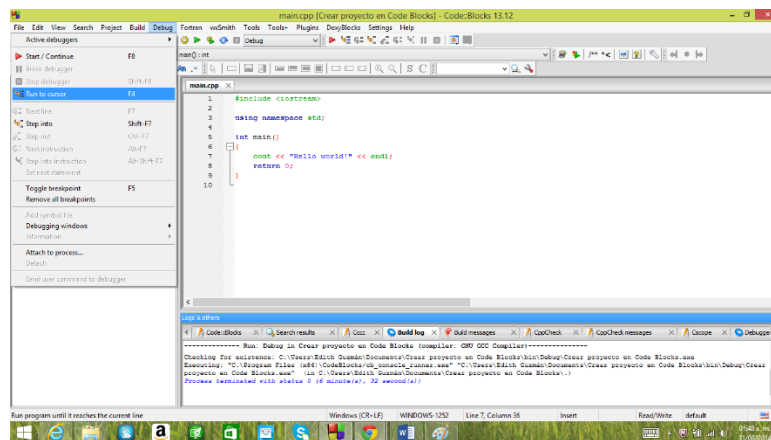
Presionar el botón de “Build and Run” dentro de la pestaña “Build”, o presionamos la tecla “F9” mientras el archivo “Main.cpp” se encuentre abierto, el programa va a compilar y mostrar los resultados del código.



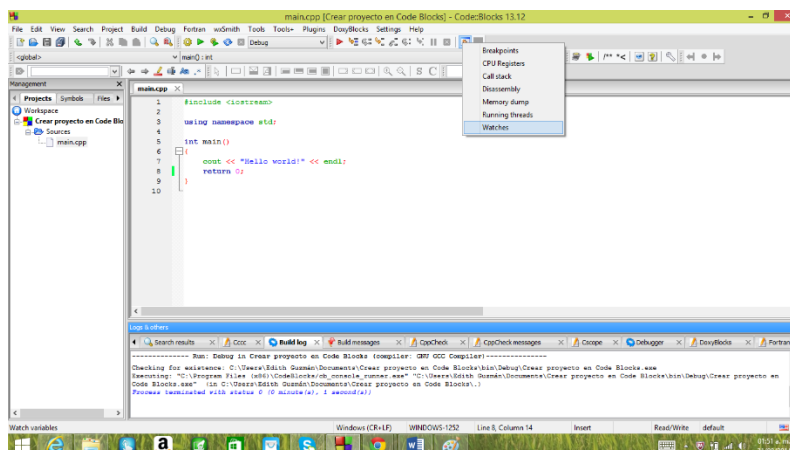
En caso de haber cerrado el proyecto, se recomienda usar la opción de abrir proyectos existentes.

2.9. Paso 9: Debug en proyecto.


El Debug permite correr el programa línea por línea, también muestra cuánto valen las variables a cada paso. Se inicia presionando la opción “Run” dentro del menú Debug.





Para ver los valores de las variables, dentro del menú de Debug seleccionar la opción **Debugging Windows** y marcar opción: **Watches**. Esto nos abre una ventana donde podemos ver las variables y sus valores.





También se encuentran a la vista otros comandos de Debug:

Debug/Continue (). Para comenzar/continuar el debug.


Run to cursor (). Corre el programa hasta donde esta el cursor.

Next line (). Avanza una línea la ejecución.

Step into (). En caso de estar en una línea que es una llamada a una función, con esta opción podemos hacer el seguimiento dentro de esta función (Ojo con las variables locales que la idea de “local” cambia).

Step out (). Termina la ejecución de la función actual y sigue en la siguiente línea desde donde fue llamado.

Break debugger (). Interrumpe el debug.

