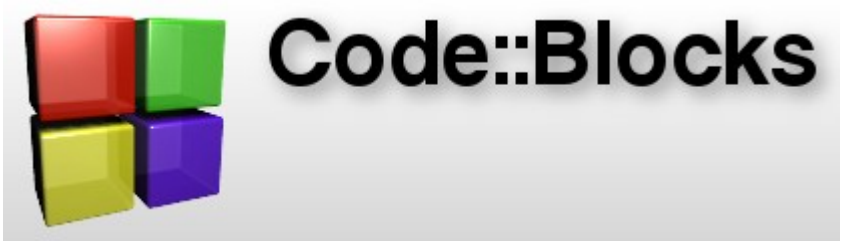
# Manual de uso de




*Programación en C/C++ estándar mediante CodeBlocks*

http://www.codeblocks.org

# Índice

1Introducción....3
2Crear un proyecto....3
2.1Insertar archivos ya existentes en el proyecto....6
2.2Insertar nuevos archivos en el proyecto....7
3Compilando y ejecutando....8
3.1Depuración....8
3.1.1Breakpoints....8

# 1 Introducción

**CodeBlocks** es un IDE (*Integrated Development Environment*, Entorno integrado de desarrrollo), que permite principalmente el desarrollo en *C* y *C++*, si bien otros lenguajes como *Python* también están soportados. En este documento se describe de manera sencilla las operaciones más comunes con el entorno, es decir: crear un proyecto, compilar, ejecutar y depurar.

