Primeros pasos para utilizar el editor Code::Blocks para C

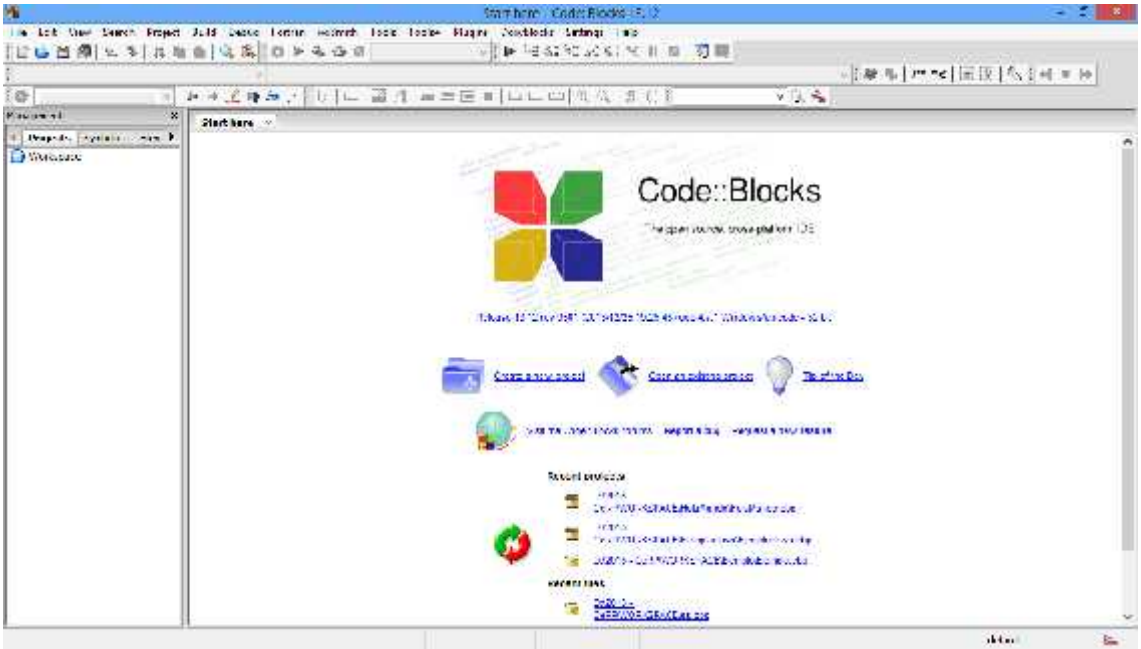
En esta edición del curso se utilizará un entorno para C llamado *Code::Blocks* (de libre distribución).
Hay diferentes versiones para usar en distintos sistemas operativos (Windows, Linux, etc.).
Este documento propone una guía de primeros pasos para empezar a utilizar *Code::Blocks*.

1. *Obtención e Instalación de Code::Blocks para C*
2. *Creación de un Project en Code::Blocks*
3. *Creación de un archivo .h en Code::Blocks*
4. *Creación de un archivo .cpp en Code::Blocks*
5. *Compilación, Linkedición y Ejecución en Code::Blocks*
6. *Uso del debugger en Code::Blocks*

1. Obtención e Instalación de Code::Blocks para C

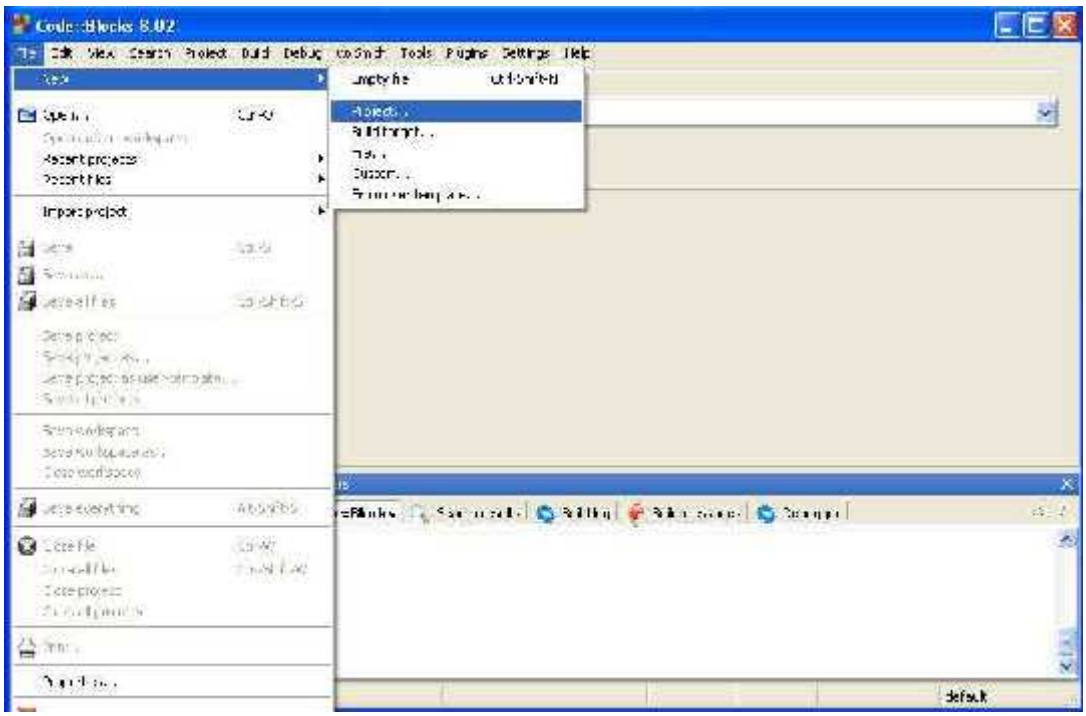
El instalador puede obtenerse desde el sitio http://www.codeblocks.org/downloads/binaries.
Desde allí es posible descargar instaladores para distintos sistemas operativos. Todos los pasos descritos en este documento son válidos para dicha versión. El instalador correspondiente se llama codeblocks-13.12mingw-setup.exe (97.9 MB). Para este curso, basta con realizar la instalación por defecto que sugiere el programa instalador al ejecutarlo.

Una vez instalado *Code::Blocks* al abrirlo vemos la siguiente pantalla:

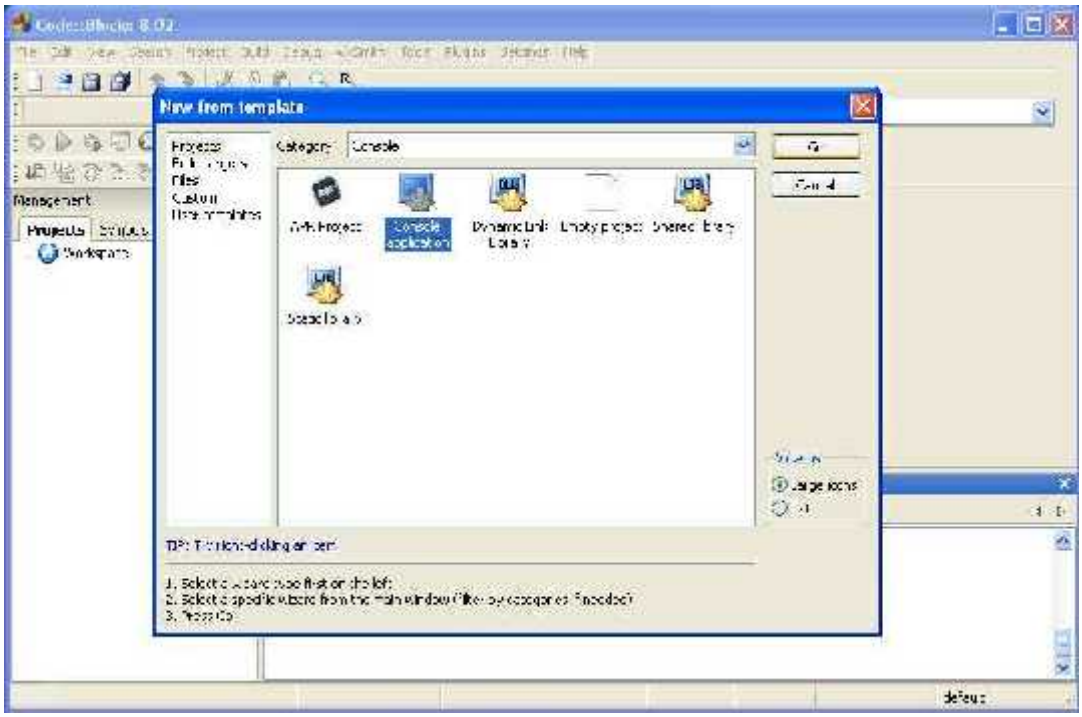


2. Creación de un Project en Code::Blocks

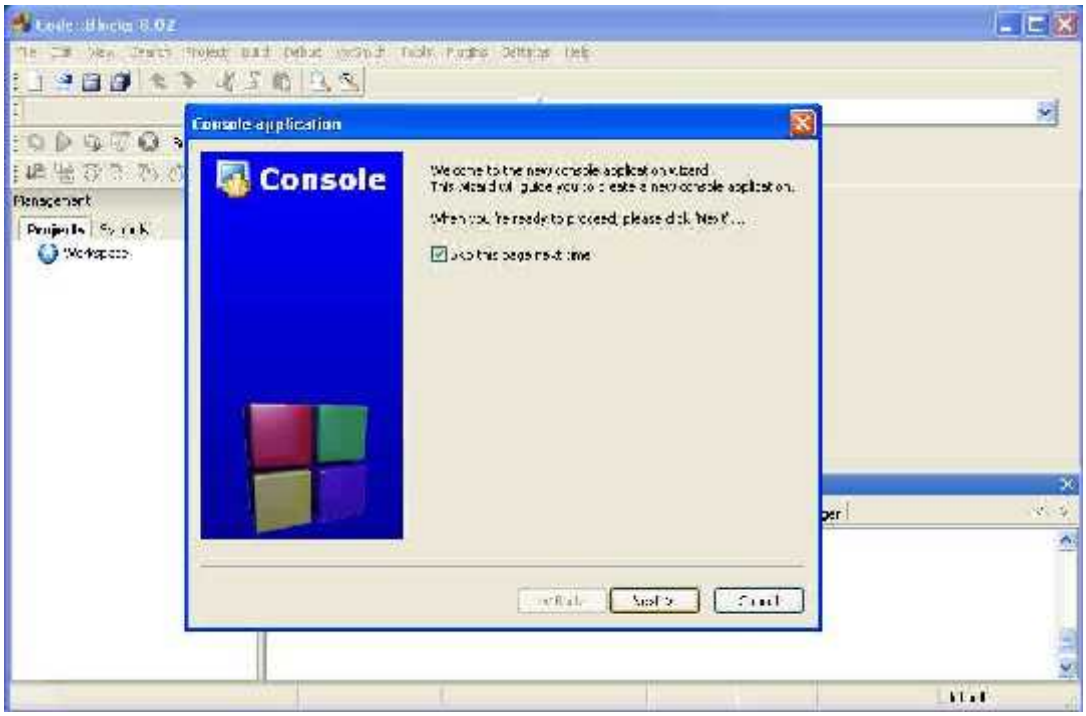
Para crear un project de trabajo en Code::Blocks, elegimos la opción *File -> New -> Project*:



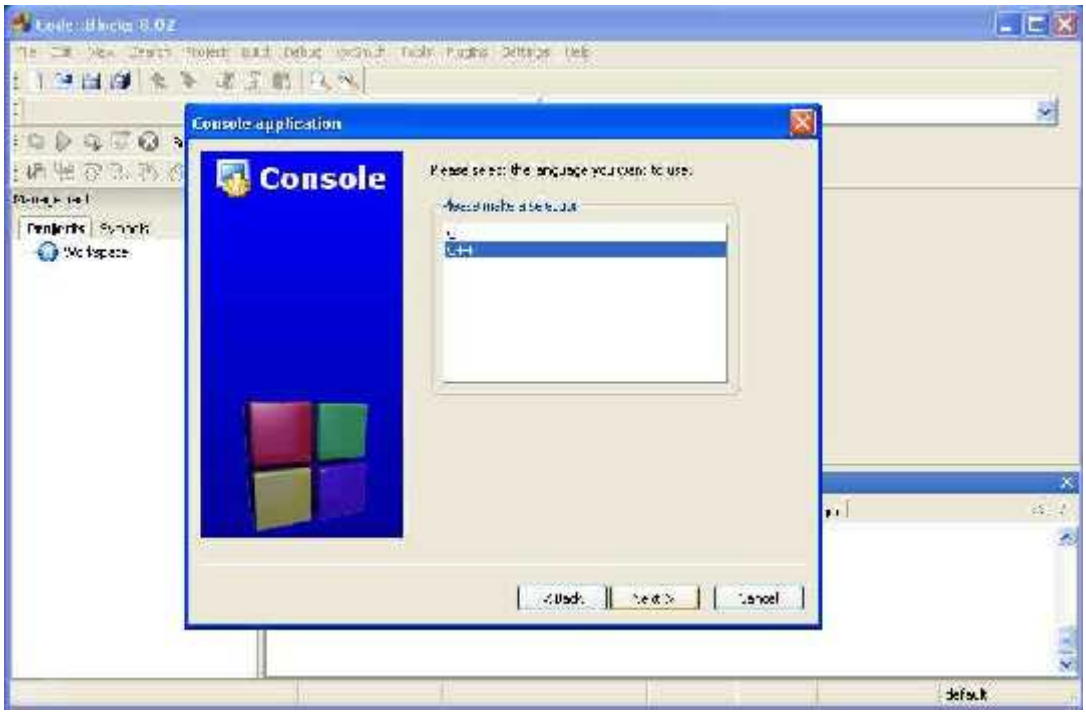
A continuación elegimos *Console Application* y presionamos *Go*:



Nos aparece la primera ventana de un asistente que nos guiará en la construcción del Project (podemos pedirle que omita dicha primera ventana la siguiente vez que creemos un Project):

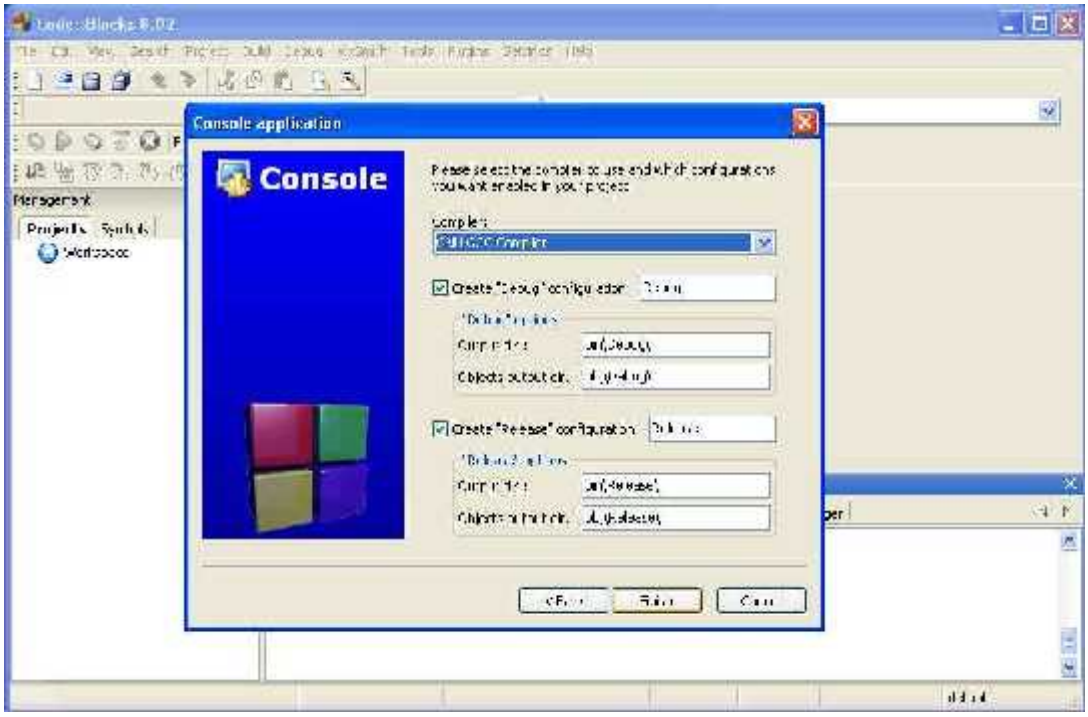


Le indicamos que queremos C++ como lenguaje de programación para el Project (si ponemos lenguaje C habrá ciertas características, tales como el pasaje por referencia, que quizás no se puedan utilizar correctamente):

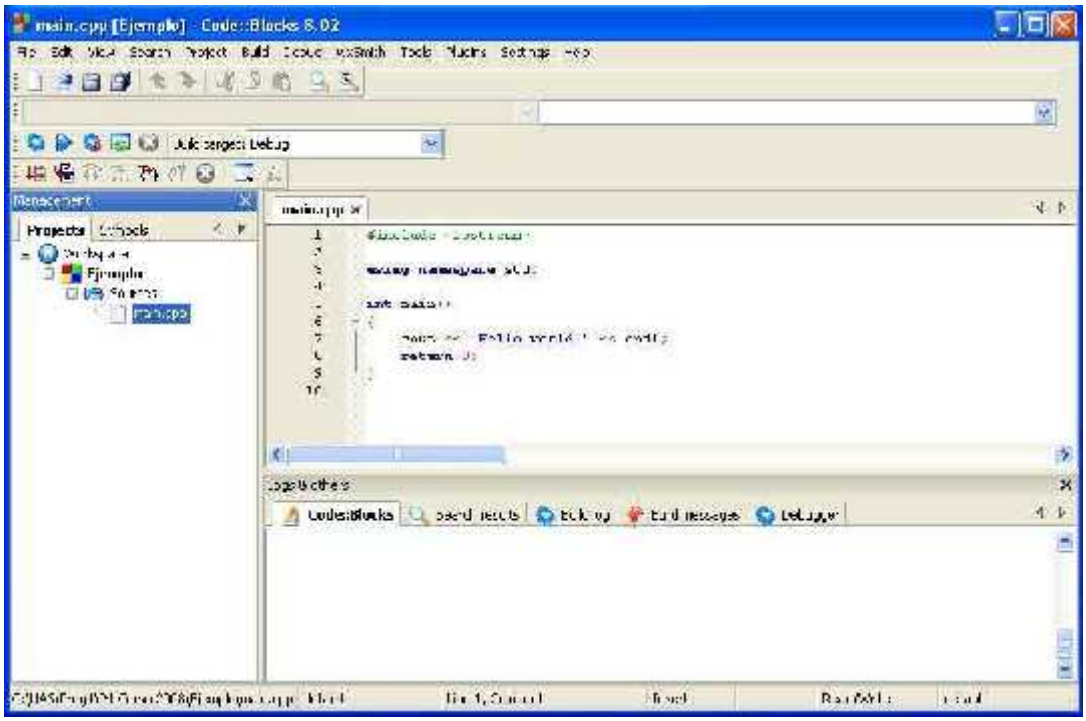


Luego le ponemos título (nombre) al Project e indicamos en qué carpeta queremos guardarlo

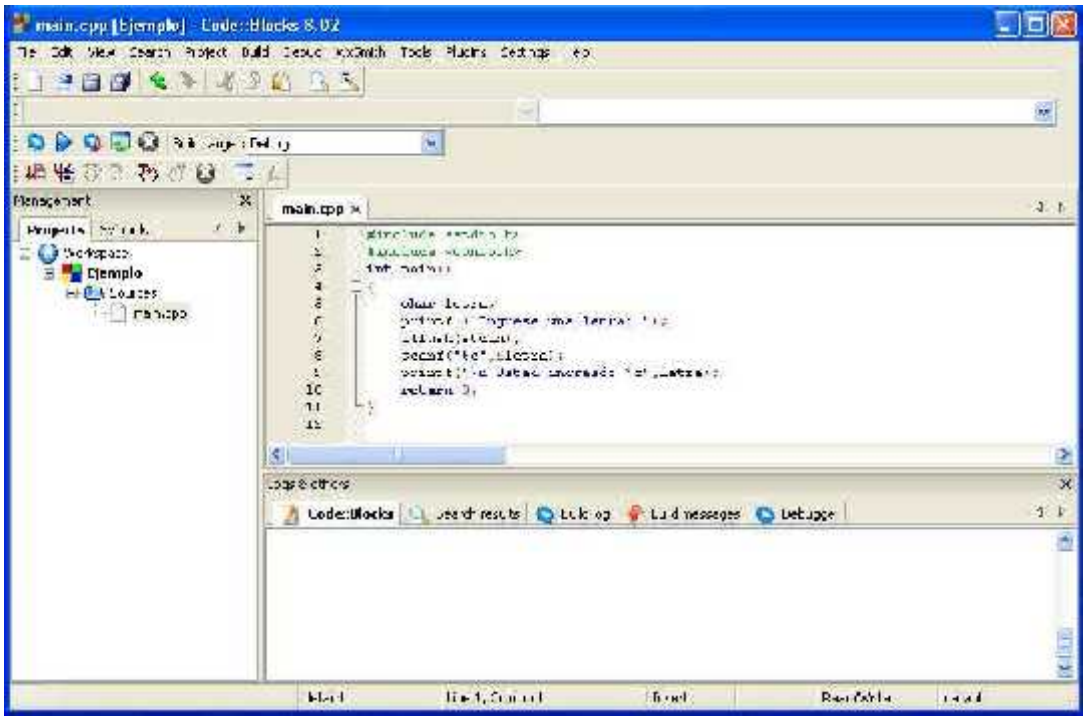
Ahora debemos indicarle qué compilador de C++ queremos usar en el Project, ya que podría haber más de uno instalado en nuestra PC. En este curso elegimos *GNU GCC Compiler*, que es el compilador que viene junto con el instalador de *Code::Blocks*. También le tildamos las opciones *Create "Debug"* y *Create "Release"* configuration:



Una vez presionado *Finish*, nos aparece nuestro Project bajo la pestaña de proyectos. El Project ya nos incorpora automáticamente el programa *main* con un código fuente por defecto.

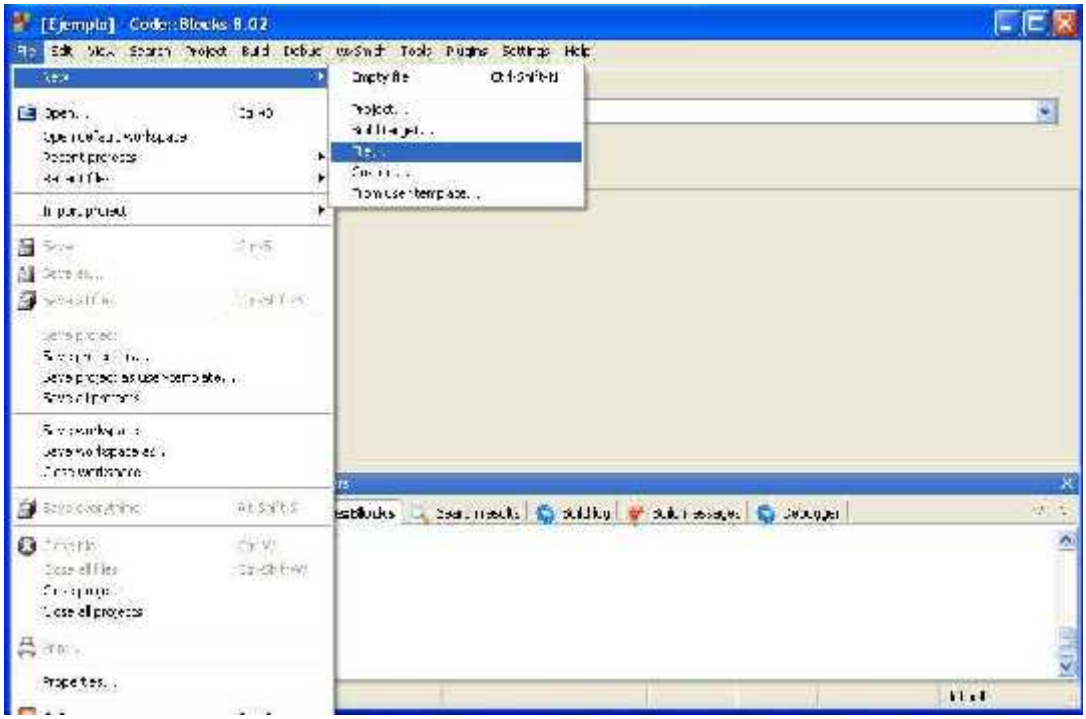


Observación: En *Code::Blocks*, el programa *main* habitualmente retorna int en lugar de void. Podemos cambiar el código fuente por defecto del *main* por otro escrito por nosotros:

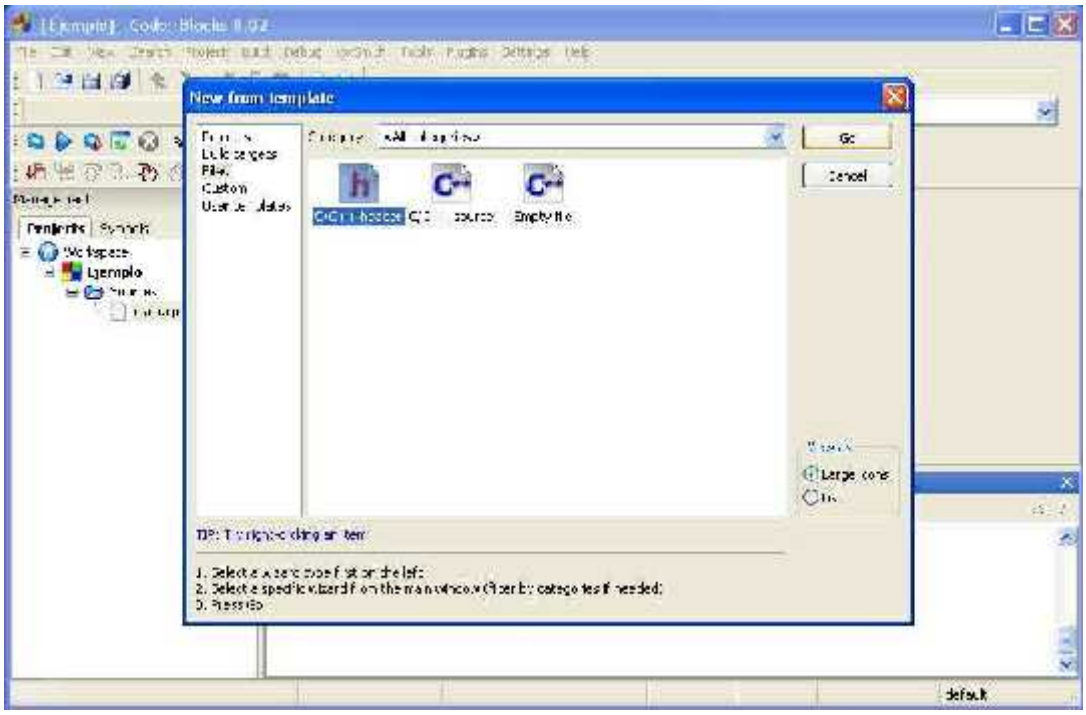


3. Creación de un archivo .h en Code::Blocks

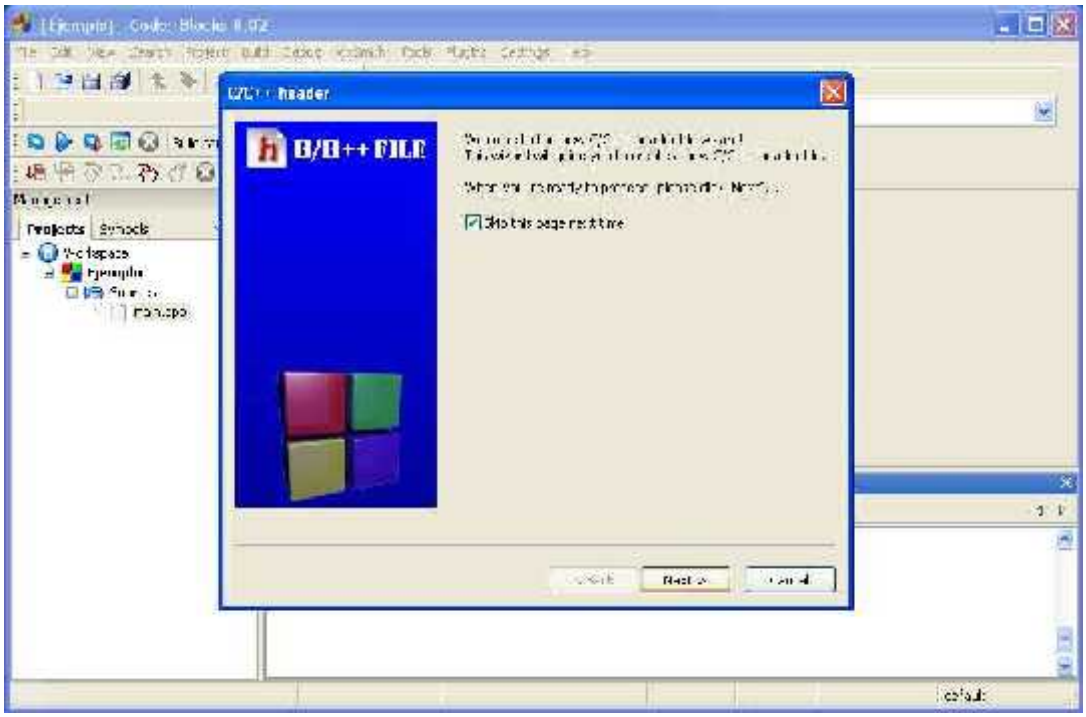
Una vez creado nuestro Project de trabajo, veremos cómo crear un archivo .h para un módulo. Elegimos la opción *File New File,…:*



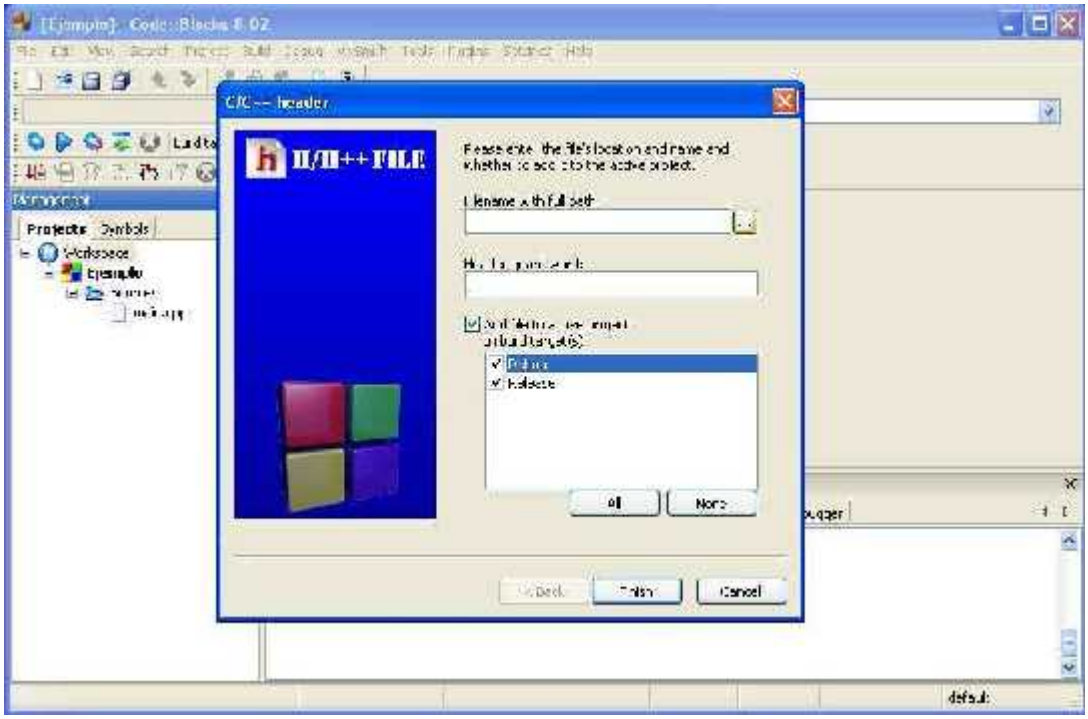
A continuación elegimos *C/C++ header* y presionamos *Go*:



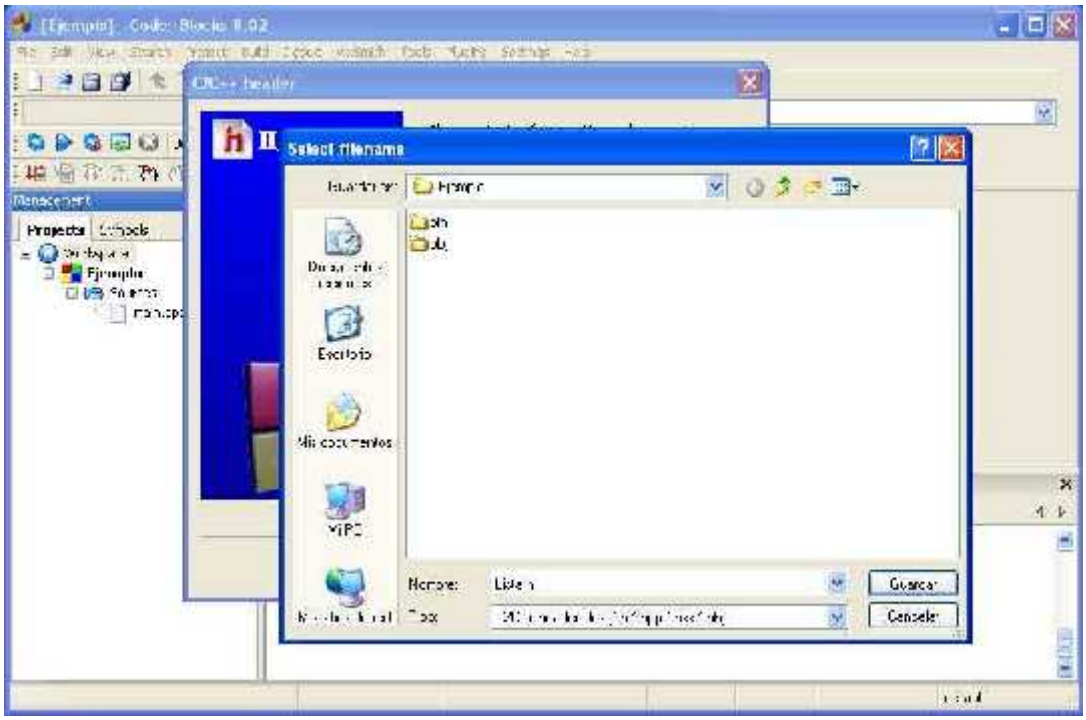
Nos aparece la primera ventana de un asistente que nos guiará en la construcción del archivo .h (podemos pedirle que omita dicha primera ventana la siguiente vez que creemos un .h):