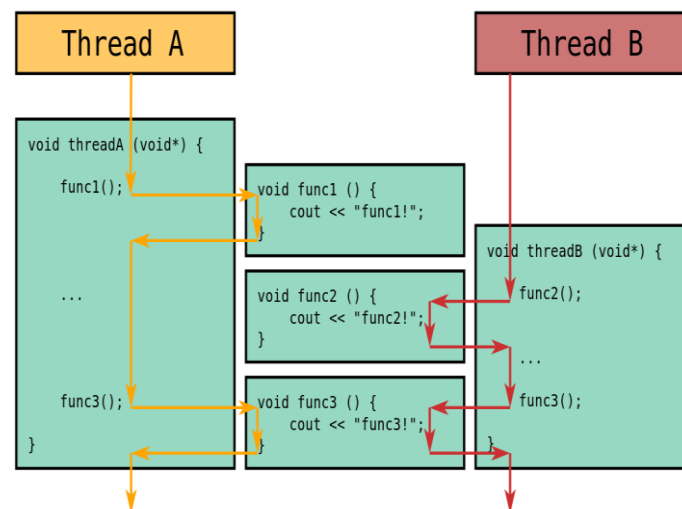
Stop debugger (). Termina el debug.

3. MARCO TEÓRICO

3.1. Hilos

Permiten que se ejecuten varias tareas o programas a la vez, leyendo cada instrucción uno por uno a lo largo de varios núcleos del procesador, donde:

- Los hilos ejecutan múltiples tareas en forma asíncrona y simultánea.
- Los hilos inician la ejecución inmediatamente después de la construcción del objeto hilo asociado
- Los hilos acceden a un recurso de forma aleatoria
- Cuando se modifican valores de dos o más hilos, uno de los hilos se modifica primero, luego el otro y viceversa.



3.2. Recursos de hilos

En archivo CPP, dentro de la estructura del programa, se usaran las siguientes clases y funciones, sin embargo:

- Para las clases, se necesita incluir en el encabezado la librería “include <thread>”
- Para cada función, se define previamente el espacio de nombre “std::this_thread::<función>”:

Nombre de Hilo	Compilador	Descripción de hilo	Tipo
thread	(C++11)	Gestiona un hilo/subproceso independiente.	(clase)
jthread	(C++20)	std::thread con soporte para unión y cancelación automática.	(clase)
yield	(C++11)	Sugiere a la implementación que re programe la ejecución de hilos.	(función)
get_id	(C++11)	Devuelve el identificador de hilo/subproceso del hilo/subproceso actual.	(función)
sleep_for	(C++11)	Detiene la ejecución del hilo/subproceso actual por una duración de tiempo especificada.	(función)
sleep_until	(C++11)	Detiene la ejecución del hilo/subproceso actual hasta que ocurra un punto de tiempo especificado.	(función)

También se usan métodos para evitar que los hilos aun sigan ejecutándose luego de que el programa principal haya concluido.

- Método `join()`: Es un método que permite esperar hasta que el hilo concluya, para luego unir el siguiente hilo a ejecutar.

```
void join(); //desde C++11
```

- Método `detach()`: Impide que el hilo muestre error mientras el hilo actual está trabajando, pero los otros hilos serán interrumpidos bruscamente.

```
void detach(); //desde C++11
```

- Método `swap()`: Intercambia dos objetos hilos o thread

```
void swap( objeto thread, otro objeto thread); //desde C++11
```

De forma opcional, se usan algunas funciones públicas para verificar el estado de los hilos, estos necesitan previamente un constructor:

- Función `joinable()`: Comprueba si un objeto identifica a un hilo que esta actualmente en ejecución o a la espera de recibir al método `join()`.

- o `thread t(función);`
- o `t.joinable();` //desde C++11

- Función `get_id()`: Devuelve el identificador de un hilo.

- o `thread t(función);`
- o `std::thread::id t.get_id();` //desde C++11

- Función `native_handle()`: Devuelve el identificador del subprocesso del sistema operativo relacionado al hilo.