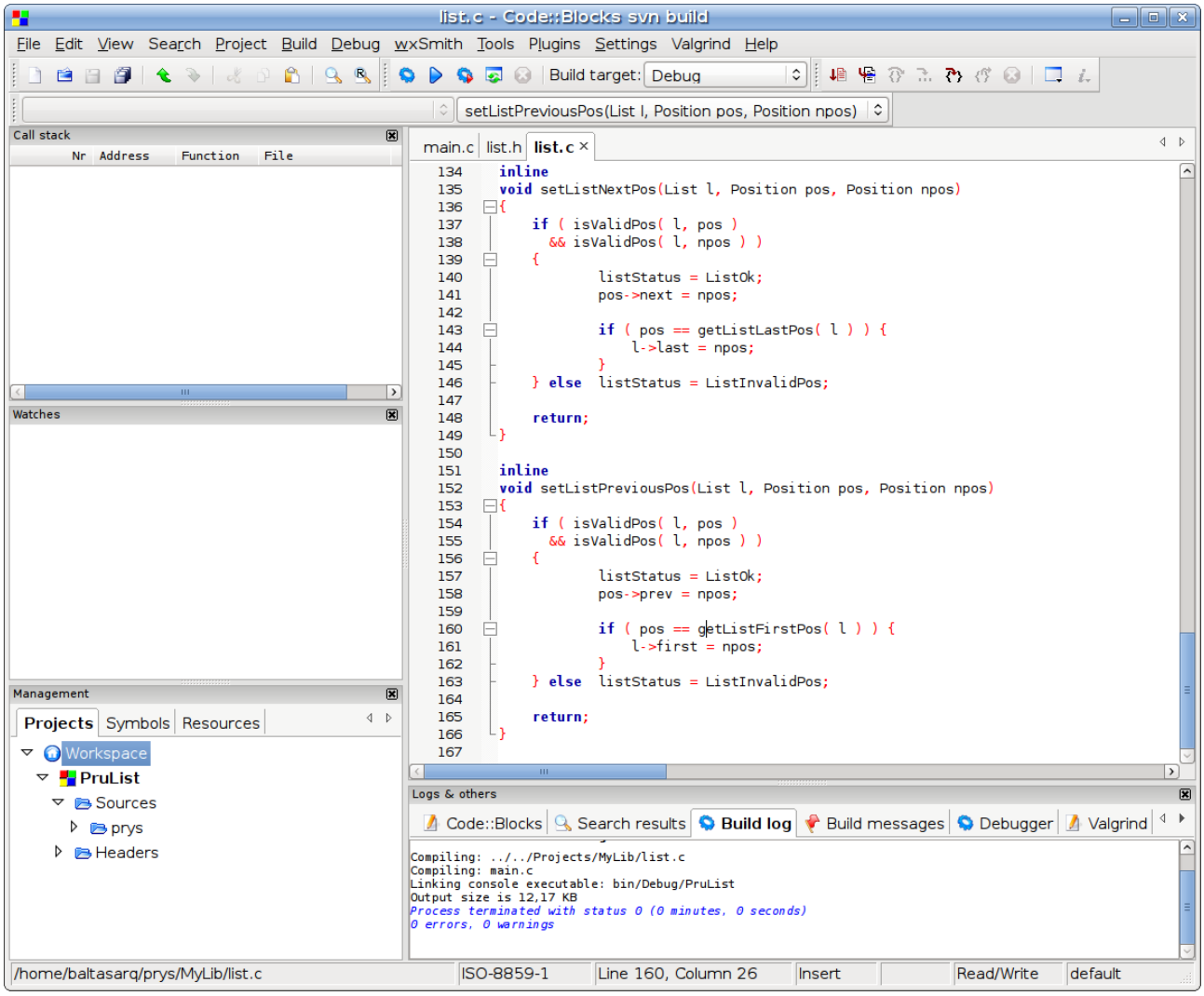



*Figura 1: El IDE Codeblocks*

En la figura 1, se aprecia la apariencia del entorno, con la parte derecha dedicada casi integramente a la edición de archivos (al ser un editor con pestañas, se pueden editar varios archivos al mismo tiempo). Debajo del edtor, se encuentra el área de mensajes, donde se obtienen varias informaciones como por ejemplo los resultados de una búsqueda. Principalmente, también se mostrarán los resultados de cualquier compilación.

En la parte izquierda se aprecian dos ventanas de depuración: *Call stack* (la pila de llamadas), y *Watches* (visores de variables), que sólo están activas mientras se está depurando un determinado proyecto. Esta es la configuración por defecto, por lo que la disposición de las ventanas pueden variar de un usuario a otro.

# 2 Crear un proyecto

Para crear un proyecto, se selecciona la opción *File >> New*, y a continuación, *Project*. Para construir el proyecto, básicamente se necesita darle un nombre, un directorio de residencia, y seleccionar el lenguaje y tipo de aplicación a desarrollar.

En primer lugar, se pregunta el tipo de aplicación a desarrollar, como se puede ver en la figura 2. El tipo de aplicación más adecuado para desarrollar aplicaciones en C o C++ estándar es *Console Application*.

