

Introducción a Code::Blocks

Code::Blocks, es una herramienta de entorno de desarrollo integrado (IDE en inglés) para el desarrollo de programas en lenguaje C++. Está basado en la plataforma de interfaces gráficas *WxWidgets*, lo que le permite correr libremente en diversos sistemas operativos, y es de licencia GPL (General Public License, o sea Software Libre). Su distribución es gratuita, tanto el IDE, como el compilador, lo que nos evita registraciones engorrosas, utilizar versiones de prueba o incluso métodos menos transparentes.

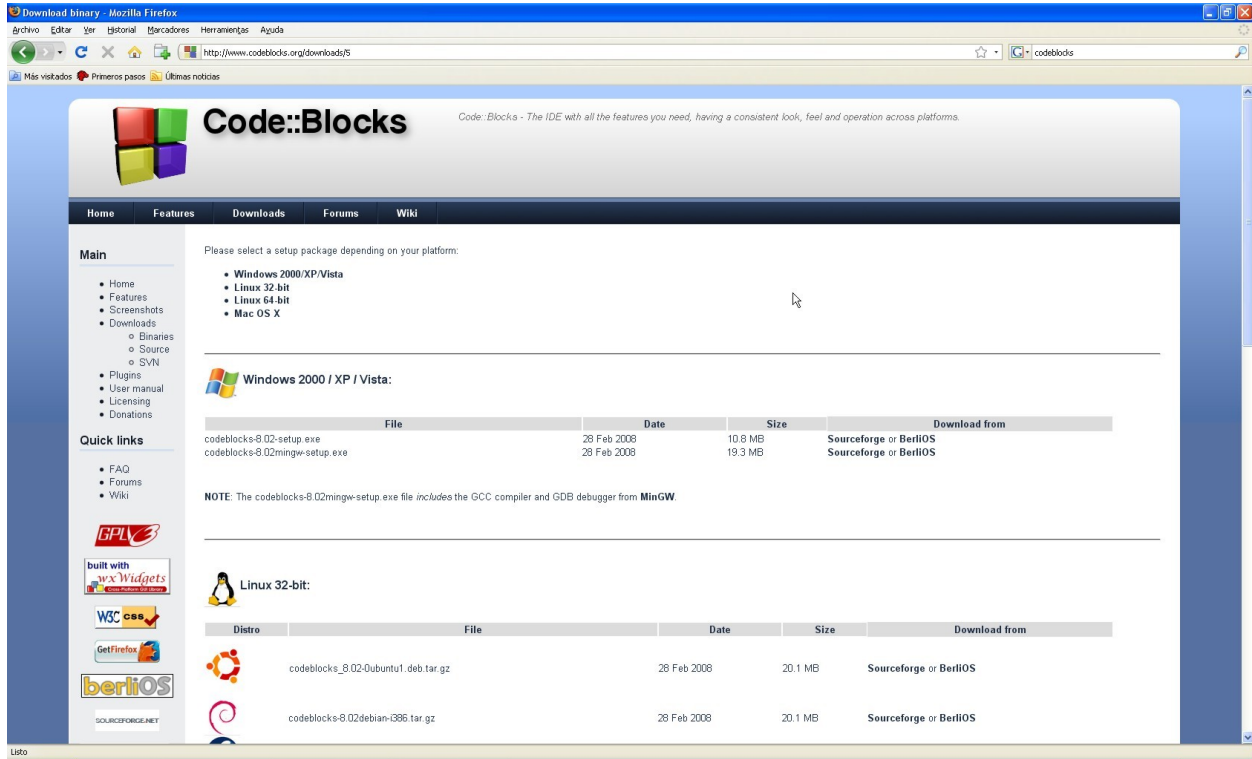
A diferencia de otros editores más antiguos, **Code::Blocks** es compatible con Windows XP y Vista, así como las distintas distribuciones de Linux. Tanto la interfaz gráfica como el compilador mantienen el mismo funcionamiento en las diferentes plataformas, lo que nos garantiza que nuestros programas van a compilar y funcionar de la misma manera más allá de la PC que usemos.

Instalación

1. Abrir un navegador web y dirigirse a <http://www.codeblocks.org/downloads>



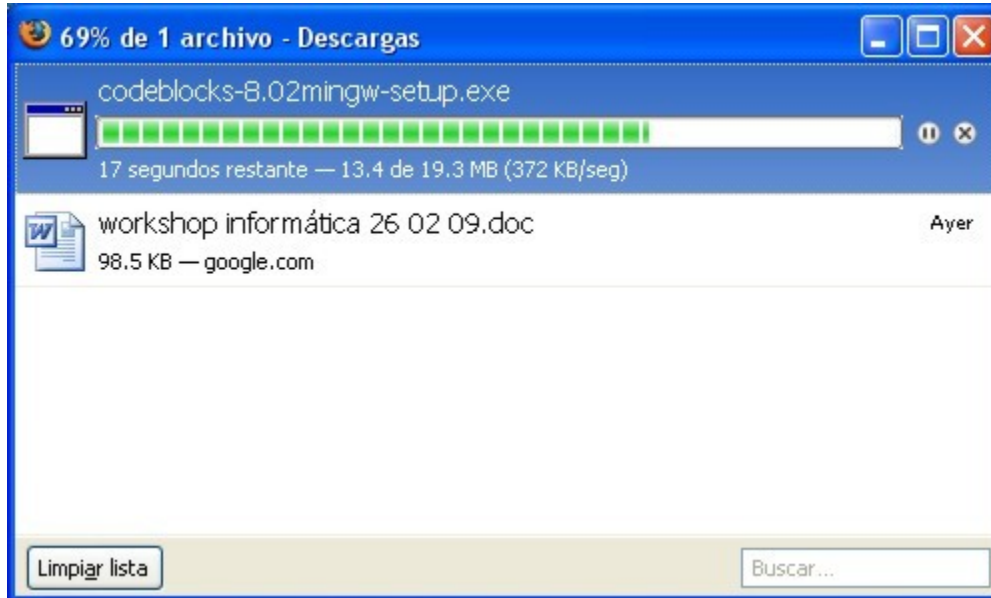
2. Hacer click en la opción “download the binary release”