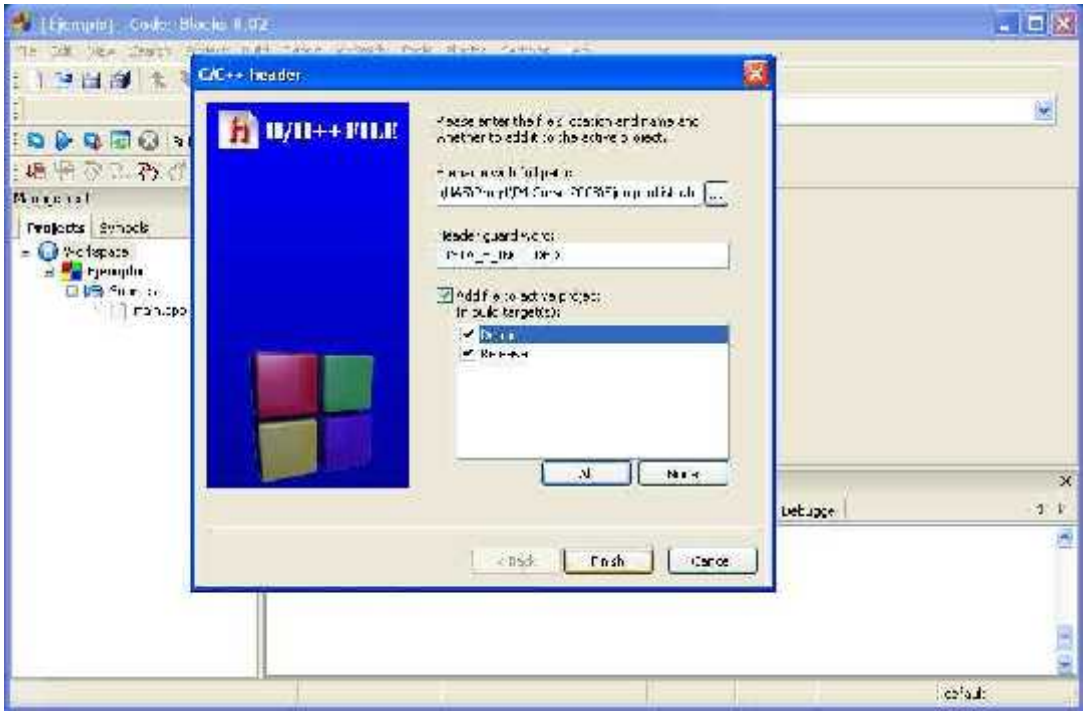
Junto al campo de texto *Filename with full path* presionamos el botón para elegir donde guardar nuestro archivo .h. Previamente tildamos las opciones *Debug* y *Release*:

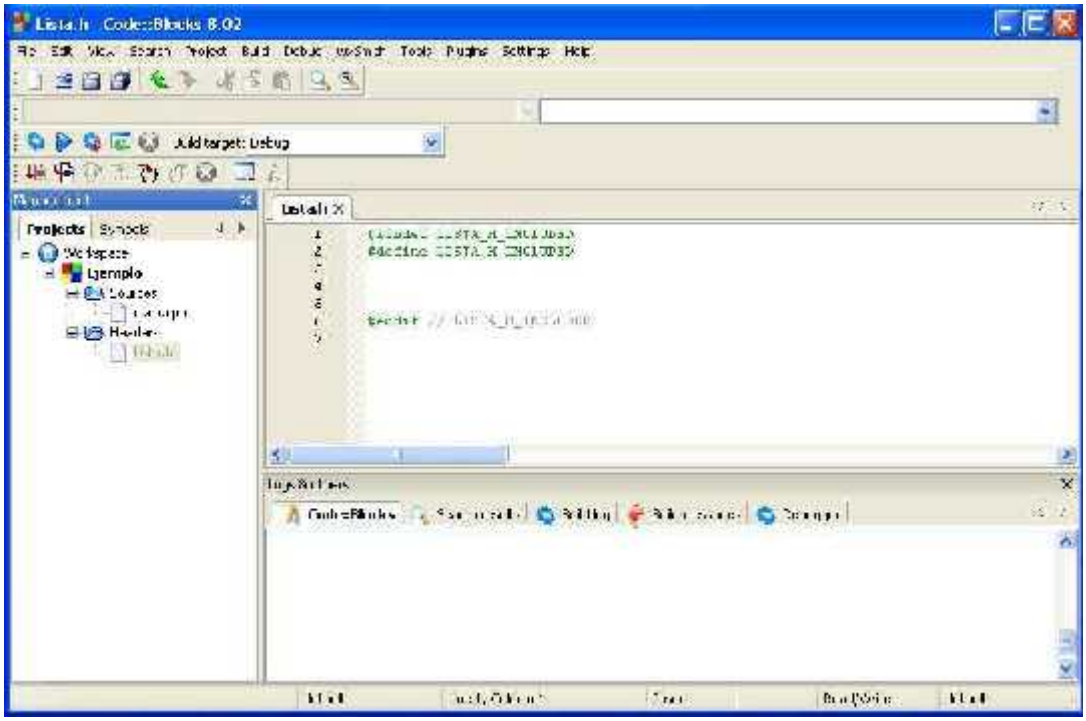


Le ponemos un nombre a nuestro archivo .h y lo guardamos en la carpeta de nuestro Project:

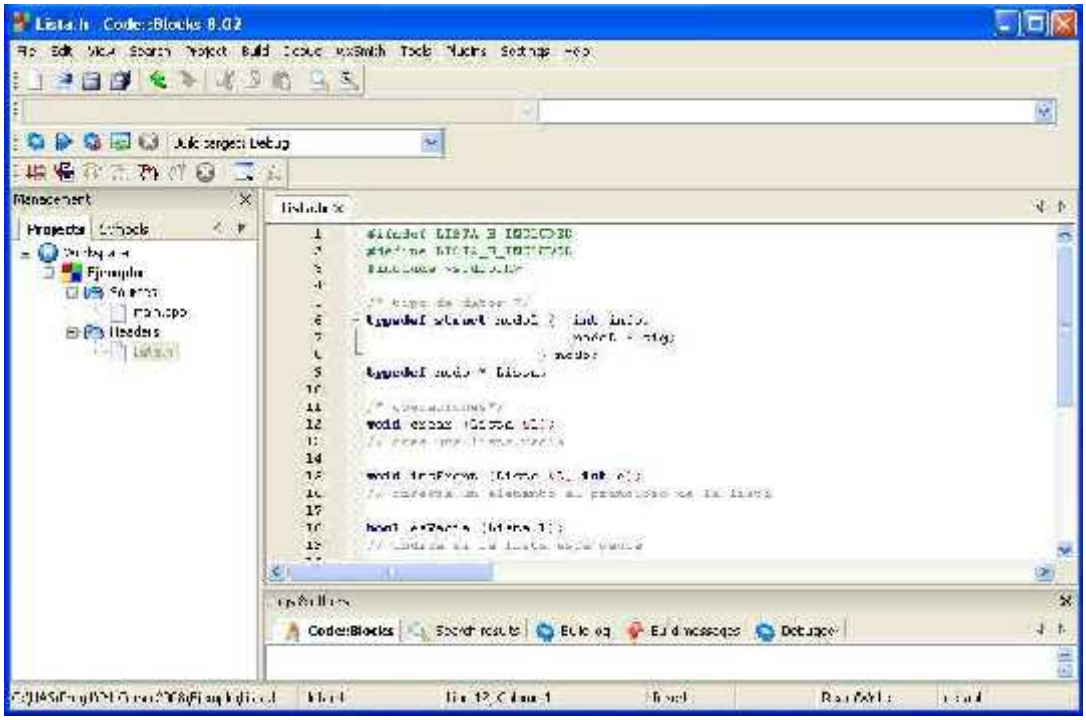


Hecho esto, en el cuadro de texto *Header Guard Word* nos aparece la palabra que pondrá automáticamente en las directivas #ifndef y #define del archivo .h Luego presionamos *finish*: El archivo .h creado aparece bajo *Headers* en el Project y nos despliega el código fuente del archivo, que momentáneamente contiene sólo las directivas #ifndef, #define y #endif:



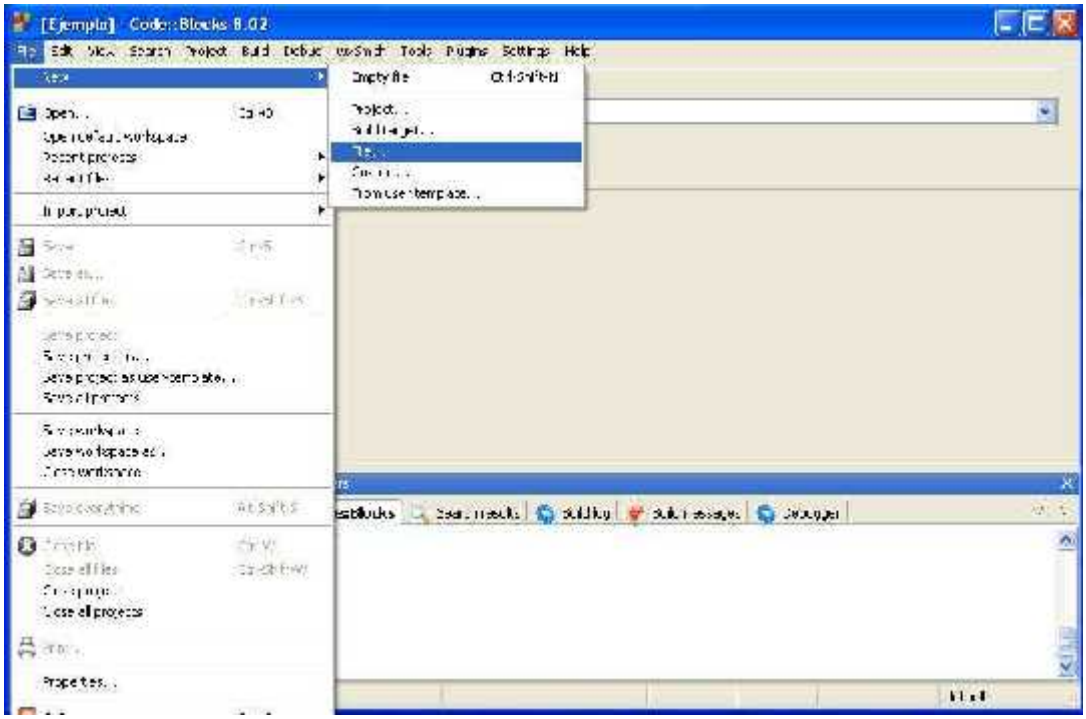


Sólo nos resta rellenar el archivo .h con las inclusiones, los tipos de datos y los cabezales de las operaciones correspondientes al módulo (nótese en el tercer cabezal de la imagen que *bool* es un tipo primitivo predefinido en *Code::Blocks*).