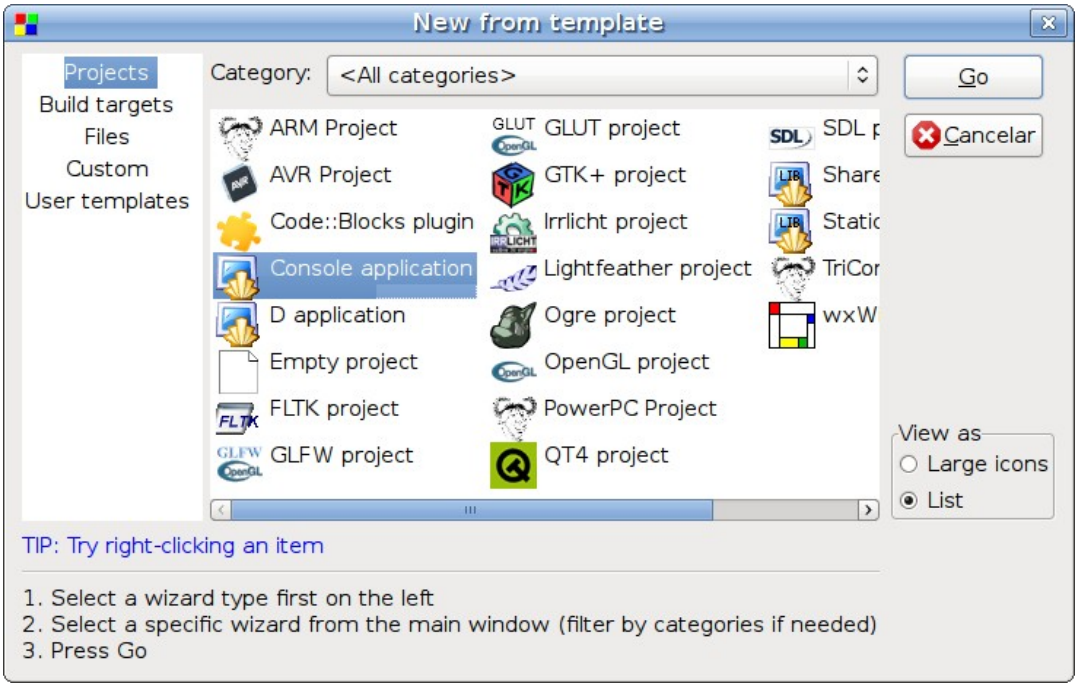



*Figura 2: Tipos de proyectos*

A continuación, se selecciona el lenguaje de programación a emplear para el proyecto. Por defecto, seleccionaremos entre *C* y *C++*, tal y como se ve en la figura 3. Una vez seleccionado el lenguaje, será necesario escoger un nombre para el proyecto, y una ubicación (directorio) en el disco duro.

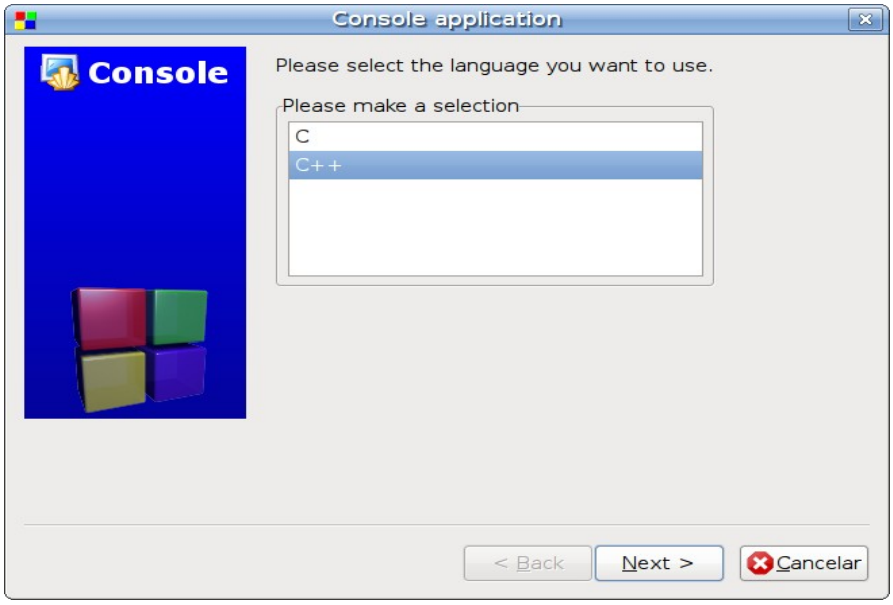



*Figura 3: Selección entre lenguajes de programación disponibles.*

El simple hecho de introducir un título hará que se complementen las entradas de directorio del proyecto y de nombre del proyecto, tal y como se ve en la figura 4. A continuación, es necesario seleccionar el compilador a emplear, además de los perfiles (modos de compilación) a utilizar, como se aprecia en la figura 5.