- o `thread T(función);`
- o `T.native_handle();` //desde C++11

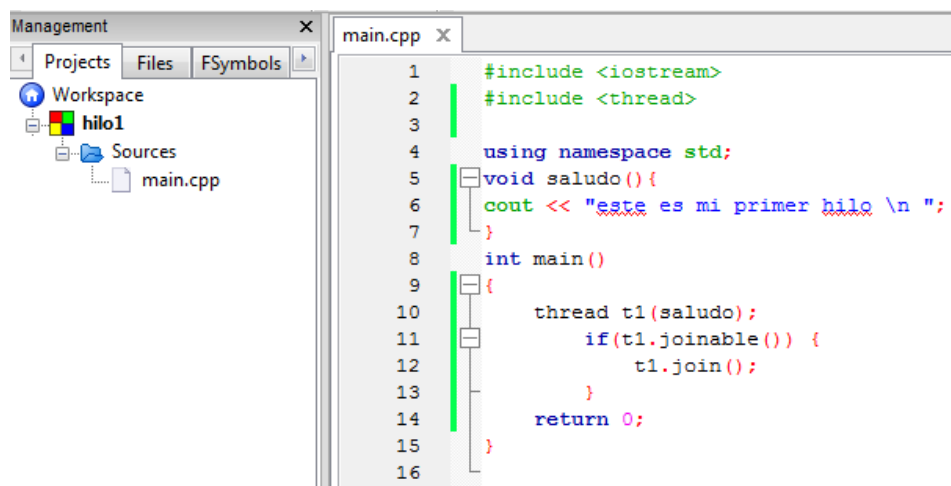
- Función `hardware_concurrency()`: Devuelve el número de hilos soportados por la implementación.

- o `static unsigned hardware_concurrency();` //desde C++11

4. Practica

4.1. Ejercicio 1

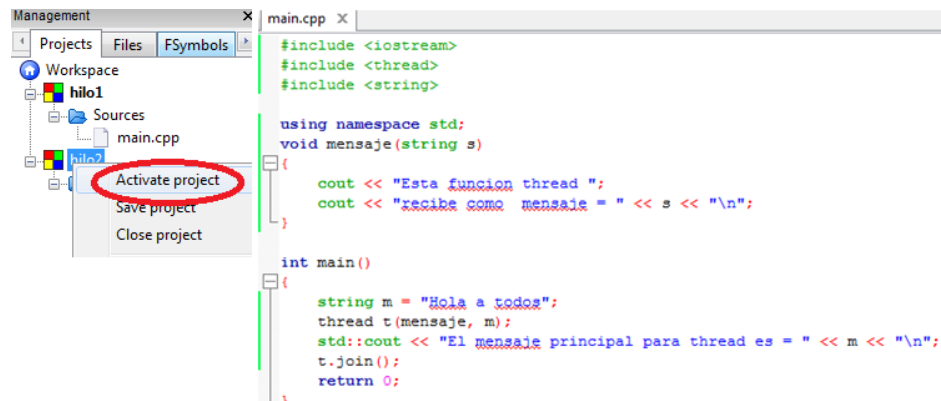
Cuando se usa el constructor "thread" dentro del programa, recibirá como primer argumento la función que se ejecutara.



4.2. Ejercicio 2

Si la función posee argumentos, en el constructor aun tendrá como primer argumento la función que se ejecutara, y los siguientes argumentos los valores para la función al que se está llamando.

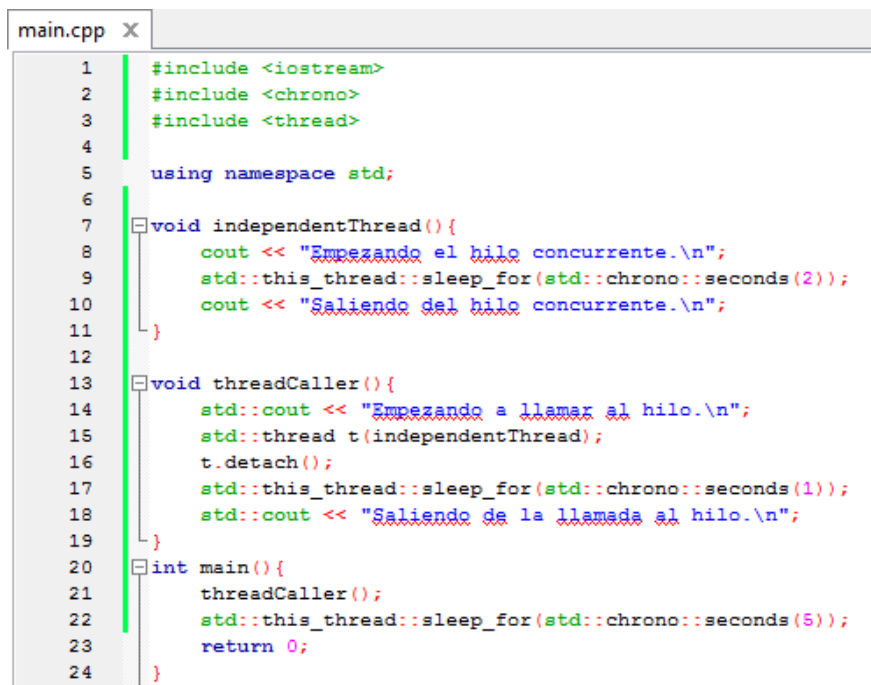
Nota: Para el segundo ejercicio, deberá de indicar cual proyecto quedara activo para ser ejecutado



4.3. Ejercicio 3

También podemos separar los hilos para que estos ejecuten de manera independiente.

Sin embargo, las instrucciones necesitan delimitarse para evitar consumir bastante memoria del CPU, por lo tanto se usa la línea “`this_thread::sleep_for(chrono::milliseconds(5));`”, para esperar 5 milisegundos y así evitar sobrecargar la CPU.



4.4. Ejercicio 4

Existe la posibilidad de hacer ejecutar múltiples hilos para repartir lista de valores, se empieza definiendo las variables.

```
main.cpp x
1  #include <iostream>
2  #include <thread>
3  #include <vector>
4  #include <string>
5  #include <bits/stdc++.h>
6
7  using namespace std;
8
9  double sum1 = 0;
10 double sum2 = 0;
11
12 struct myclass {
13     bool operator()(int i, int j) {return (i<j);};
14 } myobject;
15
16 vector<int> myvector;
17
18
```

Luego las funciones o procesos:

```
19 void task1() {
20     cout << "Task1 esta empezando ... \n";
21     double c=0;
22     while (c < 10){c++; sum1 +=c;}
23     //sum1 = 1+2+3+4+5+6+7+8+9+10
24     cout << "Task1 esta completa. \n";
25 }
26 void task2() {
27     cout << "Task2 esta empezando ... \n";
28     double c=0;
29     while (c < 5){c++; sum2 +=c;}
30     //sum2=1+2+3+4+5
31     cout << "Task2 esta completa. \n";
32 }
33 void task3() {
34     cout << "Task3 esta empezando ... \n";
35     myvector = {12,33,22,44,15,66,56,39,72};
36     std::sort(myvector.begin(),myvector.end(),myobject);
37     cout << "Task3 esta completa. \n";
38 }
```

Por último, se programan los constructores y llamados a funciones:

```

39 int main() {
40     thread t1(task1);
41     thread t2(task2);
42     thread t3(task3);
43
44     thread::id id1 = t1.get_id();
45     thread::id id2 = t3.get_id();
46     thread::id id3 = t3.get_id();
47
48     if (t1.joinable()) {
49         t1.join();
50         cout << "t1 id = " << id1 << "\n";
51     }
52
53     if (t2.joinable()) {
54         t2.join();
55         cout << "t2 id = " << id2 << "\n";
56     }
57
58     if (t3.joinable()) {
59         t3.join();
60         cout << "t3 id = " << id3 << "\n";
61         for(int c=0; c < myvector.size(); c++) {
62             cout << "vector (" << c << "): " << myvector[c] << "\n";
63         }
64     }
65     cout << "sum1: " << sum1 << ", sum2: " << sum2 << "\n";
66     return 0;
67 }

```

5. Asignación

Proponer solución para el siguiente caso:

- Crear un programa concurrente donde son creados y ejecutados tres procesos, hasta su terminación, imprimiendo mensajes para demostrar que se están ejecutando, como conteo de veces en que se ejecutó, el tiempo de espera en milisegundos por cada ejecución, etc.

6. Web gráfica

- ✓ <https://es.cppreference.com/w/cpp/thread>
- ✓ http://www.codeblocks.org/docs/manual_codeblocks_en.pdf