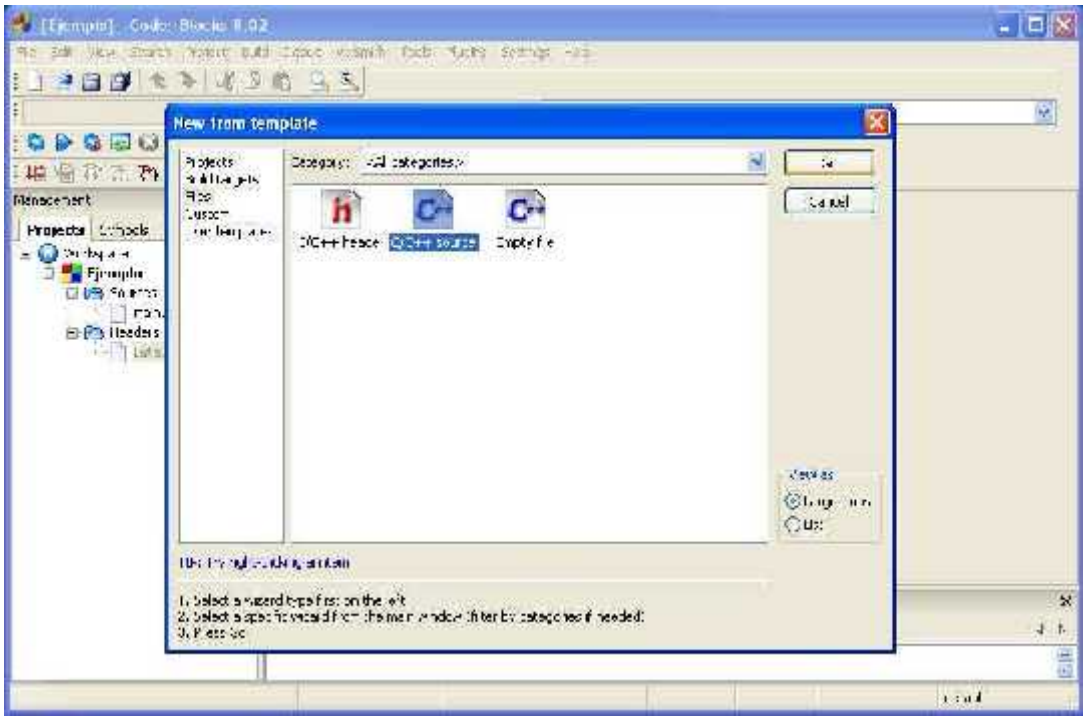


4. Creación de un archivo .cpp en Code::Blocks

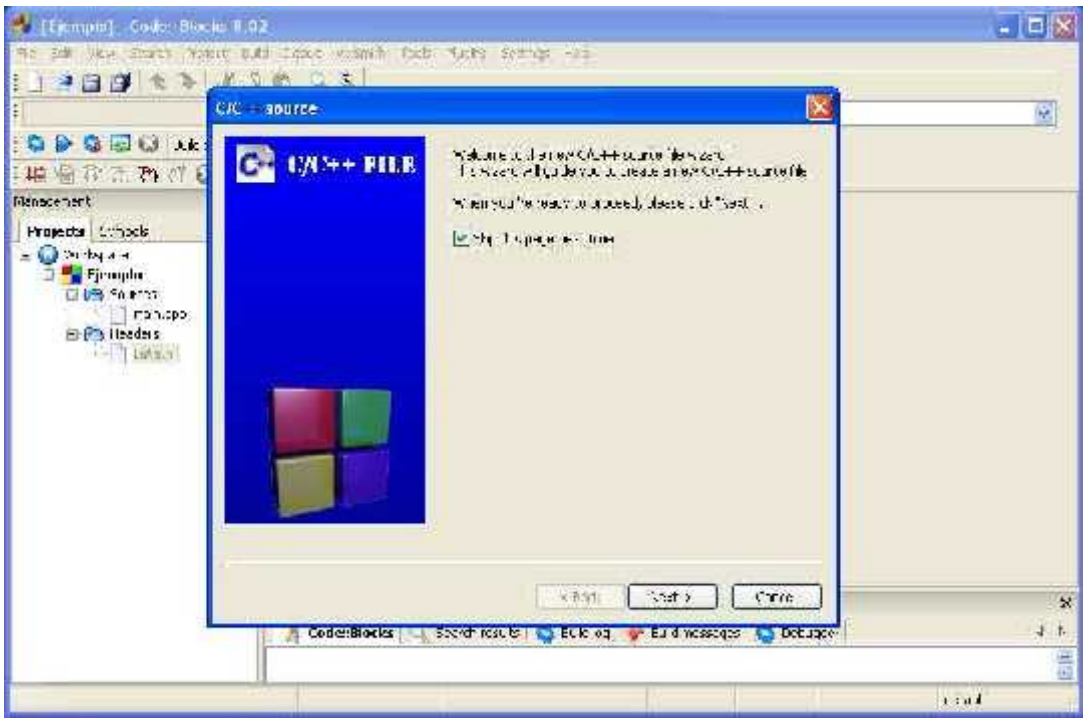
Una vez creado el archivo .h para nuestro módulo, veremos cómo crear su correspondiente archivo .cpp. Elegimos la opción *File -> New -> File*



A continuación elegimos *C/C++ source* y presionamos *Go*:

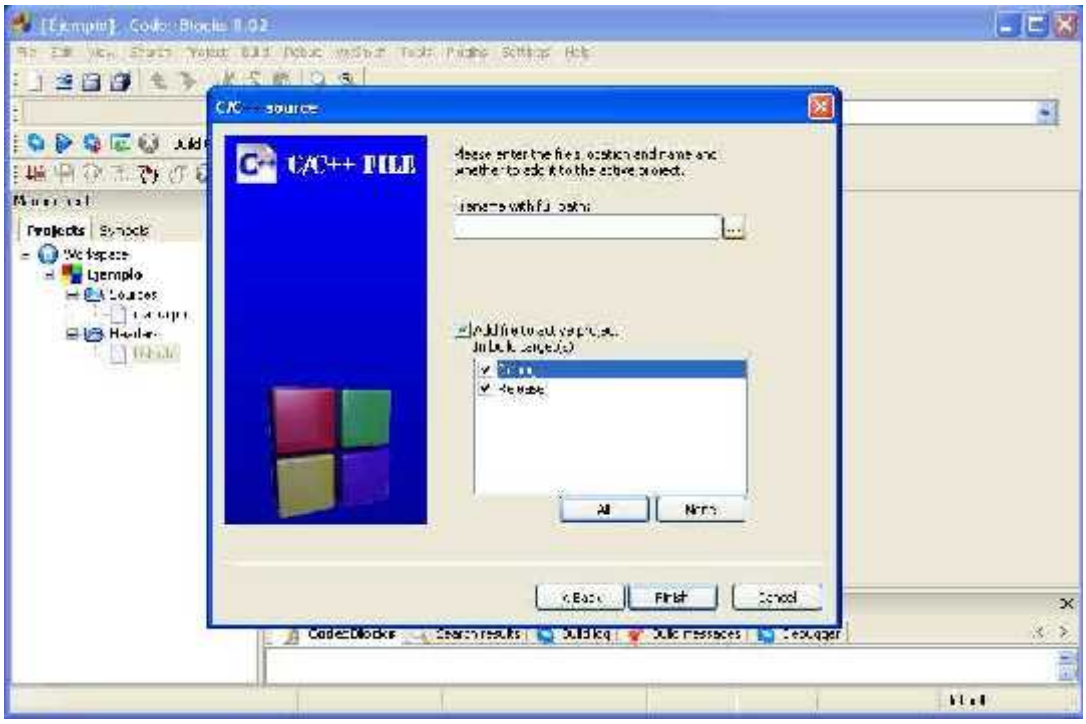


Nos aparece la primera ventana de un asistente que guiará en la construcción del archivo .cpp (podemos pedirle que omita dicha primera ventana la siguiente vez que creemos un .cpp)

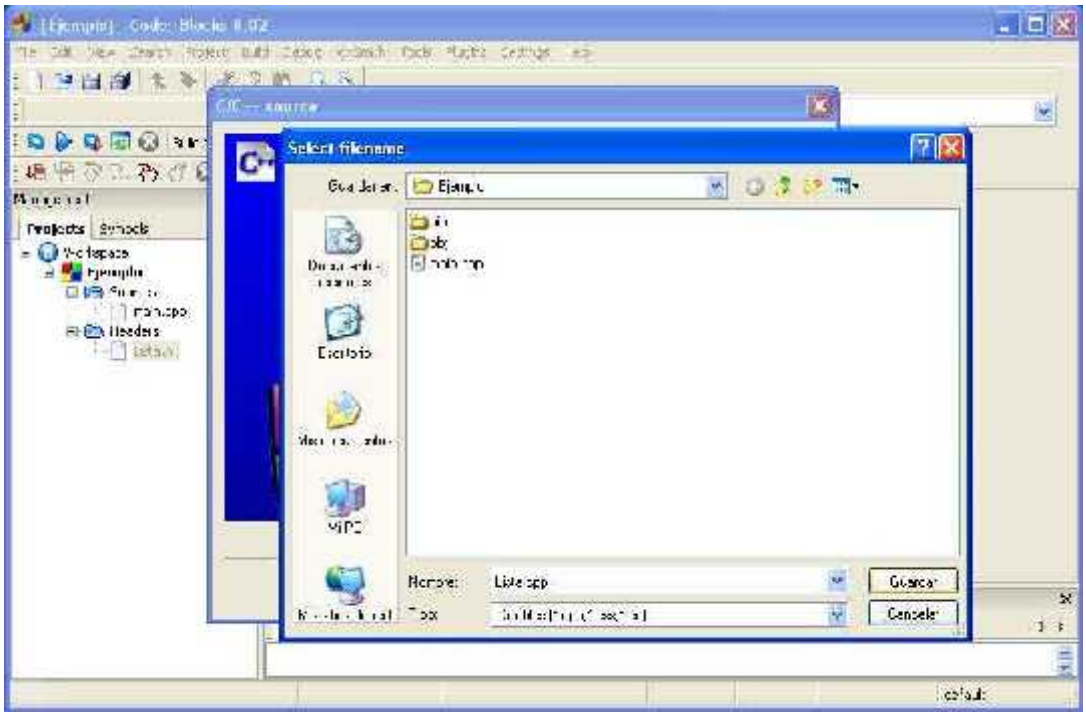


Al igual que como hicimos cuando creamos el Project, le indicamos que queremos C++ como lenguaje de programación para el archivo .cpp:

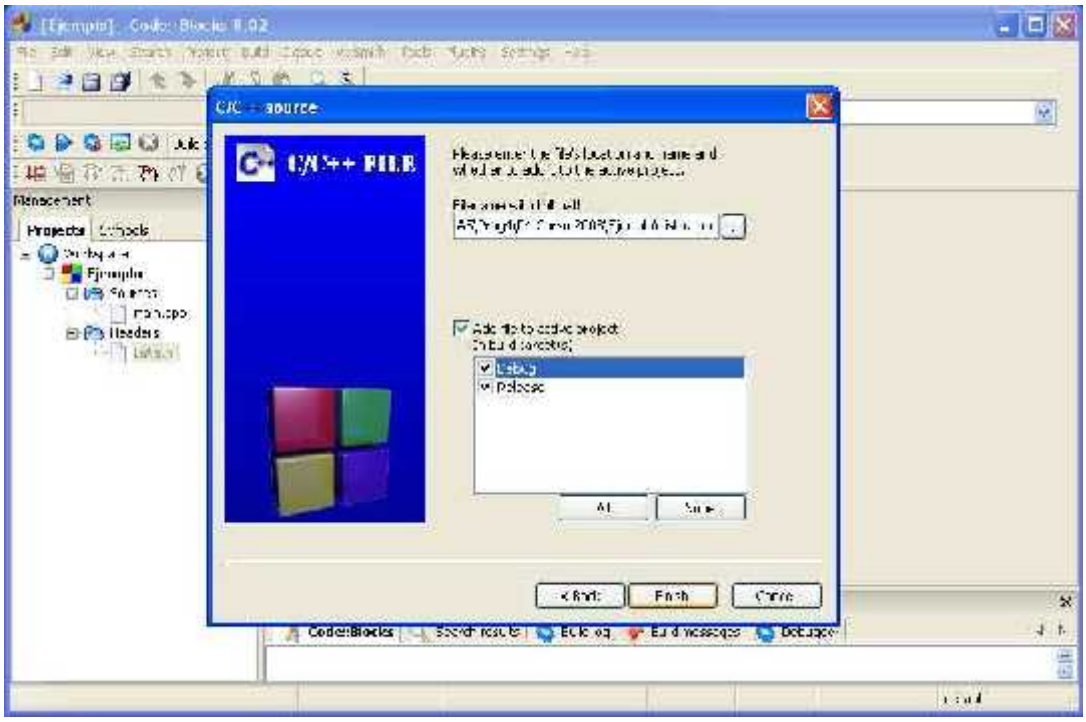
Junto al campo de texto *Filename with full path* presionamos el botón para elegir donde guardar nuestro archivo .cpp. Previamente tildamos las opciones *Debug* y *Release*