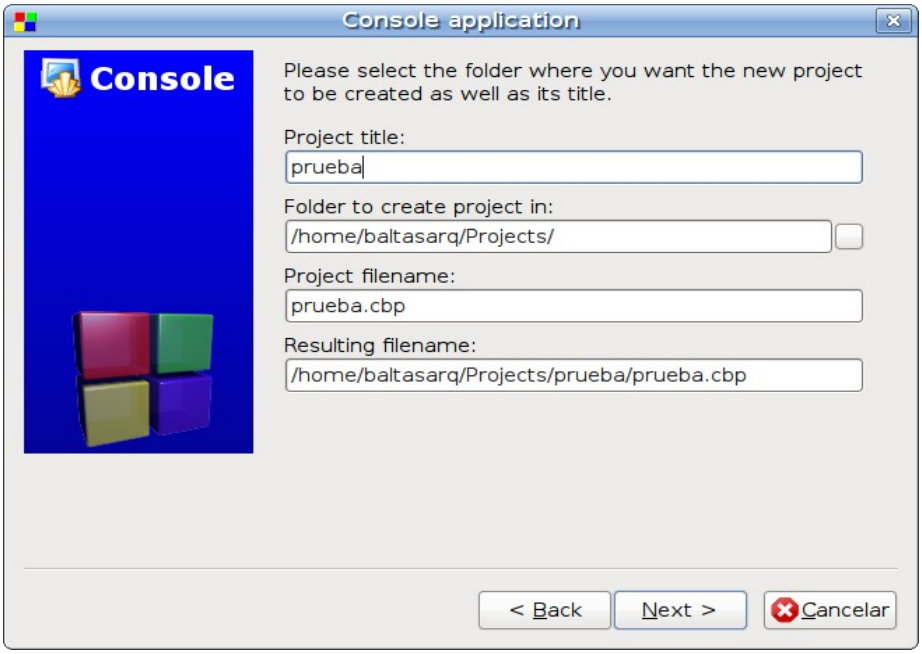



*Figura 4: Título del proyecto.*

Los perfiles por defecto son perfectamente adecuados para cualquier proyecto: *debug* servirá para el proceso de desarrollo del software, mientras que *release* selecciona las opciones del compilador de mayor optimización y elimina la información de depuración, para el momento del acabado..

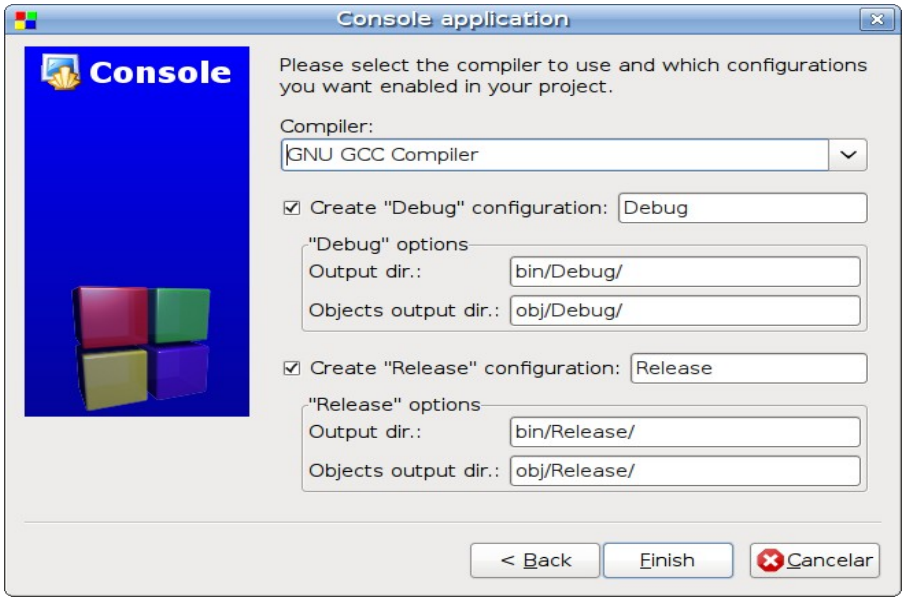



*Figura 5: Perfiles y selección del compilador.*

El compilador seleccionado es el GNU GCC, que es el que se provee en la configuración por defecto de CodeBlocks. Si para un proyecto específico se desea cambiar de compilador, se puede seleccionar ahora.