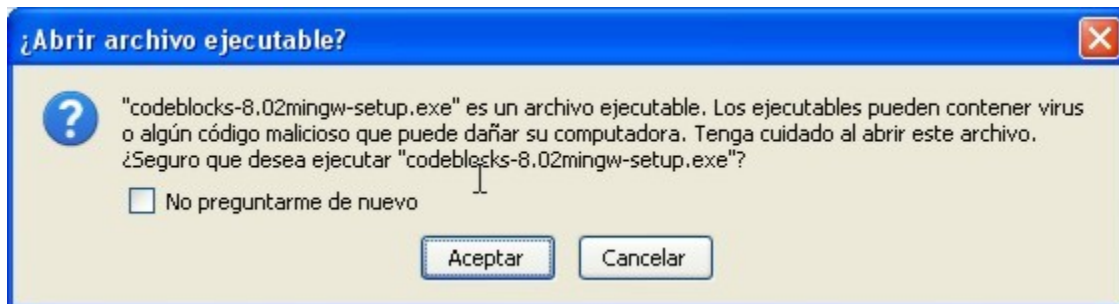
3. En caso de que el sistema operativo sea Windows® elegir la versión con MinGW (19.3MB, generalmente es la más pesada de las que hay disponibles para Windows®)



4. Guardar el archivo



5. Ejecutar el archivo (Internet Explorer ver paso 6)



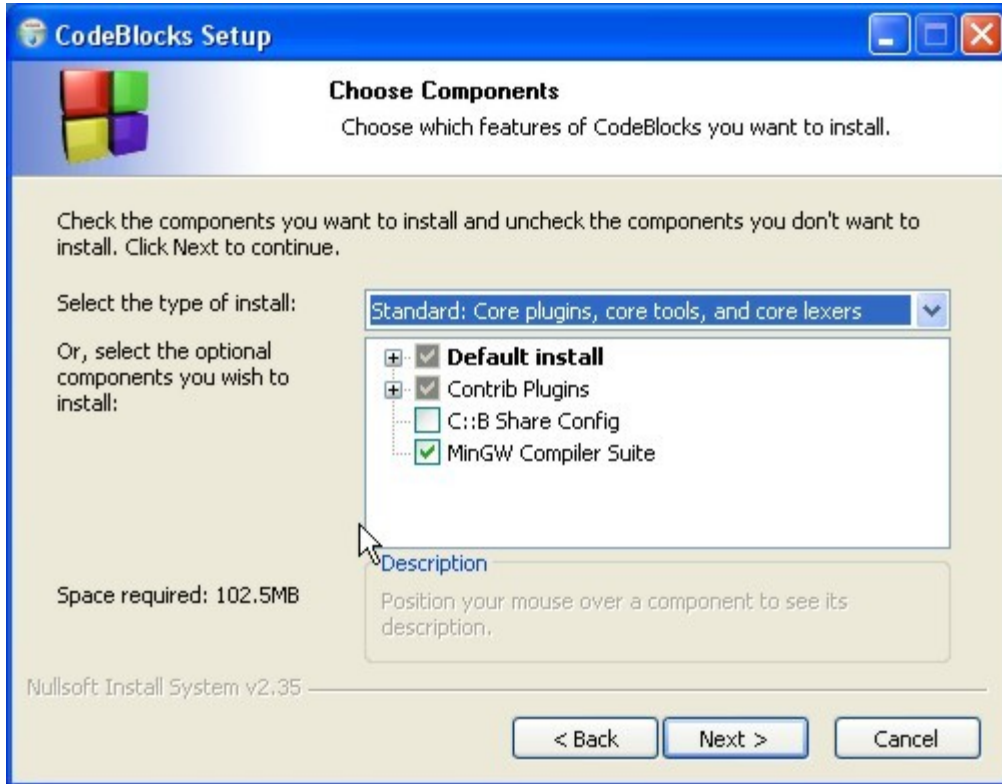