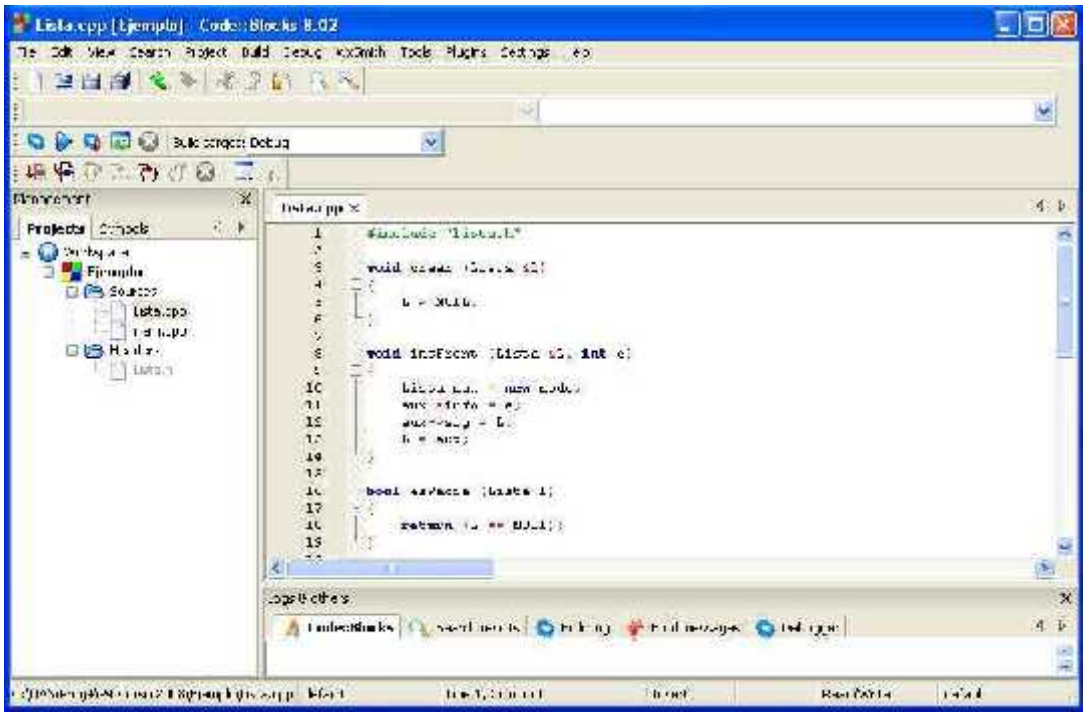
Le ponemos un nombre a nuestro archivo .cpp y lo guardamos en la carpeta del Project:



Hecho esto, nos muestra la ruta completa de donde queda guardado nuestro archivo .cpp. Luego presionamos *finish*:

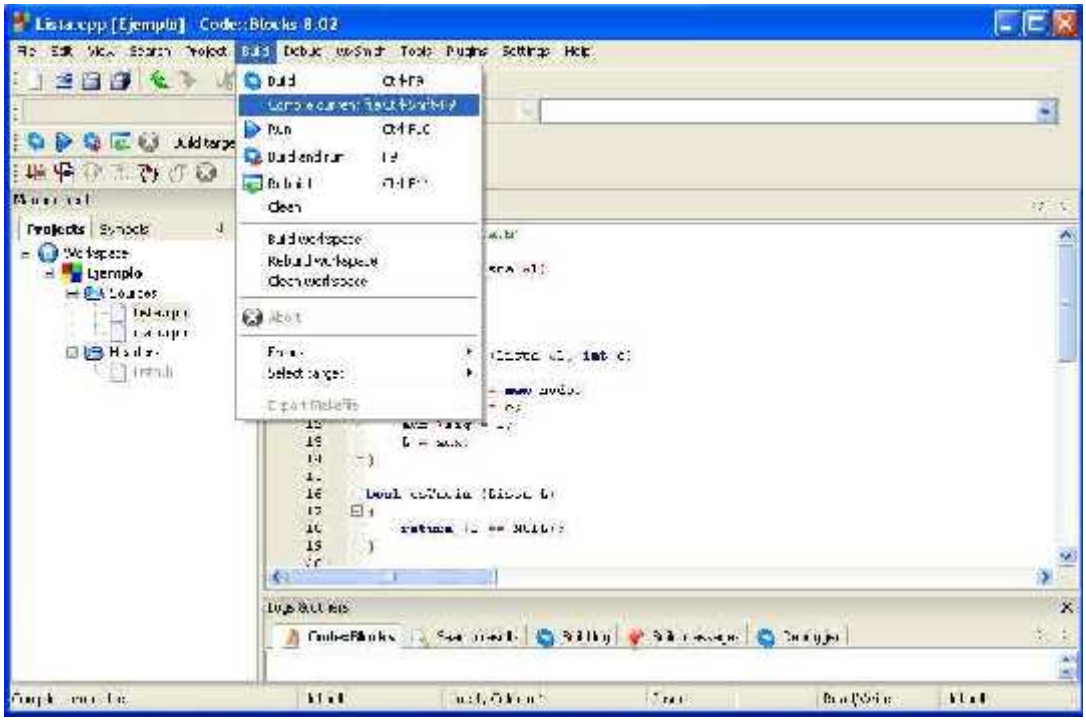


El archivo .cpp creado aparece bajo *Sources* en el Project. Sólo nos resta rellenar el archivo con la inclusión del correspondiente .h y la implementación de las operaciones pertenecientes al módulo:

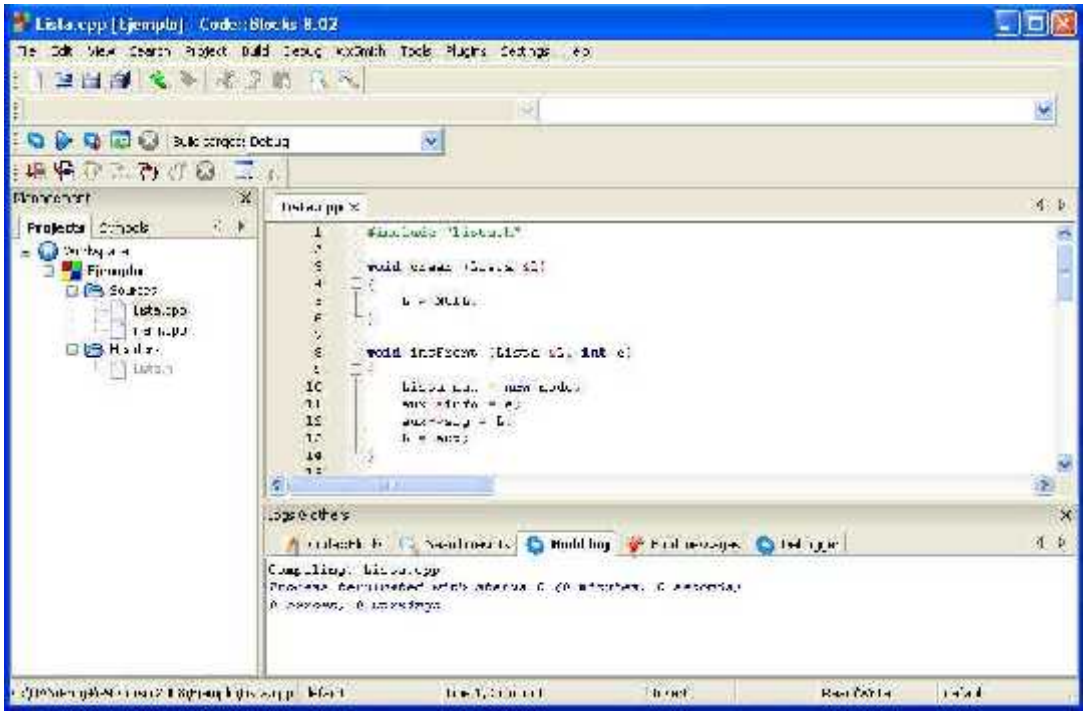


5. Compilación, Linkedición y Ejecución en Code::Blocks

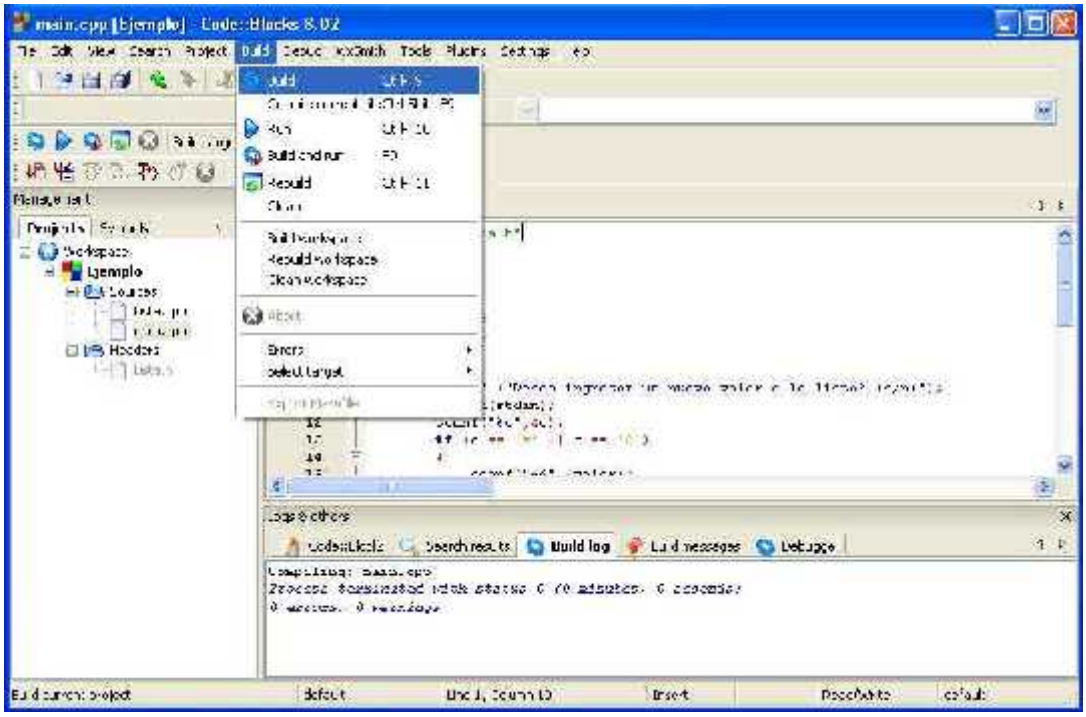
Una vez creados los archivos .h y .cpp de nuestro Project, veremos cómo compilar, linkeditar y ejecutar. Para compilar, debemos posicionarnos sobre cada .cpp y compilarlo por separado(los .h no se compilan). Para ello, elegimos *Build → Compile Current File:*



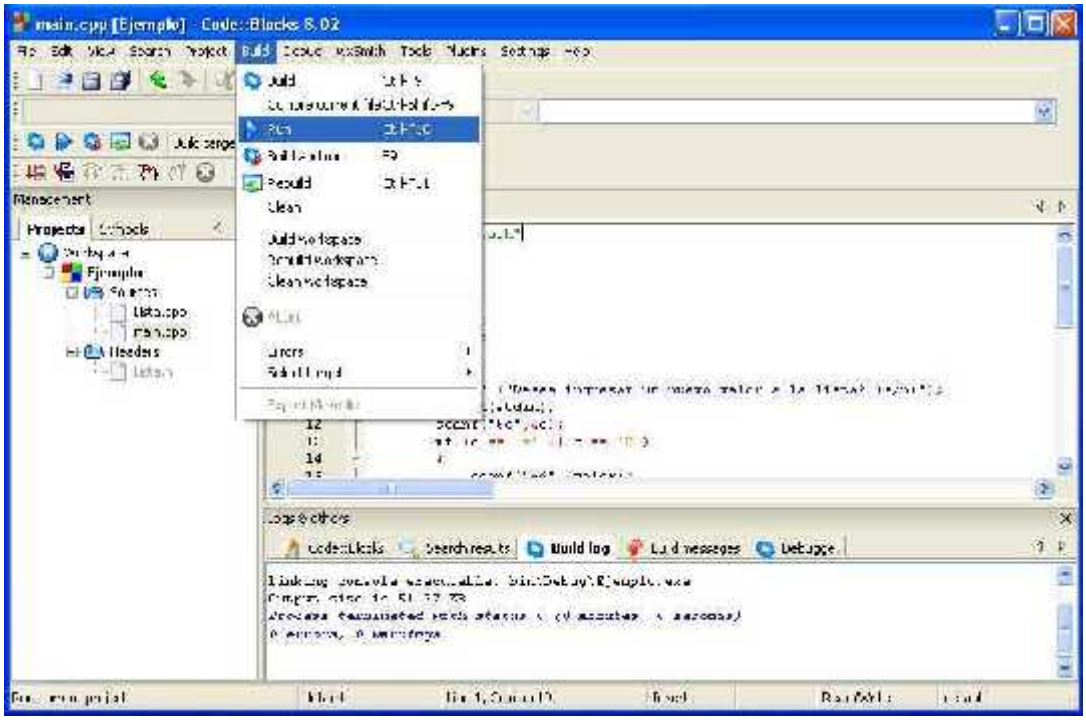
El resultado de la compilación aparece bajo la pestaña *Build Log*:



Una vez compilados todos los .cpp de nuestro Project (incluido el *main*), procedemos a linkeditarlos. Para ello, elegimos *Build* - *Build*:

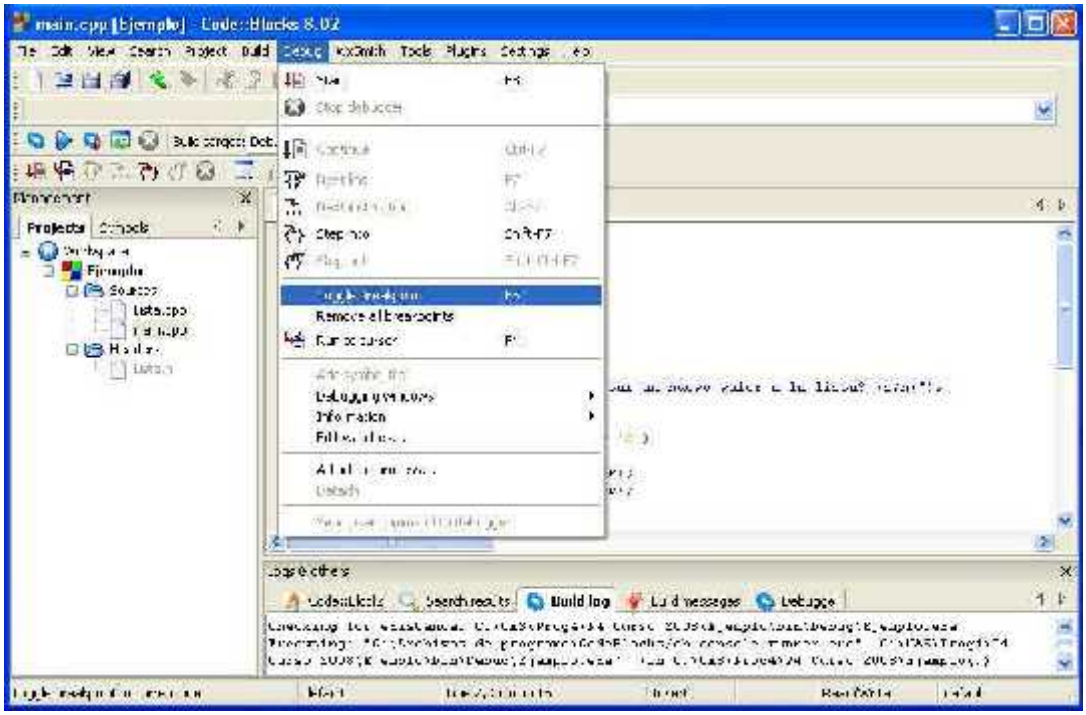


El resultado de la linkedición aparece bajo la pestaña *Build Log*. Nos resta solamente ejecutar el Project, para lo cual elegimos *Build -> Run*:

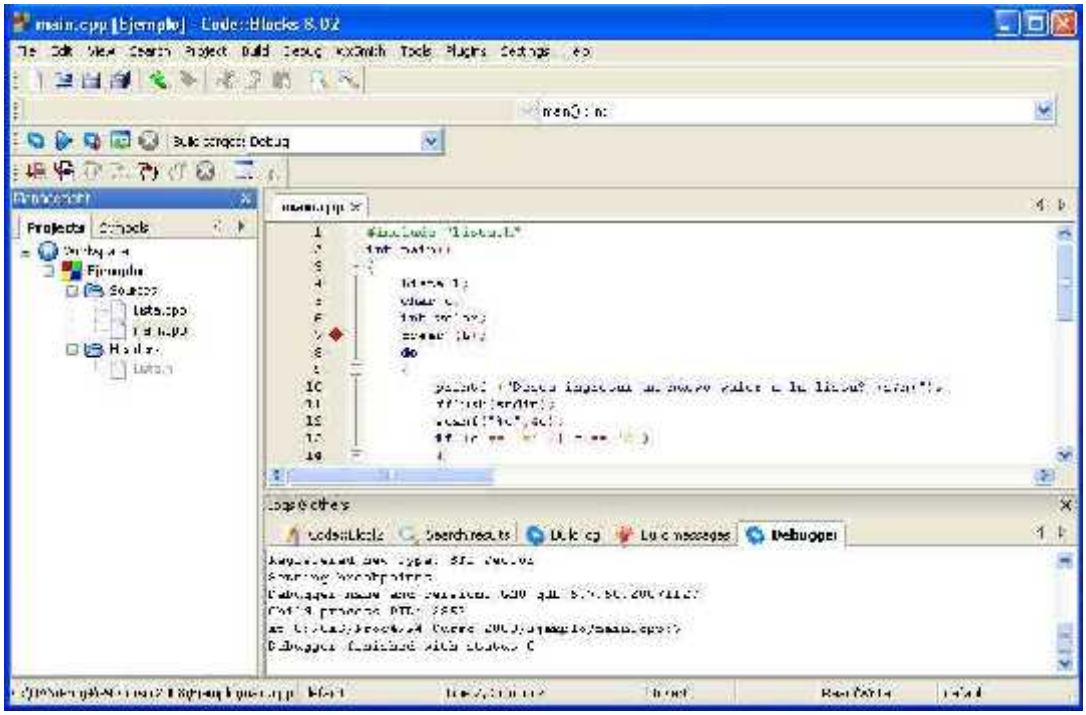


6. Uso del debugger en Code::Blocks

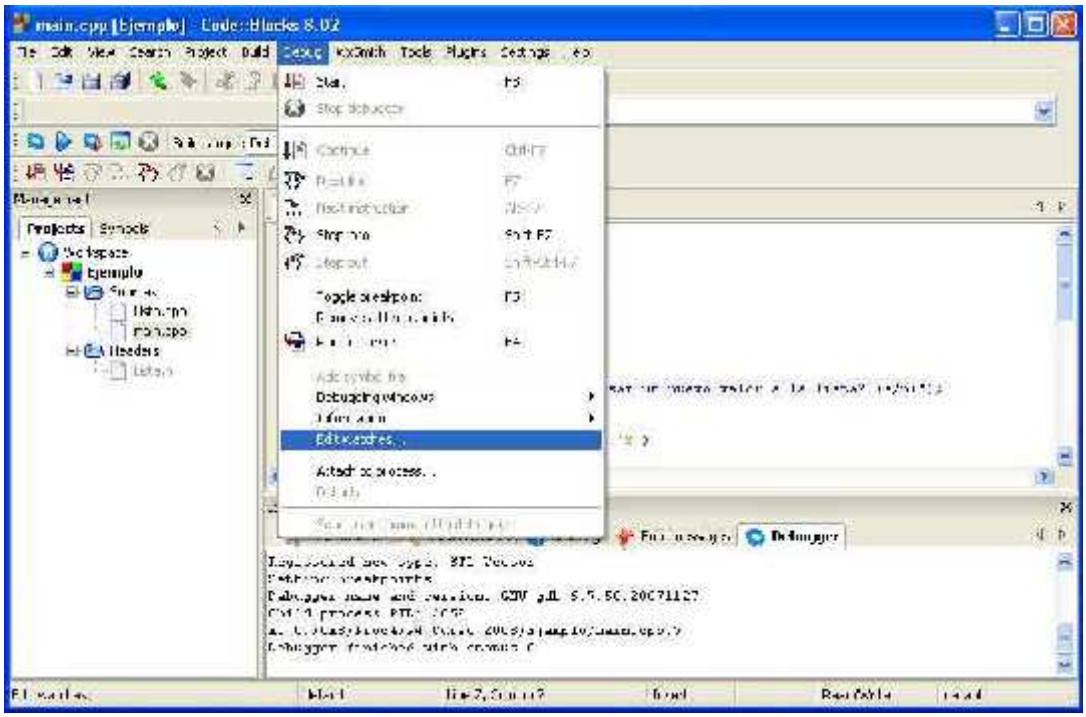
Veremos cómo usar el debugger en *Code::Blocks*. Primero elegimos la línea del programa apartir de donde queremos debuggear. Para ello, elegimos *Debug Toggle Breakpoint:*



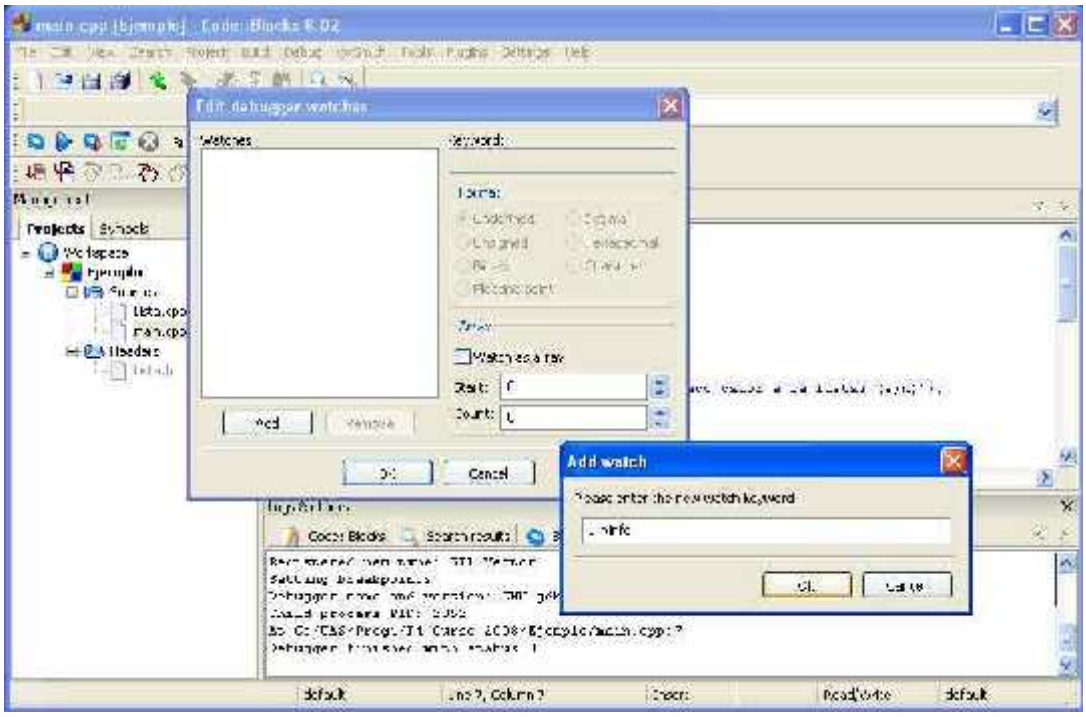
Nos aparece una marca roja junto a la línea de código a partir de donde debuggear:



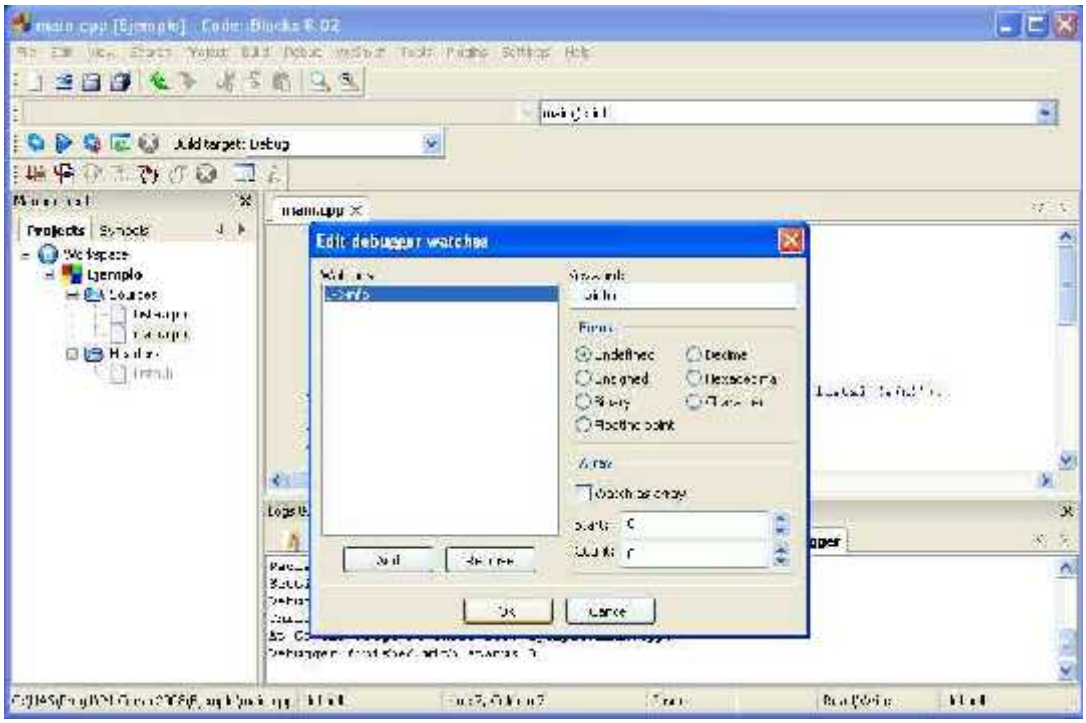
Podemos indicar las variables cuyos valores deseamos inspeccionar durante el debugging. Para ello, elegimos *Debug → Edit Watches*