Una vez pulsado *Finish*, el proyecto ha sido creado, así como el archivo *main.c*, que aparecerá en el nuevo directorio y como archivo principal de la compilación. Para comprobarlo, sólo es necesario referirse a la ventana *Management*, y seleccionar la pestaña *Project*, como se observa en la figura 6. Bajo la sección con el nombre del proyecto que se haya seleccionado, encontraremos dos carpetas: *sources* y *headers*. Bajo la primera se encuentran todos los archivos .c y .cpp, mientras que bajo la segunda se encuentran todos los archivos .h. Haciendo doble clic en cualquiera de ellos, hace que aparezcan en el editor, listo para ser utilizados. Si se pulsa el botón derecho del ratón sobre ellos, se pueden eliminar del proyecto (aunque no son eliminados del disco).

## 2.1 Insertar archivos ya existentes en el proyecto.

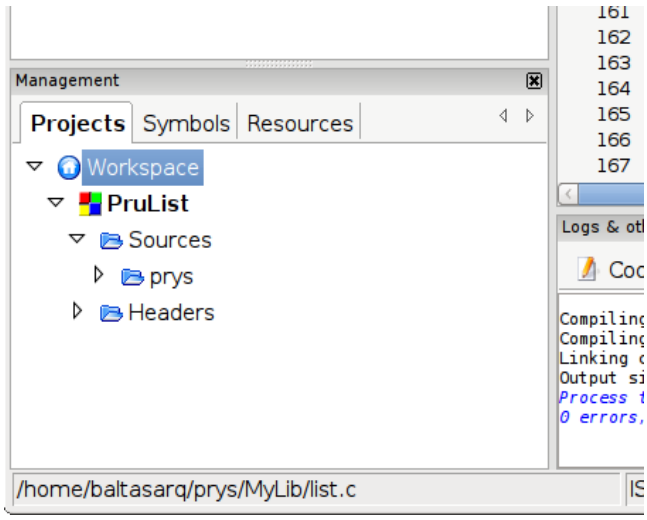



*Figura 6: Ventana* Management*, una vez creado el proyecto.*

Muchas veces, es necesario incluir archivos ya existentes en un proyecto que acabamos de crear, típicamente porque estos archivos son módulos que ya están hechos previamente y que deseamos aprovechar. Para añadir estos archivos, se utiliza la opción *Project >> Add Files*, que abre un cuadro de diálogo de apertura que permitirá seleccionarlos. En la ventana del proyecto aparecerán correctamente ordenados según sean archivos cabecera (.h) o archivos de implementación (.c, .cpp). Nótese que, para que una sentencia como `#include "cabecera.h"` sea correctamente compilada, también es necesario añadir al proyecto una indicación de dónde se encuentra ese archivo de cabecera (si es que no reside en el directorio del proyecto). Esto se logra en *Projects >> Build options*, seleccionando entonces la pestaña *Search Directories* (figura 7), y pulsando entonces el botón *add* para añadir un nuevo directorio. Entonces se escribe la nueva ruta (como se ve en la figura 8), y es añadido a la lista de búsqueda.

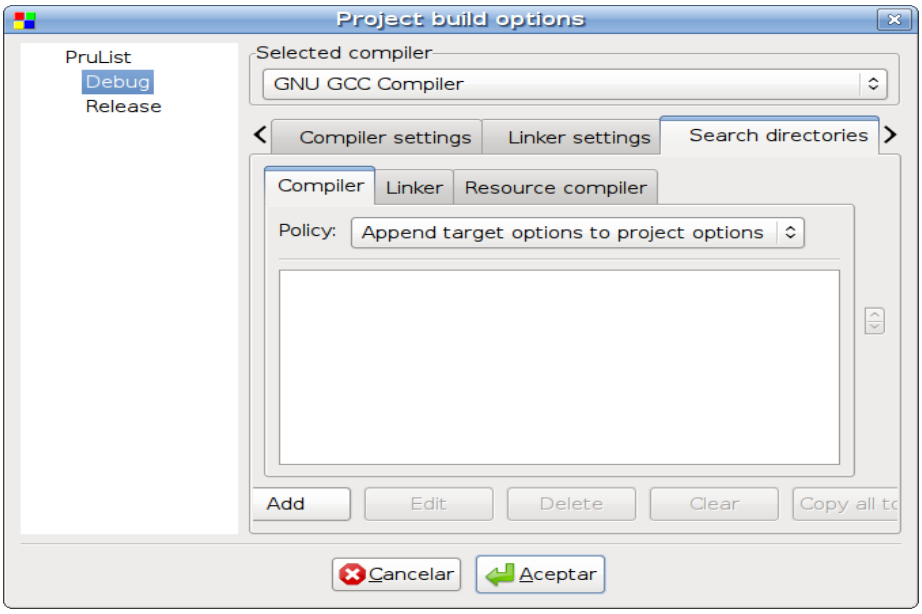



*Figura 7: Añadiendo directorios en las opciones de construcción del proyecto.*

Una vez hecho ésto, el proyecto se construirá perfectamente. Sólo es necesario tener en cuenta un detalle: si el camino del directorio a introducir contiene espacios, es posible que la compilación termine con varios errores aparentemente inexplicables (como por ejemplo, que no se encuentra c:\documents, en Windows), por lo que si es así, es mucho más conveniente indicar el nuevo directorio de búsqueda como un camino relativo, en lugar de absoluto. Así, si necesitamos un

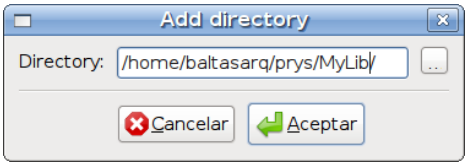



*Figura 8: Introduciendo el nuevo directorio de búsqueda.*

directorio llamado *Lib* que está al mismo nivel que el directorio del proyecto (como ejemplo, *HolaMundo/*), entonces será más conveniente fijar el nuevo directorio de búsqueda como *../Lib*.

También es posible, en esta ventana, cambiar el compilador que se ha estado usando, o las opciones de depuración u optimización, o cualquier otra que se desee añadir. Nótese que, a la izquierda, aparecen los perfiles que hayamos creado antes (por defecto, *debug* y *release*), de manera que los cambios afectan independientemente a una de los perfiles: para que afecten a ambos, será necesario repetirlos. Es sencillo comparar ambos perfiles: el primero tiene todas las opciones de depuración seleccionadas, mientras el otro no sólo no las tiene, sino que incluye todas las posibles optimizaciones.

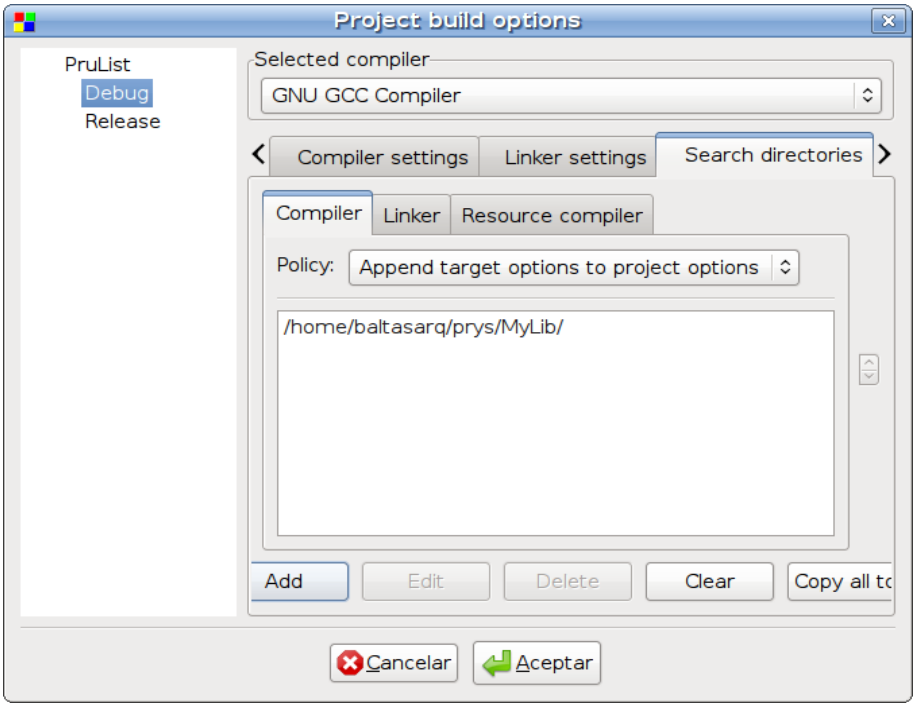



*Figura 9: Directorio de búsqueda añadido.*

## 2.2 Insertar nuevos archivos en el proyecto

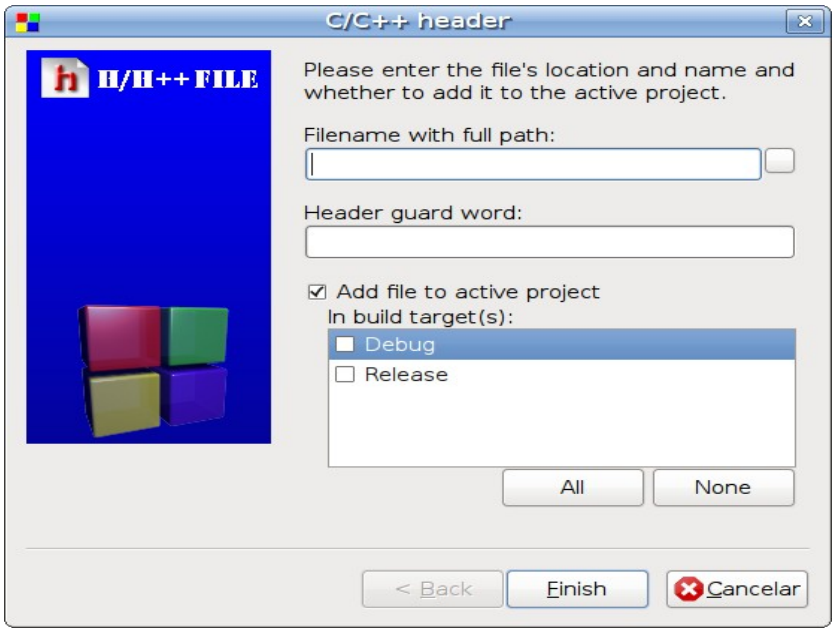



*Figura 10: Añadiendo una nueva cabecera al proyecto.*

Es típico que necesitemos incluir nuevos módulos en el proyecto, de manera que dividamos la

funcionalidad en partes independientes. Lo que desearíamos hacer en ese caso es muy parecido a lo hecho en el apartado anterior, sólo que ahora los archivos no existen previamente. Para acceder a esta opción, se selecciona *File >> New*, y entonces, *File*. Se nos da a elegir entre dos opciones: un nuevo archivo de cabecera (header, extensión .h), o de implementación (source, .c). Se escoja uno u otro, aparece una ventana muy parecida a la mostrada en la figura 10. En el apartado *file name* se indica el nombre del archivo, aunque es necesario tener en cuenta que hay que especificarlo con la ruta completa, por lo que será mucho más cómodo utilizar el botón a la derecha de este campo, que abre un cuadro de diálogo de guardado. Si se desea añadir el archivo al proyecto (que es lo más habitual), se deja marcada la opción *Add file to active project*. En este caso es necesario añadir el archivo a los perfiles *debug* y *release*. Lo más habitual es asignárselo a ambos, pudiendo hacerlo a través del botón *all* o marcando cada uno de ellos. En cuanto al campo *header guard word*, se trata de la típica constante utilizada por el preprocesador para hacerle saber si ya ha compilado esta cabecera previamente para esta unidad de traducción. **CodeBlocks** rellena este campo automáticamente, una vez introducido el nombre del archivo, por lo que no es necesario preocuparse por ello.

Finalmente, hay que tener en cuenta que si se ha creado un archivo de cabecera, y éste se encuentra en un directorio que no es el directorio del proyecto, será necesario, como se indica en el apartado anterior, añadir la ruta del directorio en el que se encuentra a las opciones del proyecto.

# 3 Compilando y ejecutando

Las opciones *Build >> build* y *Build >> run*, construyen y ejecutan el programa, respectivamente. Se pueden atajar mediante las pulsaciones de teclas Ctrl+F9 y Ctrl+F10, o, incluso, si se desean realizar ambas acciones en secuencia, pulsando F9. En cuanto a la barra de herramientas, pulsando la ruenda dentada se compila, y pulsando el triángulo de reproducción se ejecuta.

Cuando el proyecto consiste en una aplicación de consola, una ventana de consola se abre para recibir y mostrar texto, y cuando termina la ejecución espera a que se pulse una tecla.

## 3.1 Depuración

La depuración no se produce cuando el programa se ejecuta normalmente, mediante Ctrl+F10 ó F9. Para depurar un programa es necesario seleccionar la opción *Debug >> start*, o bien pulsar F8, que lanza la depuración del programa desde el principio. Si se desea empezar a depurar desde un punto determinado, o durante la depuración, hacer que el programa corra normalemente hasta una determinada línea, se sitúa el cursor sobre esa línea y se selecciona *Debug >> run to cursor*, o se pulsa F4.

A la hora de depurar una aplicación, se utilizan principalmente las opciones *Debug >> Step into* (Shift + F7) y *Debug >> next line* (F7). Mientras que la segunda va avanzando de línea en línea, y ejecutando a las funciones que sean llamadas en cada una de manera transparente, la primera permite saltar de función en función, conjuntamente con el flujo del programa.

### 3.1.1 Breakpoints

Los puntos de ruptura (*breakpoints*), se utilizan para parar la ejecución cuando se está depurando, y se pulsa F8 para que la ejecución continúe normalmente. Se marcan con un círculo rojo a la izquierda de la línea, pulsando con el ratón entre la línea y el número de línea, o bien situándose en esa línea y pulsando F5 (figura 11). Cuando la ejecución llegue a esa línea, símplemente se detendrá, y nos permitirá utilizar la ventana de observadores (*watches*), de manera que se pueda consultar el valor de cualquier variable (*Debug >> edit watches*).

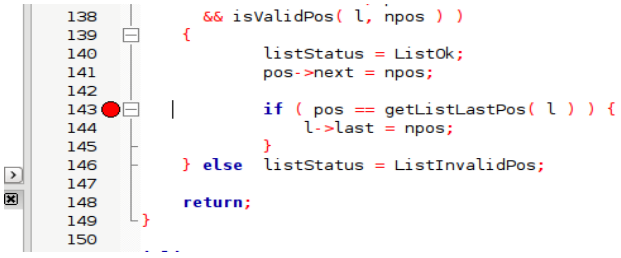



*Figura 11: Un* breakpoint *en la línea 143.*