6. Correr el instalador



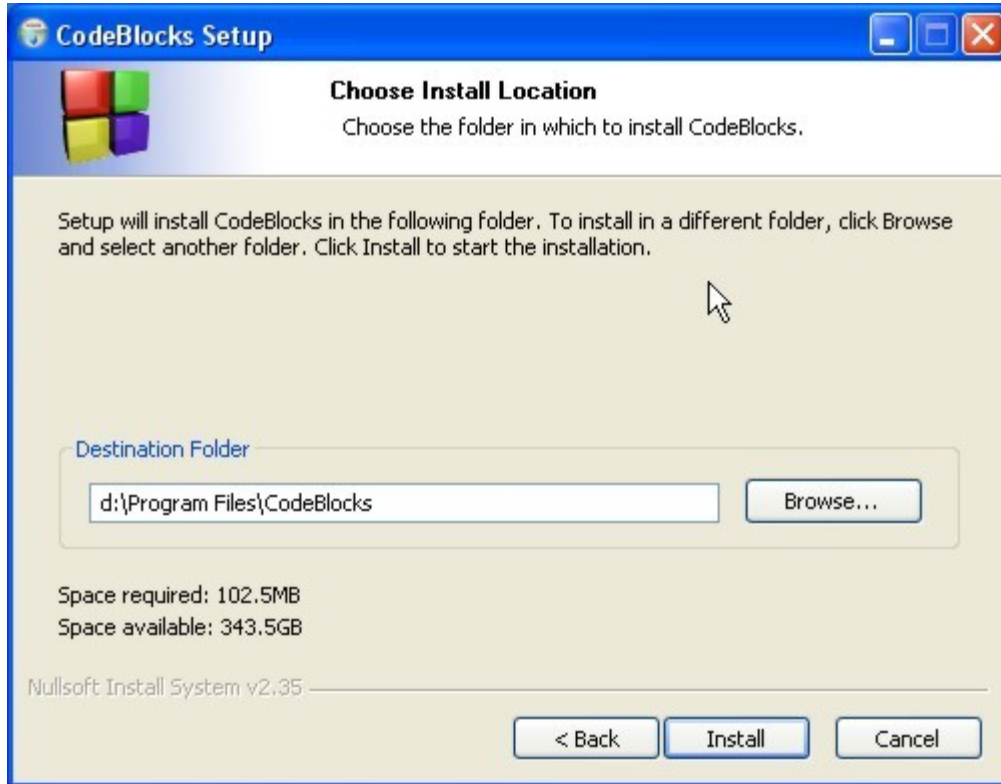
7. Aceptar la General Public License



8. Instalar los componentes Standard



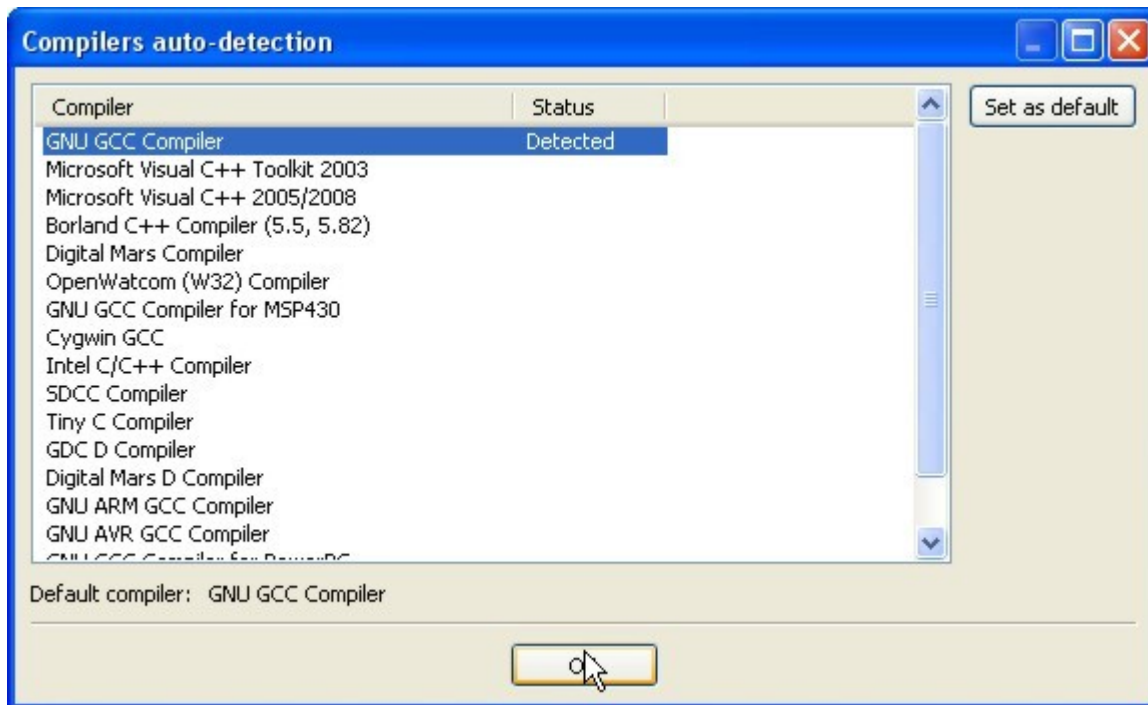
9. Elegir una ruta de instalación



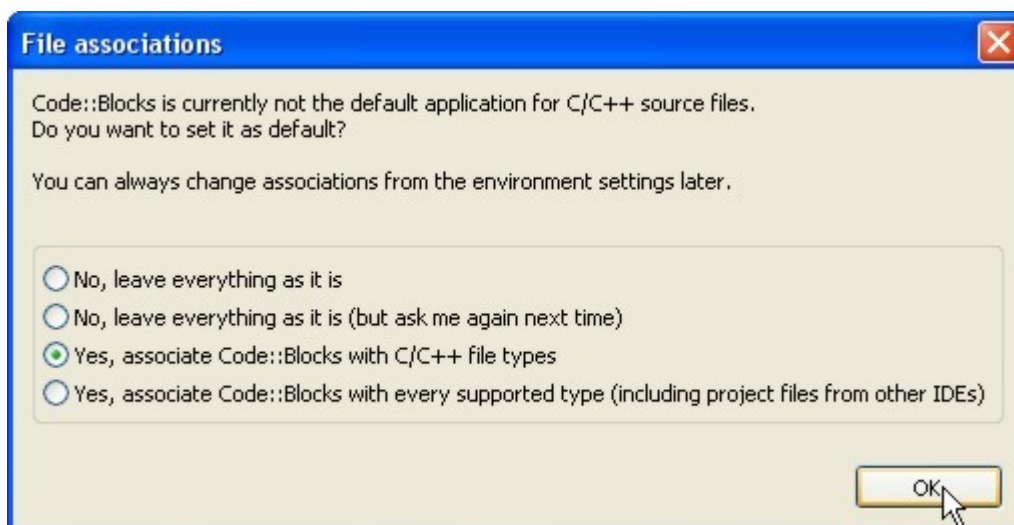
10. Correr Code::Blocks!



11. Seleccionar el compilador. Solo con clicar en OK alcanza, esto lo solicita solo cuando se ejecuta por primera vez.



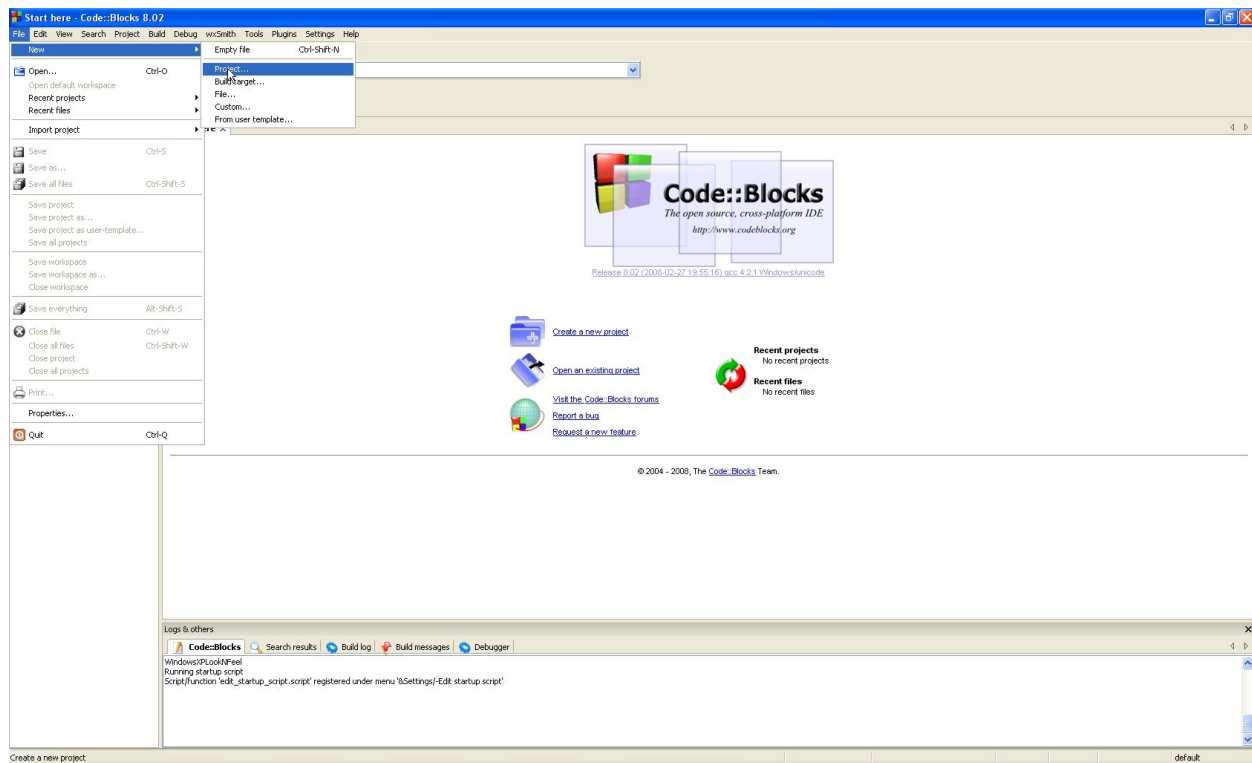
12. Asociar los archivos .C y .cpp a la aplicación (opcional)



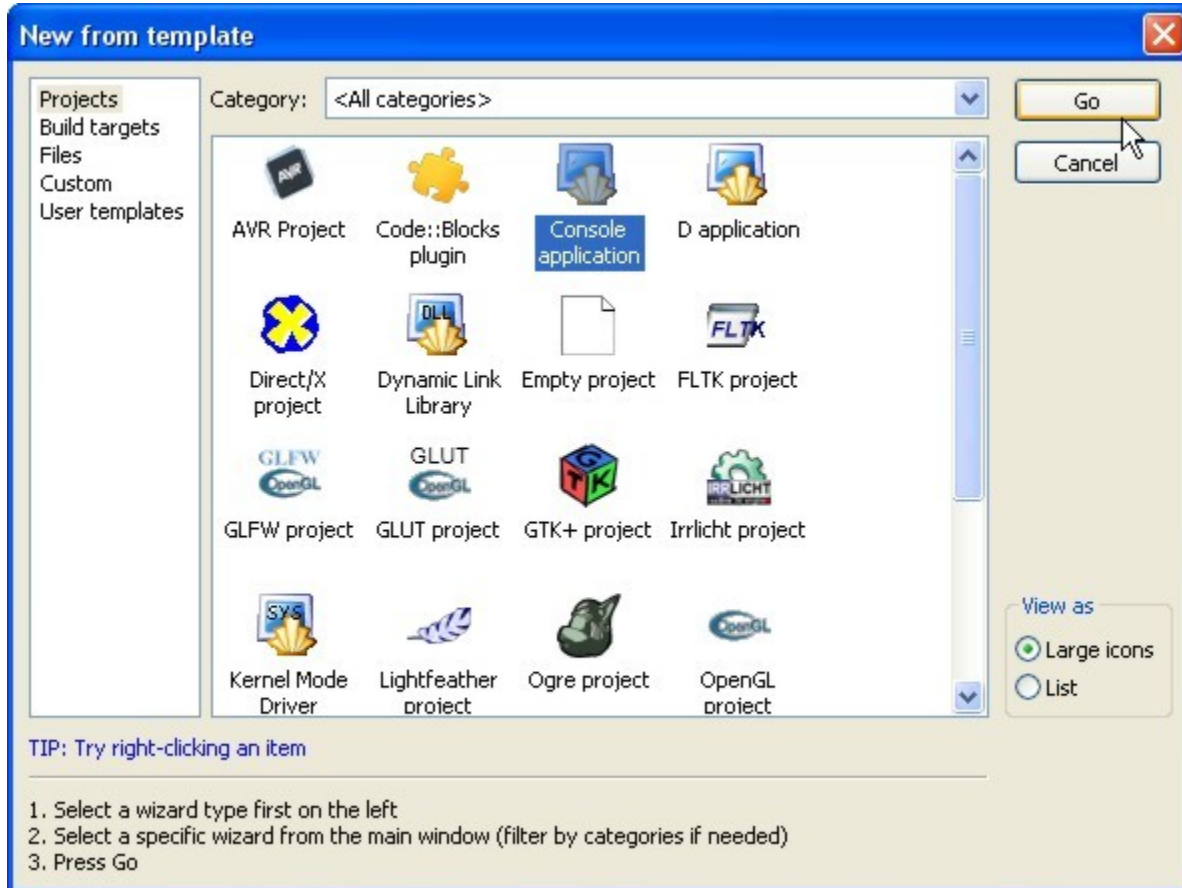
Crear un proyecto para empezar a programar

Code::Blocks permite hacer infinidad de aplicaciones que van más allá de lo que se ve en la materia. En nuestro caso vamos a trabajar con “Console Application” y lenguaje C puro (no C++). El Ide se maneja en base a “proyectos”. Es decir cada programa es un proyecto que puede tener más de un archivo .c y .h (Headers, los vamos a ver más adelante en la materia). Esto nos permite organizar nuestros archivos, así como dividir nuestro programa en porciones de código más chicas si así lo deseamos.

1. File -> New -> Project



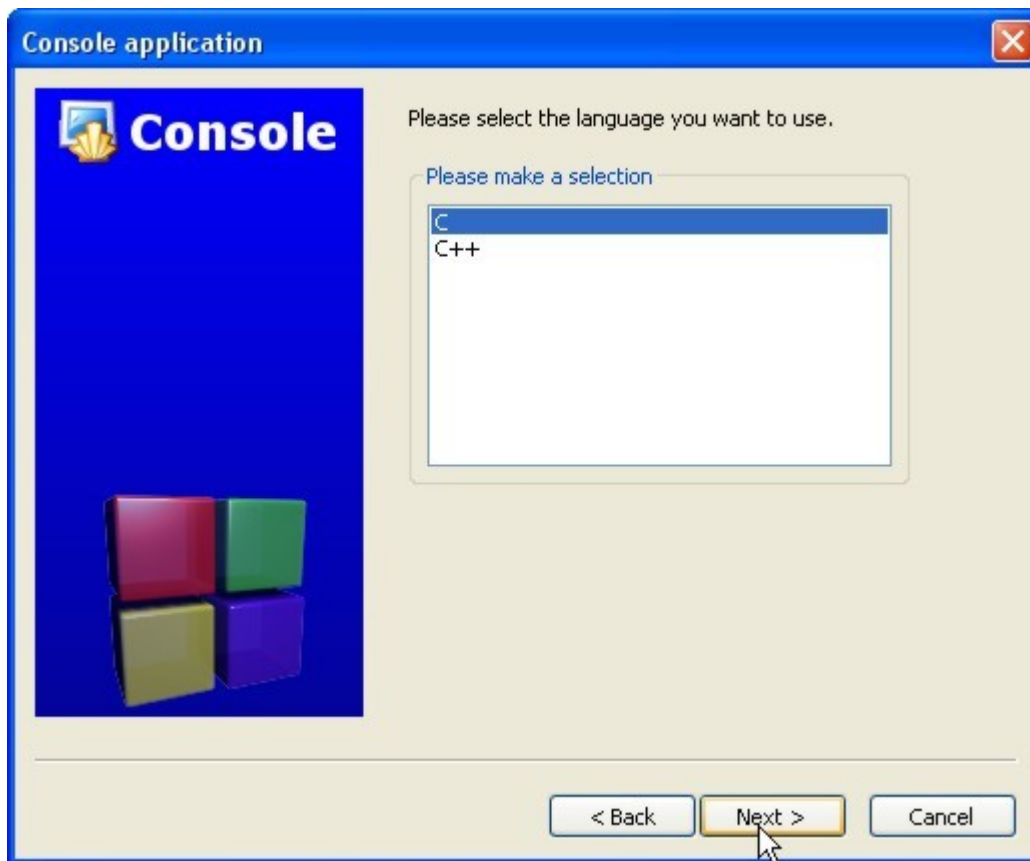
2. Seleccionamos "Console Application" y luego "Go"



3. Se abre el Asistente para crear el proyecto, clicar en Next



4. Seleccionar el lenguaje C y luego next

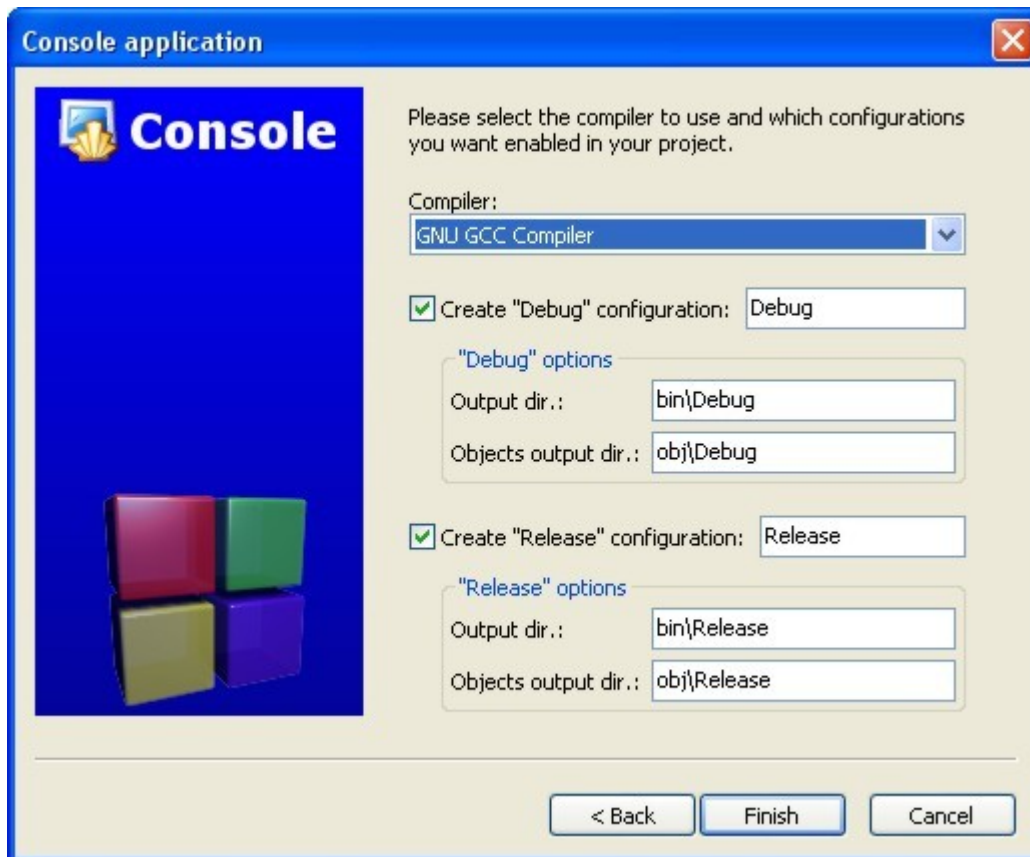


5. Aquí tenemos que ponerle título a nuestro proyecto y elegir una carpeta donde queremos que quede guardado. No hace falta crear una carpeta con el nombre del proyecto mismo, se hace automáticamente.

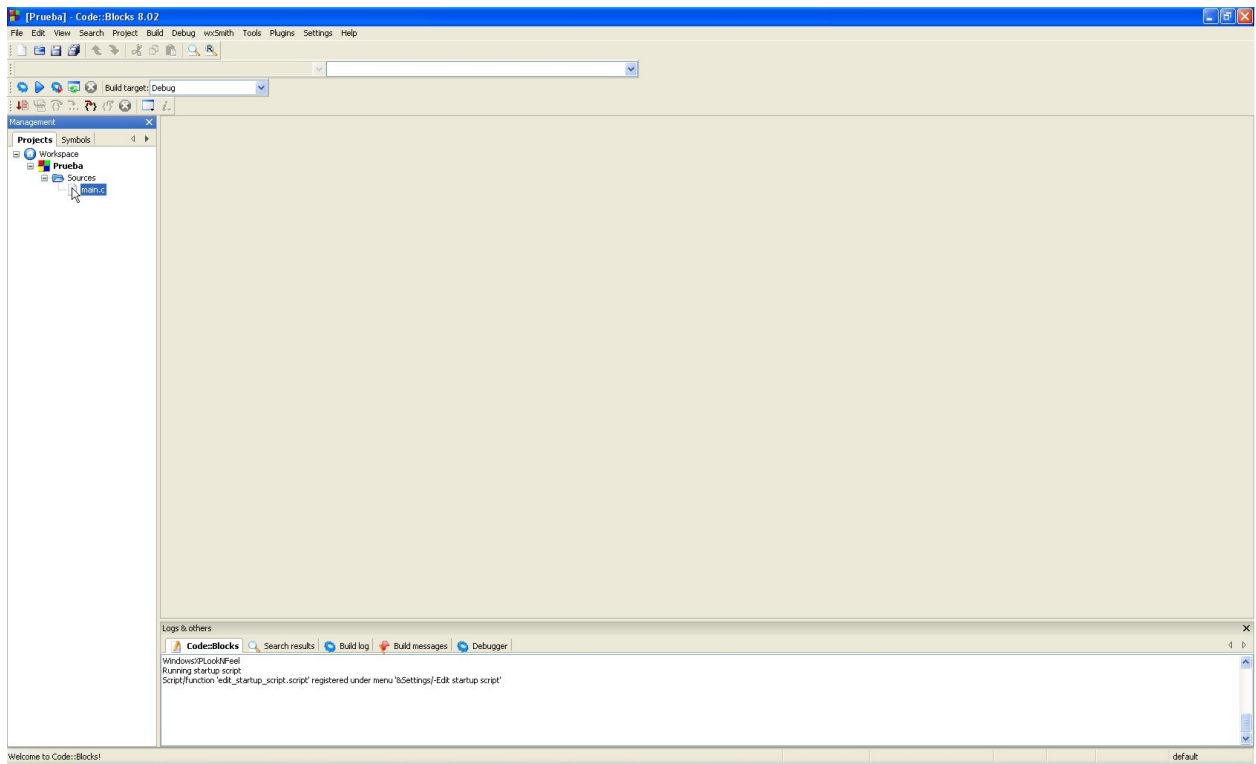
Cuidado: si estas en los laboratorios de la UP, elegí las carpetas que estén indicadas en el frente del gabinete como “seguras”, para no perder el proyecto en caso de que haya problemas técnicos.



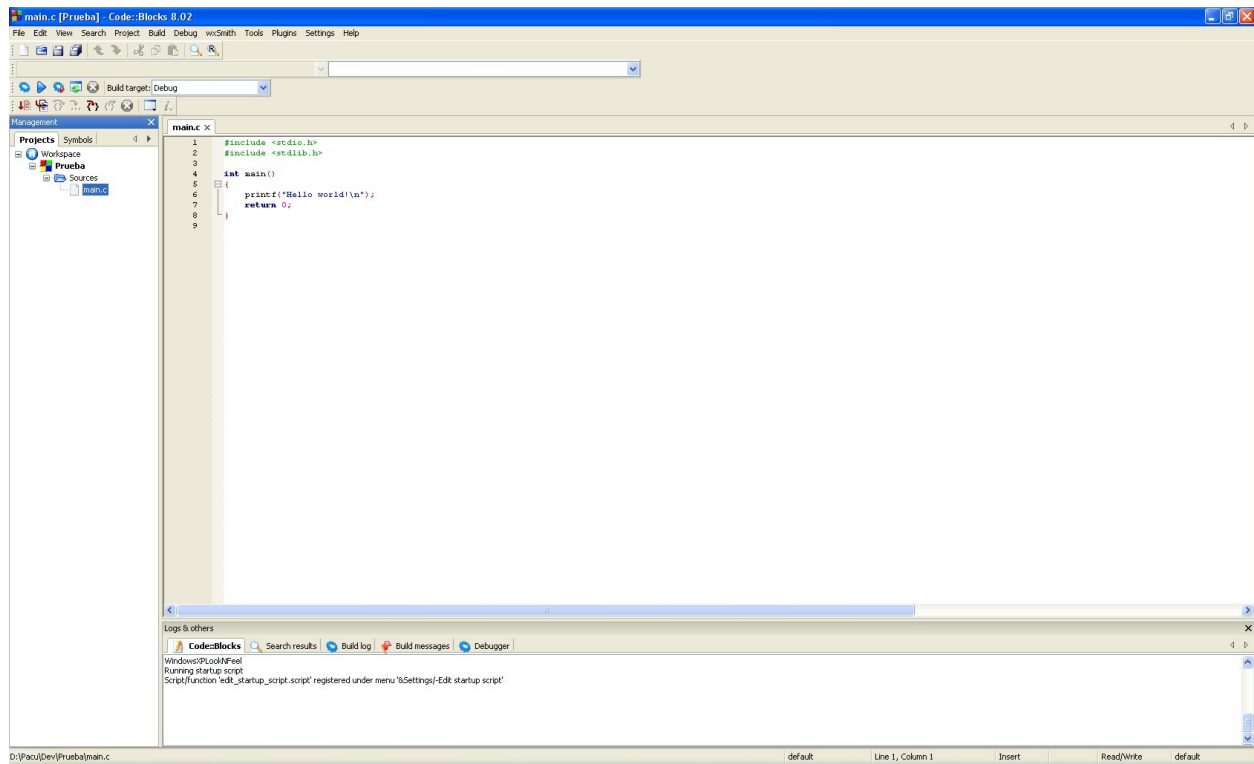
6. Aquí nos indica cuales van a ser las carpetas de nuestra versión final y de las versiones de “depuración” (debug) . Esto no hace falta modificarlo.



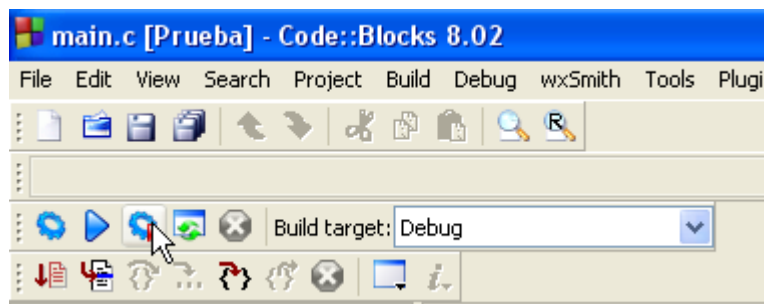
7. Ya tenemos nuestro proyecto armado. La carpeta *Sources* incluye el código generado por la aplicación. En principio, el famoso “hello world”.



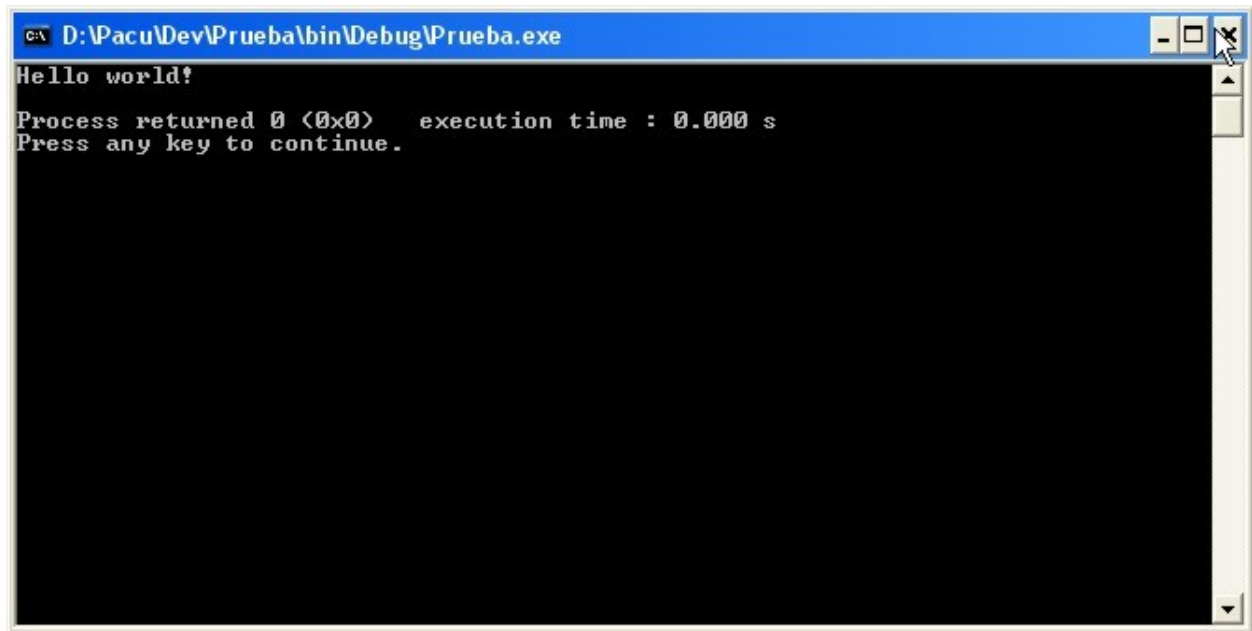
8. Si hacemos doble click en el archivo main.c veremos el código. Lo mismo aplica para cualquier otro que tengamos en el proyecto.



9. Compilar y correr (presionando el ícono del engranaje y el símbolo “play”)



10. ¡Hola mundo!



The image shows a screenshot of a Windows command prompt window. The title bar at the top is blue and contains the text "C:\ D:\PacuDev\Prueba\bin\Debug\Prueba.exe". The main area of the window is black with white text. The text displayed is: "Hello world!", "Process returned 0 (0x0) execution time : 0.000 s", and "Press any key to continue.". The window has standard Windows window controls (minimize, maximize, close) in the top right corner.

```
C:\ D:\PacuDev\Prueba\bin\Debug\Prueba.exe
Hello world!
Process returned 0 (0x0) execution time : 0.000 s
Press any key to continue.
```

Corrimos exitosamente el programa.

Pequeñas grandes diferencias con Borland C++ 3.1

```
main.c x
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main()
5  {
6      printf("Hello world!\n");
7      return 0;
8  }
```

Más allá de lo filosófico y visual, existen diferencias en cuanto al código que manejan estos dos IDE's . Como podemos ver, la más obvia es que el famoso "void main ()" se ha ido. Los programas de hoy en día, cumplen requerimientos de los sistemas operativos modernos. Uno de ellos es que todos los programas tienen cierta información sobre cómo finalizaron su ejecución. Si bien esto es algo que van a aprovechar más adelante en materias como Laboratorio III, es importante saber que todo programa que ha terminado "exitosamente", retorna el valor 0 (cero) al sistema operativo. Quienes estén más familiarizados con entornos Java conocerán la sentencia `System.exit(0)` para indicar la salida forzosa del programa y el estado con el que salió. Los estados no nos importan en este momento, pero lo que si vale la pena rescatar es que "void main ()" fue reemplazado por el la función "int main ()", que devuelve "return 0" el estado final del programa cuando alcanza la última sentencia.

Otra diferencia está en las librerías que el compilador Gcc de **Code::Blocks** invoca. Vemos que tenemos las líneas:

```
#include <stdio.h>

#include<stdlib.h>
```

¿Qué paso con la librería `conio.h`? En este caso no la necesitamos, aunque igualmente podemos agregarla si así lo necesitáramos por compatibilidad. También está disponible la librería **string.h** que utilizamos para tratar arreglos de tipo char como cadenas de caracteres.

Diferencia de mensajes de error y advertencias

Code::Blocks posee un compilador distinto a Borland C++, este compilador no sólo es diferente en su funcionamiento interno, sino también en los mensajes que pone en pantalla. Las diferencias no son grandes, y un mensaje que expresa un mismo error en ambos IDE's no presenta términos que sean indescifrables. De todas formas, el compilador Gcc es usado por miles de programadores de la comunidad de software libre, lo que nos garantiza varios millares de respuestas ingresando el mensaje de error en Google o cualquier otro buscador.

Preguntas frecuentes

¿Por qué no compilan mis programas de borland C++ en Code::Blocks?

Reemplaza el `void main` por `int main`.

¿Cómo paso un programa escrito en Code::Blocks a Borland C++?

Para pasar a borland, tienes que cambiar la extensión del archivo `.c` a `.cpp` y luego abrirlos con el borland C++

Cuando paso a Borland mis programas suelen comportarse de manera extraña.

Verifica la extensión del archivo. Si es correcta, cierra el Code::Blocks mientras ejecuta el Borland.

Fuentes:

www.codeblocks.org

<http://es.wikipedia.org/wiki/CodeBlocks>

Autores:

Fabiana Daian

edaian@palermo.edu

Francisco Gindre

francisco.gindre@gmail.com

Contenidos

Introducción a Code::Blocks.....	1
 Instalación.....	1
 Crear un proyecto para empezar a programar.....	8
 Pequeñas grandes diferencias con Borland C++ 3.1.....	17
 Diferencia de mensajes de error y advertencias.....	18
Fuentes:.....	20