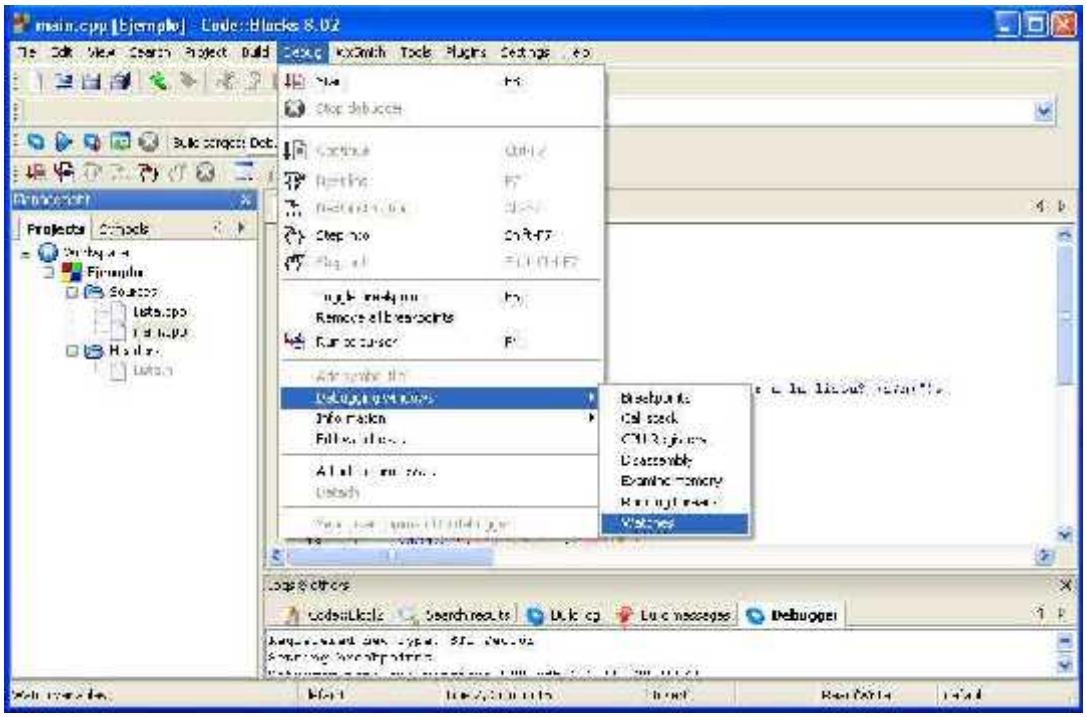
Presionamos *add* e indicamos el nombre de la variable, campo o parámetro cuyo valor deseamos inspeccionar y presionamos *ok*:



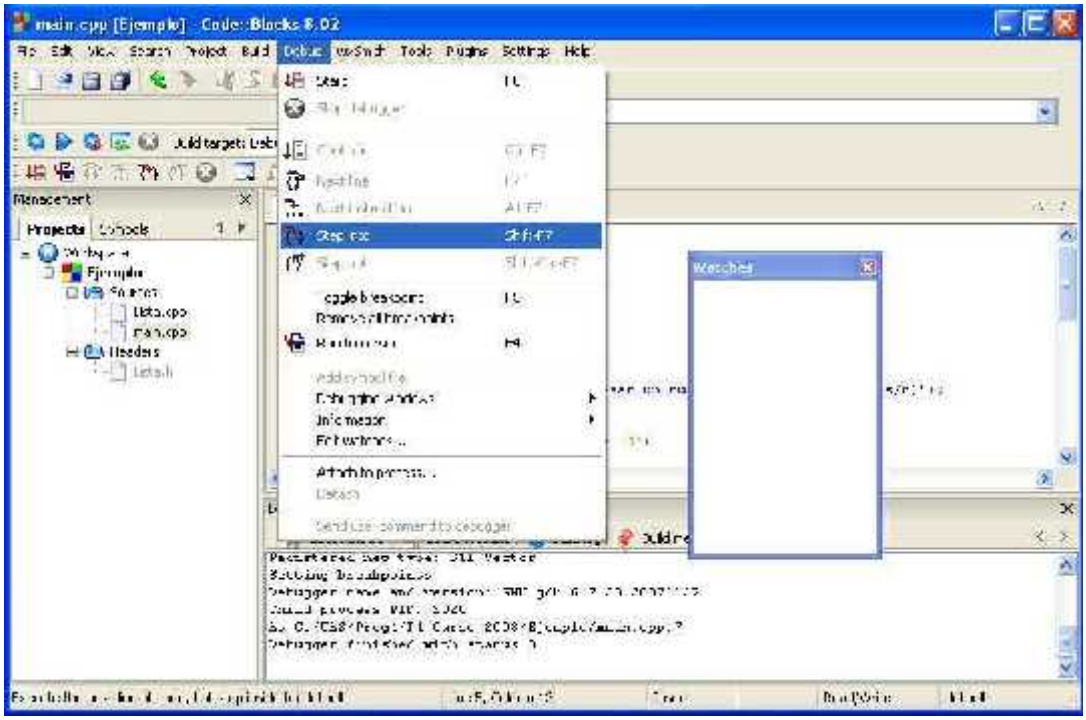
La variable agregada nos aparece en *Watches*. Podemos repetir este proceso tantas veces como deseemos, con todas las variables, campos o parámetros que queramos inspeccionar:



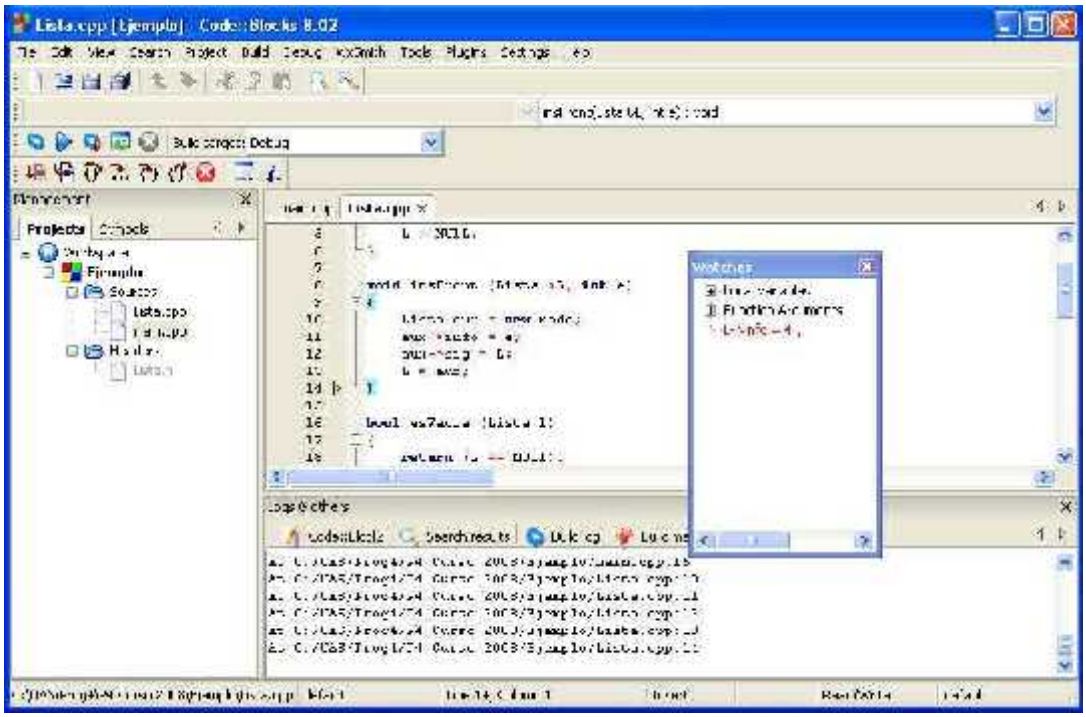
Durante la ejecución con el debugger, si queremos ver cómo cambian los valores de las variables agregadas, debemos elegir *Debug* → *Debugging Windows* → *Watches*:



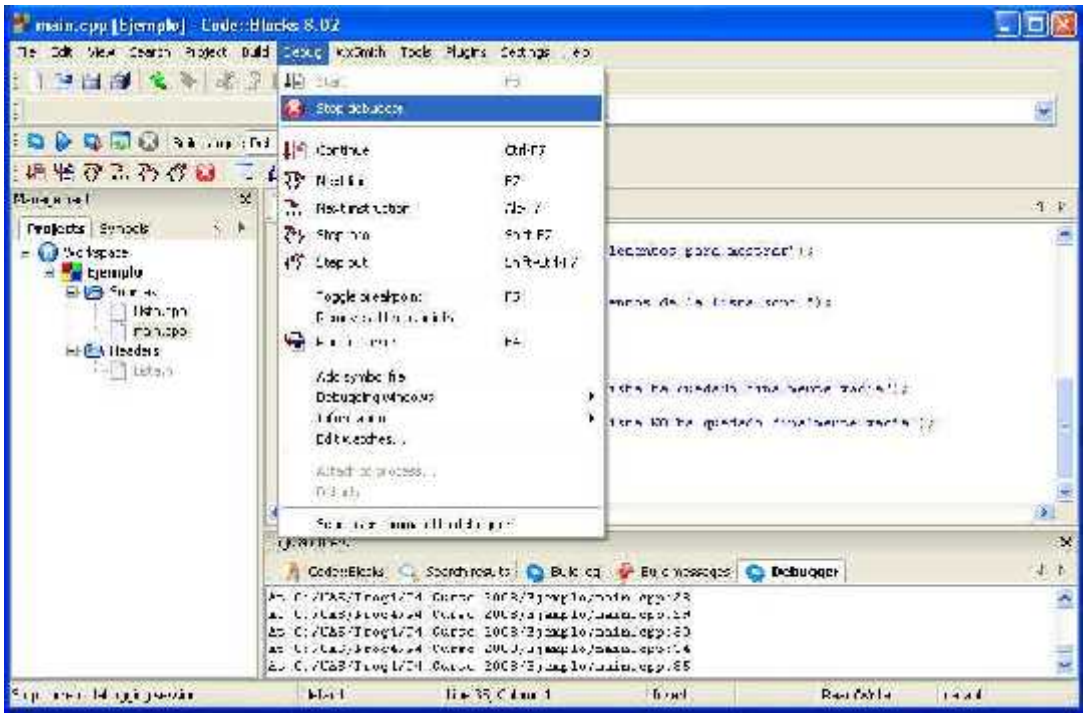
Sólo nos resta poner a correr el Debugger. Para ir ejecutando cada instrucción paso a paso y meternos dentro del código de cada subprograma invocado, elegimos *Debug Step Into*. Si queremos pasar directamente a la siguiente línea, elegimos *Debug Next Line*.



A medida que ejecutamos paso a paso, vemos cómo cambian los valores de las variables:



Para finalizar el debugger, elegimos *Debug* → *Stop Debugger*



Cuando terminamos de trabajar con el Project, lo cerramos con *File* → *Close Project*: