

Dev-C++ Instalación y primeros pasos.

Introducción.

Este documento está destinado a aquellos que acaban de empezar con la programación en el lenguaje C o C++. Describiré la instalación, configuración y uso de un Entorno Integrado de Desarrollo (en adelante IDE, Integrated Development Environment) para las plataformas Windows: el Dev-C++. Veremos todas las opciones que permite este programa, así como algunas de sus herramientas más útiles. Un IDE es simplemente una agrupación de herramientas destinadas al desarrollo, de forma que con un solo programa podamos acceder a todo lo que necesitamos para crear nuestras aplicaciones. En este caso dispondremos de un editor de texto, un depurador y un compilador de C y C++, entre otras muchas herramientas más que iremos descubriendo. El porqué de esta elección, es simple: el programa es software libre, lo que implica que podemos acceder al mismo y/o a su código fuente de forma gratuita. Además, es un entorno muy potente, y nos permitirá hacer casi todo tipo de aplicaciones en el entorno Windows.

Cabe destacar que el Dev-C++ en sí no es el compilador, sino simplemente un entorno gráfico para utilizar el verdadero compilador: el MinGW (Minimalist Gnu Windows). Este compilador, como puede deducirse por el nombre, es también software libre bajo la licencia GNU. El compilador es un port (conversión) del famoso compilador GCC de los sistemas GNU/Linux y, de hecho, puede ser utilizado como el original, por línea de comandos y con sus mismos argumentos. En este texto no se va a hablar de forma profunda acerca de cómo usar el compilador, ni de todas las opciones de compilación y optimización que éste posee (que son muchas ☺).

Para eso está la página oficial del MinGW, así como la página del compilador GCC:
<http://www.mingw.org/>
<http://www.fsf.org/software/gcc/gcc.html>

Instalación del Dev-C++.

Ya hemos comentado que el propio Dev-C++ no es un compilador en sí, sino un entorno de programación que utiliza un compilador. Podríamos bajarnos el IDE por un lado, y el compilador por el otro, instalarlo y utilizarlo. Sin embargo teniendo en cuenta que en la página tenemos un paquete que lo reúne todo, será cuestión de aprovechar la circunstancia y no complicarnos la vida.

Vamos a la página oficial del Dev-C++: <http://www.bloodshed.net/dev/devcpp.html>

En la sección Downloads tenemos la siguiente versión: **Dev-C++ 5.0 beta 8 (4.9.8.0) (12 MB) with Mingw/GCC 3.2** y varios lugares para descargarlo. Pinchamos en cualquiera de ellos y a esperar; son unos 13 megas aproximadamente (no confundir con la versión de 2MB, que incluye únicamente del IDE, no el compilador)

NOTA: la última actualización a fecha de 1 de Diciembre de 2003 es la 4.9.8.5, la cual está disponible para la descarga la misma página web. (Se descarga únicamente el ejecutable y se descomprime en el directorio de instalación, por lo que seguimos necesitando descargar la versión 4.9.8.0) Sin embargo, más adelante explico el funcionamiento del programa de actualización por web que sirve para actualizar automáticamente -valga la redundancia- nuestro programa, así que no es estrictamente necesario que descarguéis la actualización ahora.

No os dejéis engañar porque sea una beta, es bastante estable. Aunque tiene sus pequeños bugs, éstos no son muy comunes.

Una vez descargado tendremos un ejecutable auto-instalable. Doble clic en él y se podrá a punto. Lo único que tenemos que seleccionar es el directorio de destino así como si queremos iconos en el escritorio, menú inicio o quick-launch. Es recomendable dejar el directorio por defecto (c:\dev-cpp) o al menos usar una ruta de directorios que

Dev-C++ Instalación y primeros pasos.

no tenga espacios en los nombres de directorios. Parece ser que el compilador tiene algún tipo de problema con ello, aunque a mí nunca me ha pasado. Una vez terminado ya tenemos el IDE listo para funcionar.

Conociendo el entorno.

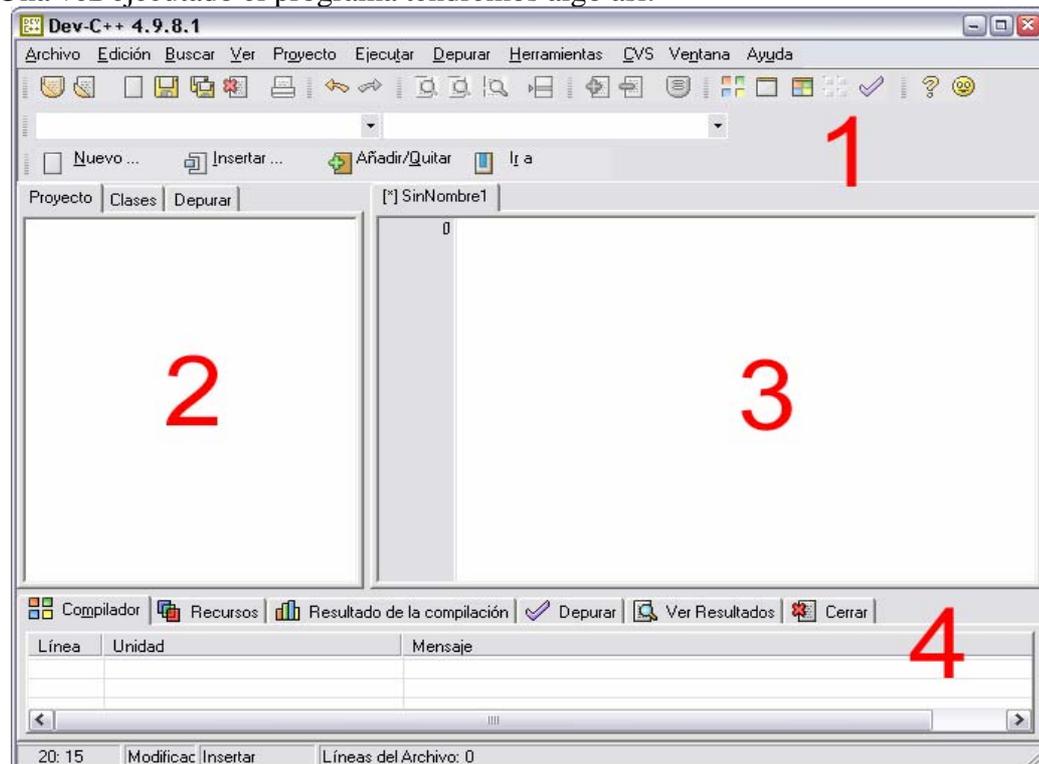
Antes que nada me gustaría comenzar con la definición de algo con lo que vamos a tratar mucho a lo largo del documento: el concepto de proyecto.

Un proyecto no es más que una agrupación de todos los ficheros que requiere una aplicación, así como de su configuración específica. Cuando en el Dev-C++ creamos un proyecto se crea un fichero .dev. Es un fichero de texto que contiene referencias a los ficheros necesarios para compilar nuestra aplicación (ficheros de código, ficheros de encabezado...) así como las opciones necesarias específicas de esa aplicación (como pueden ser referencias a bibliotecas no estándar que se necesitan para compilar) También cumple una función similar a la de la utilidad **make** de linux: comprueba las dependencias entre los ficheros que componen el proyecto para saber cuáles han sido modificados (esos archivos serían los únicos que necesitaría recompilar) Tanto es así que el programa crea un fichero llamado makefile.win ☺

Resumiendo y en otras palabras: cuando queramos crear una aplicación distribuida en varios ficheros o que dependa de más bibliotecas/recursos es recomendable utilizar un proyecto.

Bien, lo habíamos dejado después de la instalación ¿verdad?, pues continuaremos ejecutando el programa. En la primera ejecución nos pedirá que seleccionemos el idioma, así como su estilo de iconos. El Dev-C++ se distribuye “de serie” con el idioma castellano, así que sólo es cuestión de buscarlo en la lista que aparece. La apariencia de los iconos es cuestión del gusto de cada uno. Cabe destacar que hay dos opciones de idioma castellano. Yo uso spanishcastellano.lng; la otra opción es (spanish)Latinoamérica. Puede que ambas traducciones difieran un poco una de otra.

Una vez ejecutado el programa tendremos algo así:



Dev-C++ Instalación y primeros pasos.

Vamos entonces a describir cada una de estas cuatro áreas:

1. Menú y barras de herramientas

Aquí tenemos los menús con los típicos comandos de Windows (abrir, guardar, copiar y pegar...) También tenemos una serie de iconos en las barras de herramientas que no son más que una parte de las opciones que tenemos en los menús, así que por tanto, no creo que haga falta explicarlos uno por uno. Si aún así no te aclaras, puedes dejar el ratón encima de un icono durante unos segundos y aparecerá una ayuda emergente. En ella se incluye una descripción básica y el atajo de teclado asociado a la función del icono, si este existe. (Como veremos, los atajos de teclado también se pueden configurar)

2. Explorador de proyectos y clases e información de depuración.

Dependiendo de la pestaña que seleccionemos en esta área tendremos acceso a:

- a) *Explorador de proyectos*, que muestra los archivos por los que está formado nuestro proyecto -y por tanto nuestra aplicación- bien sean de código, de encabezados, o de recursos.
- b) *Explorador de clases*, una de las funciones más útiles y que ya conoceréis si habéis visto el Visual Studio. En este caso veremos cada una de las estructuras/clases definidas en los ficheros de nuestro proyecto, así como los métodos y datos que forman parte de la estructura/clase, incluyendo sus argumentos y su tipo. También veremos una lista de las funciones globales que tenemos en el proyecto, también con sus argumentos. Pulsando doble clic en un método, función o clase, iremos directamente al archivo y línea donde se ha definido.
- c) *Información de depuración*, aquí podremos definir las variables que queramos cuando estemos depurando un programa.

3. Área de edición.

Aquí aparecerán los ficheros de código que abras. Puedes tener abierto más de un fichero a la vez, y seleccionarlo por medio de las pestañas que aparecerán en la parte superior de este área.

4. Resultados de la compilación y controles de depuración.

En ésta serie de pestañas encontraremos información acerca del proceso de compilación y depuración. Cuando seleccionemos una pestaña se expandirá para mostrarnos los resultados, al tiempo que aparecerá una nueva pestaña que se sumará a las cinco anteriores: la pestaña *Cerrar*. Pulsándola volveremos a tener el mismo espacio que teníamos antes.

En la pestaña *compilador* veremos los errores y advertencias que ha generado la compilación de nuestro código (si los hubiera). Pulsando doble clic en uno de ellos nos remitirá directamente a la línea que provocó dicho error o advertencia. En la pestaña *resultados del compilador*, tendremos toda la salida que genera el compilador gcc (que recordemos se maneja por línea de comandos) Esto también incluye errores y avisos, al igual que en la pestaña anterior. Sin embargo no tenemos la opción del doble clic para ir directamente a la línea que provocó el error.

Dev-C++ Instalación y primeros pasos.

En la pestaña *depurar* tendremos los controles de depuración que son los mismos que los que hay en el menú Depurar (salvo dos excepciones que existen en el menú, pero no aquí. Eso lo veremos con más detalle posteriormente)

Pasamos ahora a comentar de manera un poco más detallada los menús y sus opciones. Supongo que sois personas familiarizadas con el entorno Windows, así que no me pararé en explicar las opciones comunes a las aplicaciones Windows, ni aquellas funciones que sean totalmente obvias. ☺

Menú archivo:

- *Nuevo*: Aquí podemos crear un nuevo fichero de código fuente en blanco (que no es otra cosa que un fichero de texto); un nuevo proyecto seleccionado entre varios tipos un fichero de definición de recursos (los recursos se utilizan en la programación con la API win32); o crear una plantilla a partir del código que tenemos que se convertirá en otro tipo de proyecto. Una plantilla en este caso, es un nuevo tipo de proyecto, que nos sirve de base para empezar a programar. Por ejemplo puedes definir una plantilla “hola mundo” que ya contenga los ficheros y el código necesario para imprimir en pantalla el famoso mensaje. De la misma manera los programas en win32 tienen unas estructuras básicas (función WinMain, definición e inclusión de encabezados...) que poseen todos los programas. Puedes generar una plantilla, y así no tienes que teclear lo básico cada vez.

- *Propiedades*: Muestra las características de los ficheros del proyecto (o del fichero actual si no hay proyecto) como líneas totales, líneas de código, líneas comentadas, tamaño...

- *Importar*: Solo tenemos la opción de importar un proyecto de Visual C++. Lo que hace esta opción es transformar el proyecto de visual C++ en un proyecto de Dev-C++, con sus dependencias de bibliotecas y ficheros de código, pero **no** transforma el código para que sea posible su compilación directamente con Dev-C++. Sólo lo he probado con proyectos simples y funciona bastante bien. Cabe decir que es inútil transformar un proyecto de Visual C++ que utilice MFC, ya que Dev-C++ no tiene soporte para ellas. Su mayor utilidad reside en transformar un proyecto que utilice las API Win32, que sí soporta el compilador.

- *Exportar*: Podemos exportar el fichero de código actual a un fichero HTML o RTF, o exportar el proyecto a HTML. Puede ser útil para la distribución o para colgar el código en una página web de forma ordenada.

Menú Edición:

- *Insertar*: Tenemos las opciones de insertar la fecha y hora actual o una plantilla de una cabecera básica para la descripción del fichero fuente. Sin embargo podemos agregar nuestros propios elementos a este menú. Más adelante veremos cómo hacerlo. El icono insertar que aparece en una de las barras de herramientas cumple la misma función que éste menú.

- *Intercambiar header/source*: Conmuta entre los archivos de código fuente y su archivo de encabezado asociado. Para que esto funcione, ambos archivos deben tener el mismo nombre, y cambiar solamente en su extensión (Por ejemplo: main.cpp y main.h) Esta opción también está disponible haciendo clic derecho en el área de edición y seleccionando la misma opción (siempre que tengamos un fichero abierto, claro)

- *Añadir o quitar marcador / ir a marcador*. Sirve para crear una marca en una línea concreta de un fichero de código (no tienen porqué estar todas en el

Dev-C++ Instalación y primeros pasos.

mismo fichero) para luego desplazarnos a ella de una forma más rápida. También tenemos un icono en la barra de herramientas con esta función.

- *Agregar comentario*: Agrega un comentario de C++ (//) si hacemos una selección y elegimos esta opción, comentará todas las líneas de la selección.
- *Quitar comentario*: Realiza la función contraria a la anterior.
- *Poner/quitar margen*: Hace una sangría a la línea/selección.

Menú Búsqueda:

Tiene las opciones típicas de búsqueda y reemplazo. Cabe destacar una función de búsqueda en varios ficheros, así como los comandos *Ir a línea* y el siempre útil comando *Ir a función*, que busca entre las funciones del fichero actual.

Menú Ver:

Nada del otro mundo. Podemos seleccionar qué barras de herramientas queremos que muestre el Dev-C++, así como si queremos que muestre el explorador de proyectos/clases. También podemos elegir si queremos que el explorador de proyectos y clases, así como los resultados del compilador; se muestren anclados al programa principal (por defecto) o como una ventana aparte. Por último podemos cambiar las opciones de los resultados del compilador para que esté siempre visible, o sólo cuando sea necesario (es decir, cuando compilemos nuestro programa y surjan errores o advertencias) Yo os recomiendo la segunda opción.

Menú Proyecto:

- *Nuevo código fuente*: crea un nuevo fichero de código que se añade automáticamente al proyecto. Cuando lo salves por primera vez te pedirá el nombre del fichero.
- *Añadir al proyecto*: permite añadir ficheros al proyecto.
- *Quitar del proyecto*: muestra un cuadro de diálogo con los ficheros pertenecientes a nuestro proyecto para poder eliminar los que queramos (se eliminan del proyecto, no se borran de su ubicación)
- *Propiedades del proyecto*: para cada proyecto podemos definir unas opciones diferentes de las comunes a las definidas en el compilador. Por ejemplo podemos vincular con más bibliotecas, añadir 'flags' de compilación o activar la depuración, de forma que todas estas configuraciones sólo se aplicarán a nuestro proyecto.

Dev-C++ Instalación y primeros pasos.

Menú Ejecutar:

- *Compilar*: compila y vincula el fichero o el proyecto entero para crear el ejecutable/librería. Como ya dije los archivos que no fueron modificados no son recompilados.
- *Compilar el archivo actual*: compila el fichero que estás editando actualmente. No se vincula.
- *Ejecutar*: Ejecuta el programa (si existe ejecutable) ¡Ojo! Este comando no compila, por lo que si has modificado tu programa, pero no lo recompilas, y le das a este botón (suponiendo que ya tuvieras un ejecutable en el directorio de trabajo) ejecutará la versión anterior. Puede parecer una tontería, pero considerando que me ha pasado unas cuantas veces (la mayoría de las cuales han acabado en intentos de suicidio) creo que es mejor recalcarlo.
- *Parámetros*: Si queremos que nuestro ejecutable reciba algún tipo de parámetro se pone aquí. Sólo los parámetros, no hace falta repetir de nuevo el nombre del ejecutable.
- *Compilar y ejecutar*: Compila, vincula y automáticamente ejecuta el programa (si no hubo ningún error, claro)
- *Reconstruir todo*: Borra todos los archivos de código objeto, para recompilar todos los ficheros de código fuente. También vincula todos. En definitiva, reconstruye el proyecto de cero. Se consigue el mismo efecto que pulsando “Limpiar resultados” y luego “Compilar” en este mismo menú.
- *Revisar sintaxis*: Comprueba si la sintaxis del código es correcta. No genera código objeto alguno. Es lo mismo que invocar al compilador con la opción **-fsyntax-only**
- *Limpiar resultados*: Borra todos los archivos de código objeto y el ejecutable, si existen.
- *Análisis de perfil*. Muestra la información del profiler, que consiste en datos estadísticos acerca de las funciones del programa, como el número de veces que son llamadas, el tiempo que tardan en ejecutarse, el porcentaje de tiempo total del programa que se ejecutan... etc. Sirve para encontrar las funciones más utilizadas o más lentas, con el fin de optimizar éstas. Aunque el IDE contempla estas opciones, no funcionan correctamente en esta versión, aunque cuando actualicemos el programa a la versión 4.9.8.5 funcionarán correctamente.
NOTA: Añadiendo la opción **-pg** tanto en las opciones del compilador (ya sea el de C o C++) como en las opciones del linker (más adelante veremos dónde podemos añadir éstas opciones) obtienes el mismo efecto.
Deberás ejecutar el programa una vez para que la información de perfil se genere, y por último seleccionar ésta opción. Incluir información de perfil hace que el programa se ejecute más lentamente, así que no debe activarse en programas finales.
- *Reiniciar ejecución del programa*: Esta opción sólo está disponible si estás ejecutando tu programa. Selecciónalo para reiniciar el programa.

Menú depurar:

Ya que veremos este asunto más adelante de forma detallada, no me voy a extender mucho. Espero que las explicaciones que no se entiendan ahora sean respondidas entonces.

- *Depurar*: Creo que está claro, ¿no? Inicia la depuración del proyecto. ☺
- *Parar ejecución*. Finaliza la ejecución del programa, así como el modo depuración.
- *Parámetros*. Igual que la opción del menú compilar, pero sólo se aplica cuando estemos depurando el proyecto.

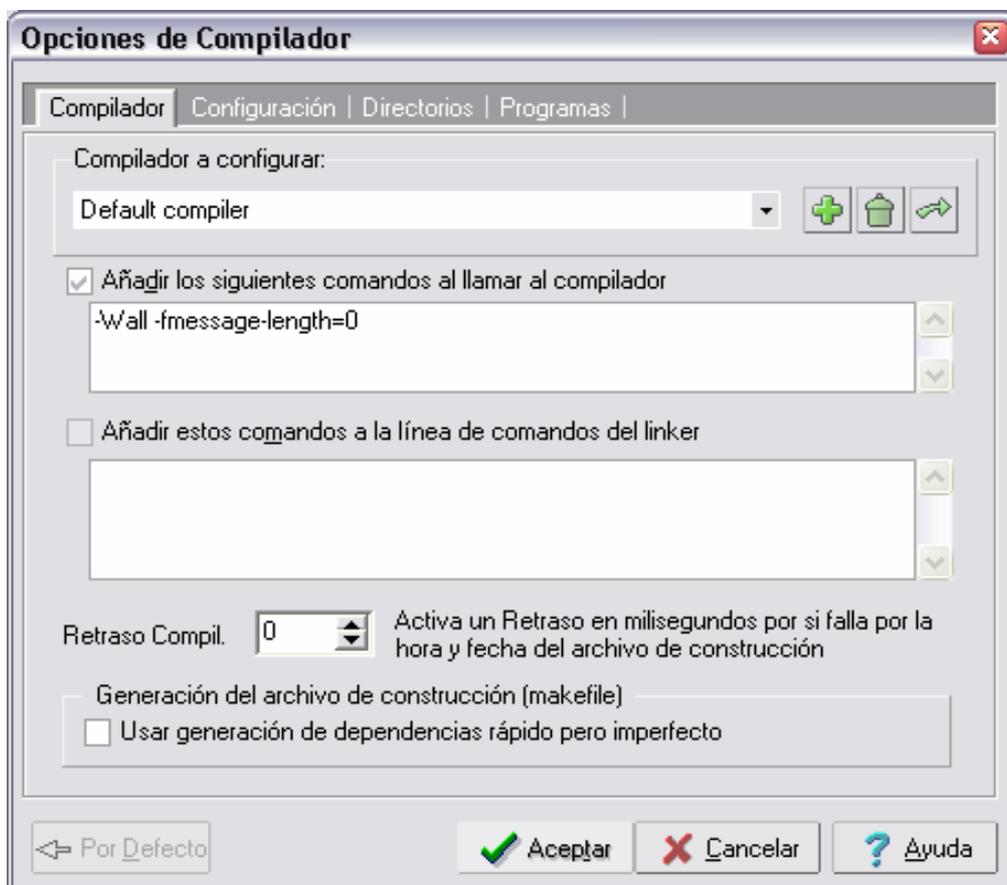
Dev-C++ Instalación y primeros pasos.

- *Añadir quitar punto de ruptura:* Añade un *breakpoint*, es decir, un punto en el cual la ejecución del programa se detendrá.
- *Siguiente paso:* Siguiente instrucción del programa
- *Avanzar paso a paso:* Igual que el anterior, pero ejecuta los pasos dentro de un procedimiento.
- *Saltar paso:* Esta opción debería ser traducida como continuar. Continúa la ejecución del programa hasta que se encuentre otro *breakpoint* o finalice.
- *Ir a cursor:* Se ejecuta el programa hasta que llegue al punto donde está el cursor (si no hay un *breakpoint* antes)
- *Añadir watch:* Abre un cuadro de diálogo para introducir la variable que queremos observar en tiempo de ejecución. La variable y su contenido aparecerá en el área 2 (siempre que tengamos seleccionada esa pestaña).
- *Observar variables:* Selecciona la pestaña 'Depurar' del área 2.
- *Ver ventana del CPU:* Debug avanzado ☺ Podemos observar los registros de la máquina en un momento de la ejecución. No lo uso, así que no puedo comentar mucho más.

Menú Herramientas

Aquí tenemos las opciones de configuración del IDE y del compilador. También tenemos algunas herramientas ya incluidas con el programa, aunque también podemos agregar objetos a este menú para definir nuestras propias herramientas, y tenerlas a mano.

- Opciones del compilador: Compilador



Dev-C++ Instalación y primeros pasos.

- *Compilador a configurar:*

Podemos definir diferentes opciones de compilación, como si tuviéramos varios compiladores iguales pero con configuraciones diferentes.

- *Añadir los siguientes comandos al llamar al compilador.*

Como se he repetido hasta la saciedad, minGW es un compilador basado en GCC, que es de línea de comandos, y por tanto se pueden introducir los mismos argumentos. En este lugar puedes introducir los argumentos que consideres necesarios para el compilador. Ten en cuenta que estos argumentos se aplicarán siempre y a todos los proyectos. Concretando, no es buena idea poner los argumentos para la optimización aquí, pues se aplicarían siempre, y tendrías que estar modificando esto, si quieres compilar un proyecto con información de depuración. (Además, los argumentos más comunes de depuración y optimización, vienen implementados con menús, así que no es necesario que lo escribas aquí)

Si estás interesado en lo que tengo puesto es

-**wall** muestra todas las advertencias del compilador.

-**fmessage-length=0** sirve para que en el recuadro de *resultados del compilador* el texto no aparezca cortado (sacado de [Aditsu's Unofficial Dev-cpp FAQ](#)) El número de la opción es el número de caracteres antes de que se produzca un salto de línea. Si n es 0 el mensaje aparece en una única línea, así podemos verlo sin cortar en el IDE.

- *Añadir estos comandos a la línea de comandos del linker.*

Igual que el anterior, pero para introducir argumentos para el linker (al vincular el código objeto y generar el ejecutable)

- *Retraso de compilación:*

Agrega un retraso en milisegundos al compilar. Según he leído sirve para corregir algunos errores.

- *Utilizar generación de dependencias rápido pero imperfecto.*

Con *dependencia* el compilador se refiere al control que tiene para saber qué archivo se ha modificado, y cuantos dependen de él (porque en ese caso habría que recompilar el archivo modificado y, consecuentemente, todos aquellos que dependen de él) Si chequeas esto, no se comprobará si los ficheros de cabecera han sido modificados, con lo que los cambios que hagas a éstos sólo serán perceptibles cuando compiles el fichero/os de código asociado al mismo. Es recomendable quitarlo. Te ahorrarás quebraderos de cabeza, y el aumento del tiempo que tarda en compilar es muy pequeño.

• Opciones del compilador: Configuración

NOTA: Las capturas están recortadas para que ocupen menos espacio.



Dev-C++ Instalación y primeros pasos.

- *Compilador de C:*

Soporte para Programas ANSI Estándar en C:

Equivale a la opción de compilación `-ansi`. El compilador intenta cumplir lo más estrictamente posible el estándar ANSI. Más información (inglés) en <http://gcc.gnu.org/onlinedocs/gcc-3.2.3/gcc/C-Dialect-Options.html#C%20Dialect%20Options>

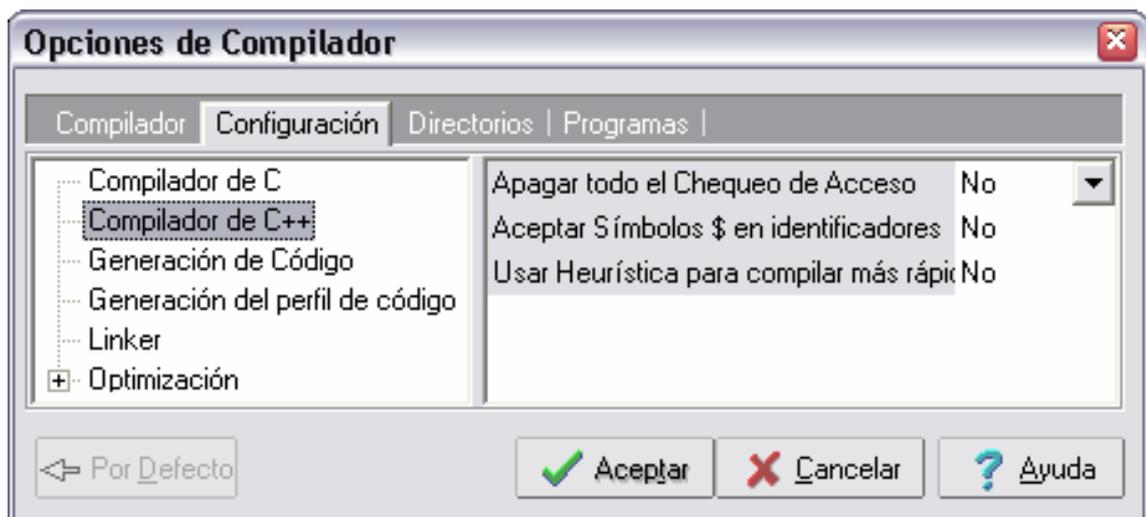
Trata de Soportar Preprocesadores Tradicionales de C:

Equivale a la opción `-traditional-cpp`. Más información (inglés): <http://gcc.gnu.org/onlinedocs/gcc-3.2.3/cpp/Traditional-Mode.html#Traditional%20Mode>

No mostrar mensajes de alerta.

Equivale a la opción `-w` (ojo, minúscula) El compilador no muestra ninguna advertencia. No lo recomiendo, ya que éstas son de gran utilidad, y de todas maneras si el programa sólo genera advertencias compilará igual. Más información (inglés) en:

<http://gcc.gnu.org/onlinedocs/gcc-3.2.3/gcc/Warning-Options.html#Warning%20Options>



- *Compilador de C++*

Apagar todo el Chequeo de Acceso:

Equivale a la opción de compilación `-fno-access-control` Desactiva todas las comprobaciones de acceso.

Aceptar Símbolos \$ en identificadores:

Equivale a la opción de compilación `-fdollars-in-identifiers` El C tradicional permitía usar el carácter \$ en identificadores, aunque el ISO C e ISO C++ lo prohíben. Activando esta opción podremos utilizar identificadores que contengan el símbolo \$.

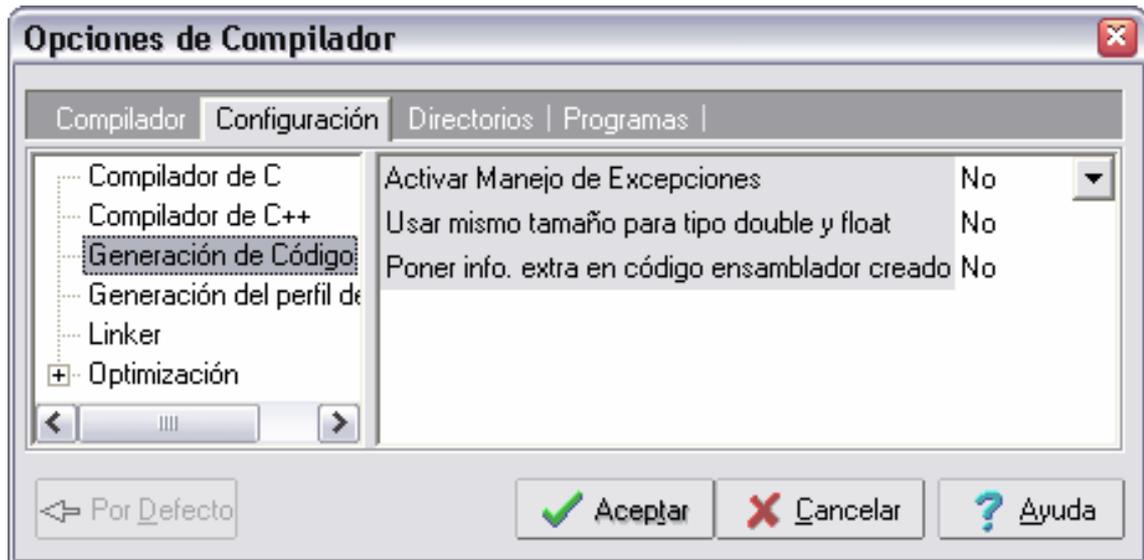
Usar Heurística para compilar más rápido:

Ni idea de qué hace ni a qué opción equivale. Yo lo tengo siempre desactivado.

Más información sobre opciones del compilador de C++ en esta página (en inglés)

<http://gcc.gnu.org/onlinedocs/gcc-3.2.3/gcc/C---Dialect-Options.html#C++%20Dialect%20Options>

Dev-C++ Instalación y primeros pasos.



- *Generación de código.*

Activar Manejo de Excepciones:

Equivale a la opción de compilación `-fexceptions`. Activa el uso de las excepciones. Si no se especifica nada, se supone que el compilador lo activa para C++ y desactiva para C, aunque no se que opciones concretas especifica el IDE al activar o desactivar esta opción. Al activarlo se fuerza incluso para programas en C. Para desactivarlo en programas C++ se usa `-fno-exceptions`.

Usar mismo tamaño para tipo double y float:

Equivale a la opción de compilación `-fshort-double`. Ojo porque el código compilado con esta opción activada puede no ser compatible con el código compilado con esta opción desactivada.

Más información sobre opciones de generación de código en esta página (en inglés)

<http://gcc.gnu.org/onlinedocs/gcc-3.2.3/gcc/Code-Gen-Options.html#Code%20Gen%20Options>

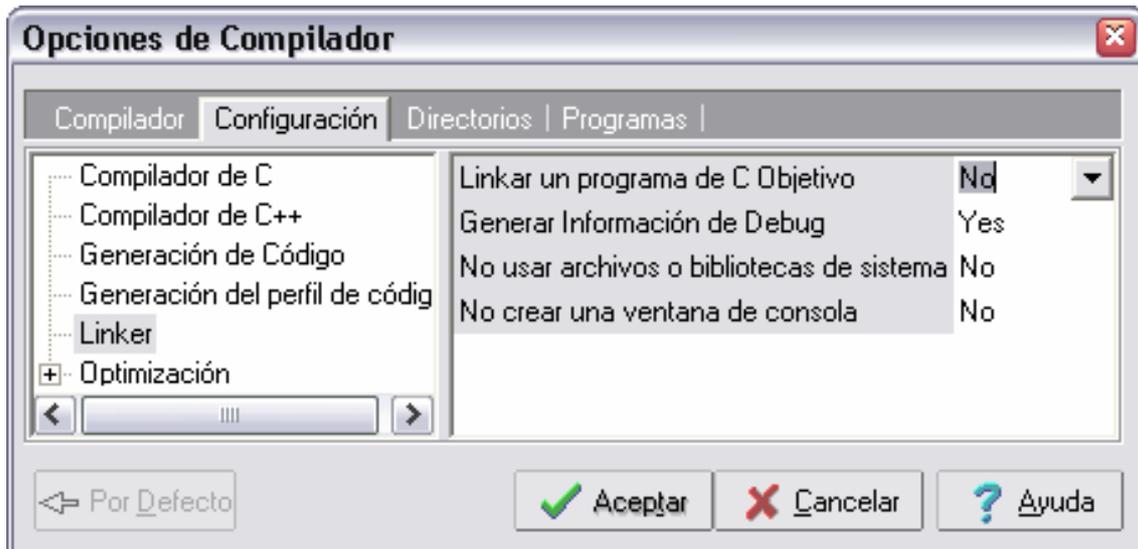
- *Generación del perfil de código.* (No hay imagen porque sólo tiene una opción)

Generación del perfil para análisis:

Sirve para generar la información necesaria para hacer un profiling del código. Como he dicho antes hay que actualizar el programa para que funcione correctamente. Lo veremos un poco más adelante.

Por cierto, comentar que el profiling no siempre funciona con algunas optimizaciones activadas, o algunas opciones de vinculado, como por ejemplo el parámetro `-s` del linker.

Dev-C++ Instalación y primeros pasos.



- *Linker.*

Linkar un programa de C objetivo.

Pues lo dicho, vincula código de Objective C, una especie de C orientado a objetos, al igual que C++, pero más parecido a C.

Generar Información de debug.

Equivale a la opción de compilador `-g3` Genera la máxima información posible para la depuración. Más información sobre argumentos para la depuración (inglés): <http://gcc.gnu.org/onlinedocs/gcc-3.2.3/gcc/Debugging-Options.html#Debugging%20Options>

No usar archivos o bibliotecas de sistema.

Sólo busca las librerías en los directorios especificados explícitamente en línea de comandos, por lo que no buscará las bibliotecas de los directorios por defecto del compilador.

No crear una ventana de consola.

El programa no creará una ventana de consola de símbolo del sistema.

Más información sobre argumentos para el linker (en inglés):

<http://gcc.gnu.org/onlinedocs/gcc-3.2.3/gcc/Link-Options.html#Link%20Options>

- *Optimización (No hay imagen)*

Realizar un número menor de optimizaciones

Equivale a la opción `-fexpensive-optimizations`

Optimizar (el parámetro es la letra 'O', NO el número cero)

Equivale a la opción `-O1` (que a su vez equivale a `-O`)

Optimizar más

Equivale a la opción `-O2`

La mejor optimización

Equivale a la opción `-O3`

Aquí hay otras opciones recomendadas para optimizar, reduciendo el tamaño del ejecutable.

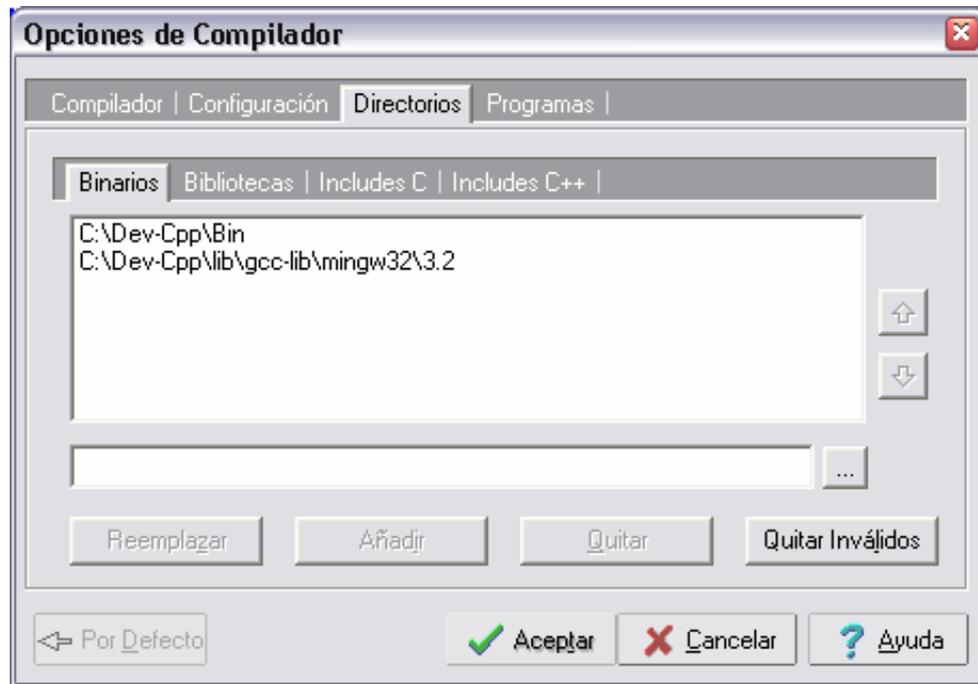
`-Os` optimiza para ahorrar espacio (Se introduce en los parámetros del compilador)

`-s` elimina la tabla de símbolos del ejecutable (Se introduce en los parámetros del linker)

Dev-C++ Instalación y primeros pasos.

Más información sobre la optimización y sus argumentos en la página (en inglés):
<http://gcc.gnu.org/onlinedocs/gcc-3.2.3/gcc/Optimize-Options.html#Optimize%20Options>

- Opciones del compilador : Directorios



Aquí podemos definir los directorios donde el programa buscará por defecto. Podemos definir directorios separados para los archivos ejecutables (binarios), las bibliotecas, o los ficheros de encabezados de C y C++

- Opciones del compilador : Programas (no hay imagen)

Por último, podemos cambiar los programas que el IDE ejecuta. Por ejemplo si en vez de usar el compilador gcc.exe, quisiéramos usar el compilador my_gcc.exe, deberíamos definirlo aquí. Como dice en el diálogo, y debido a que necesitaríamos un compilador que recibiera los mismos argumentos que el GCC, usaremos esto si queremos usar un cross-compiler (es decir un compilador que se ejecuta en una plataforma, y genera código de otra)

Cabe destacar, que todas las opciones que hemos visto aquí se aplicarán para todos los proyectos o ficheros que utilicemos con el Dev-C++. Esto lo comento porque la mayoría de estos apartados los podremos redefinir cuando usemos un proyecto. Por ejemplo podemos seleccionar unos argumentos de compilación, así como nuevos directorios donde buscar bibliotecas y encabezados, de forma que sean específicos para un proyecto (llamémosle proyecto1). Sin embargo el resto de proyectos que creemos (o los ficheros compilados de forma independiente que usemos) no se verán afectados por las opciones de ese hipotético proyecto1.

- Opciones del entorno:

- *Principal*

- C++ por defecto en nuevo proyecto*

Dev-C++ Instalación y primeros pasos.

Cuando creamos un nuevo proyecto, el lenguaje por defecto será C++. De todas maneras podemos cambiarlo en las opciones que nos da cuando creamos el proyecto.

Minimizar en la ejecución de programa

Minimiza el IDE cuando ejecutamos un programa (no cuando depuramos)

Doble clic para abrir archivos del administrador de proyecto

Si no se selecciona al hacer un simple clic en el explorador de proyectos se abrirá automáticamente el fichero

- *Interfaz*

De nuevo nada del otro mundo. Podemos establecer el idioma, el estilo de los iconos, así como varias opciones de representación de los datos.

- *Directorios*

Podemos seleccionar los directorios donde el IDE buscará sus archivos específicos, como el directorio con los ficheros de idiomas, o los ficheros con plantillas de proyecto.

- *Programas*

Ni idea de para qué sirve o su utilidad, cualquier ayuda será bien recibida ☺

- *Asociaciones*

Marca las casillas para asociar en el SO las extensiones que quieras para que se abran de forma predeterminada con el Dev-C++

- *Soporte CVS*

Para especificar el ejecutable del CVS (Concurrent Version System) El CVS es un método para que varias personas puedan trabajar en un mismo proyecto, subiendo el código a un servidor y bajando la versión más actual, de forma concurrente (es decir, varias personas a la vez sin que surjan problemas al actualizar un mismo fichero a la vez) Una explicación más detallada la tenéis en <http://www.olea.org/como-empezar-cvs/>

- Opciones del editor

Diferentes opciones para formatear el código, así como para su mejor comprensión. Por ejemplo podemos cambiar número de espacios en blanco que genera una tabulación, a qué extensiones de archivos se le debe aplicar la sintaxis resaltada (si trabajáis con ficheros linux es recomendable añadir .cc que es la extensión de ficheros C++)

También podemos cambiar las fuentes que usa el programa, o los colores y formato de la sintaxis resaltada (tiene algunas configuraciones predefinidas de los compiladores más comunes, como Borland o Visual C++)

La mayoría de las opciones son auto-explicativas, o tienen ayuda emergente al poner el puntero del ratón encima.

Sólo destacar dos opciones:

- *Código*

Aquí podemos añadir nuestros fragmentos de código para que luego estén disponibles en el menú insertar. Por ejemplo si usamos mucho una función específica o que hayamos creado, la podemos escribir aquí, para que con dos pulsaciones de ratón, podamos meterla en el código cuando queramos. Es muy útil para las plantillas de función de C++.

Dev-C++ Instalación y primeros pasos.

También podemos definir un código que se insertará por defecto en los proyectos vacíos.

- *Explorador de clases*

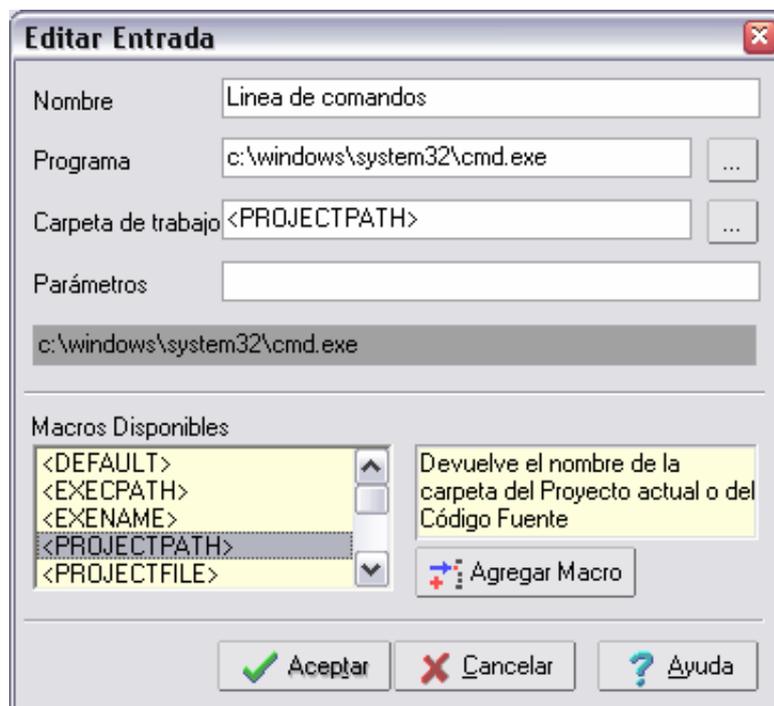
Podemos activar o desactivar el explorador de clases, así como los archivos en los que tiene que buscar, y si debe mostrar colores o los miembros y métodos heredados (si se trata de una clase derivada)

- Configurar atajos de teclado

Para definir las teclas que queremos usar para activar prácticamente todas las funciones del Dev-C++. Seleccionas el atajo y pulsas la tecla o combinación de tecla correspondiente. Para borrar un atajo de teclado existente, lo seleccionas y pulsas ESC.

- Configurar herramientas

En el diálogo que aparece cuando seleccionamos esta opción podemos definir nuevos elementos del propio menú herramientas:



Nombre: Nombre con el que aparecerá la herramienta en el menú.

Programa: Nombre y camino del archivo ejecutable.

Carpeta de trabajo: Carpeta donde se ejecutará la herramienta.

Parámetros: Argumentos a pasar al archivo ejecutable.

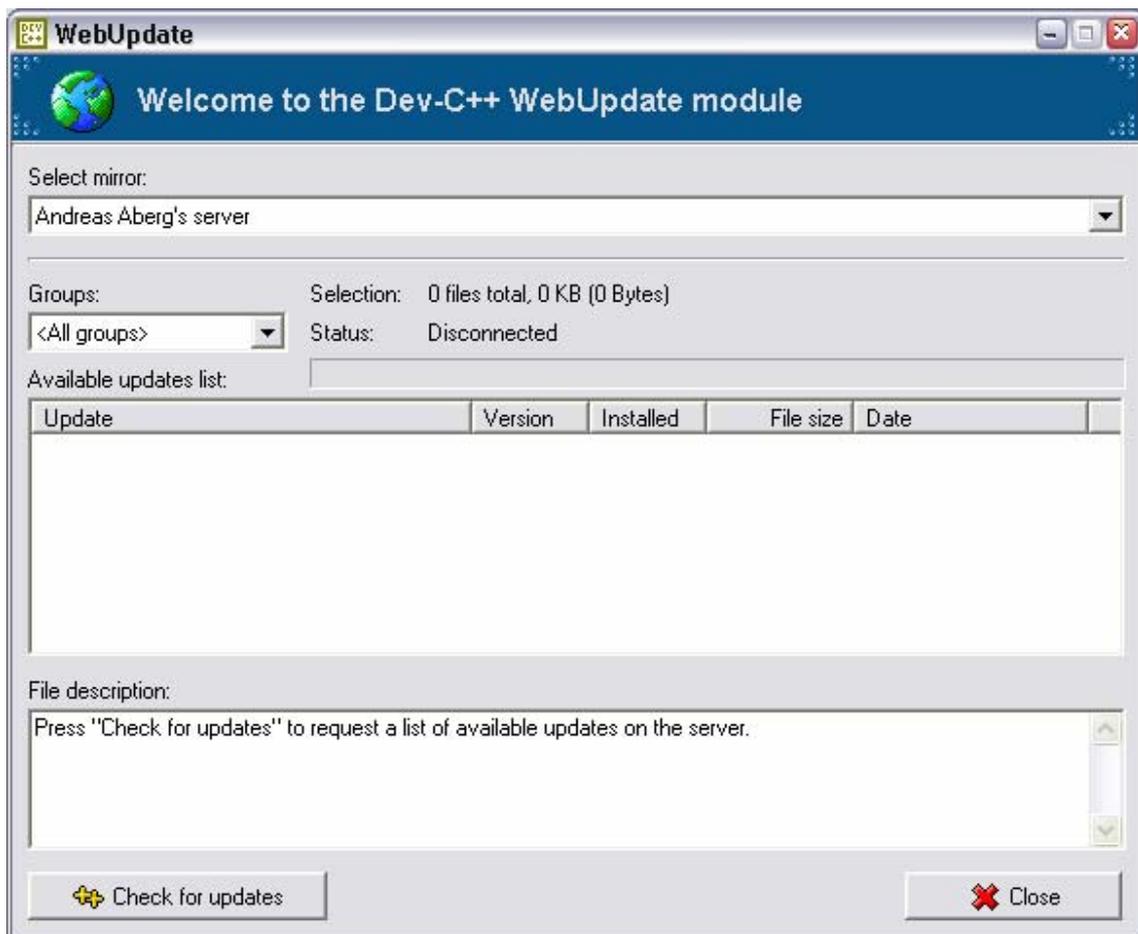
Tenemos algunas macros disponibles para facilitar las cosas. Al seleccionar una de ellas aparecerá la explicación a la derecha.

Dev-C++ Instalación y primeros pasos.

Como ejemplo del uso y la utilidad de las herramientas, tenéis en la captura de pantalla una definición que suelo usar mucho. Con esto aparecerá una opción en el menú de herramientas, que al pulsarla abrirá una consola de sistema en el directorio actual del proyecto ☺

Hay más posibilidades como, por poner un ejemplo, configurar un ensamblador para que con una pulsación de ratón nos compile el fichero actual (que suponemos es código ensamblador, obviamente) para que lo podamos vincular en nuestro proyecto

- Buscar actualizaciones.
Una útil herramienta: la actualización automática.



Ejecutando este programa podremos buscar actualizaciones del Dev-C++ por Internet. Pero no sólo actualizaciones, sino traducciones del programa en varios idiomas, las sugerencias del inicio traducidas así como documentación de referencia (normalmente en inglés) Y por si fuera poco los DevPacks. (Esto lo explico en el siguiente punto, de momento considerad que son nuevas bibliotecas para el Dev-C++) Es bueno usarlo una vez al mes o así, sobre todo porque los DevPacks se actualizan aquí mucho más a menudo que en las páginas oficiales, además de poder encontrar DevPacks que no se encuentran en las dichas páginas.

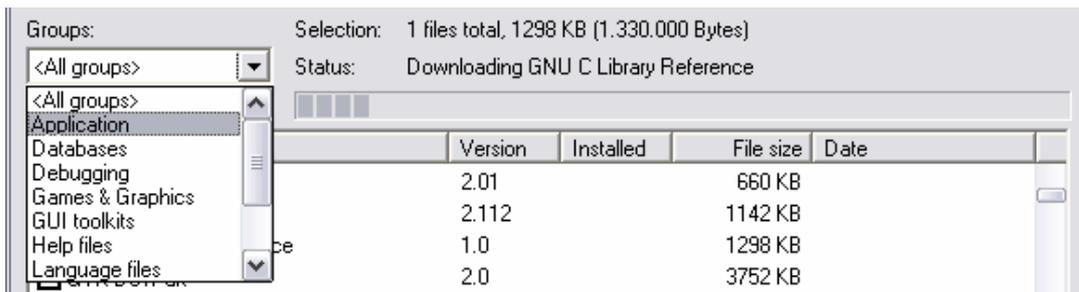
El funcionamiento es sencillo:

En la pantalla *Available updates list*, el programa nos dice qué actualizaciones están disponibles. Con la lista *Groups* que está justo encima de esta pantalla a la izquierda tenemos varios filtros para limitar el tipo de actualizaciones que queremos que el programa nos muestre (por ejemplo, sólo actualizaciones del dev-c++, o bien dev-packs orientados a multimedia, o solamente dev-packs de documentación... etc)

Dev-C++ Instalación y primeros pasos.

En la pantalla *File Description*, tenemos información útil acerca de la actualización que nos estamos descargando.

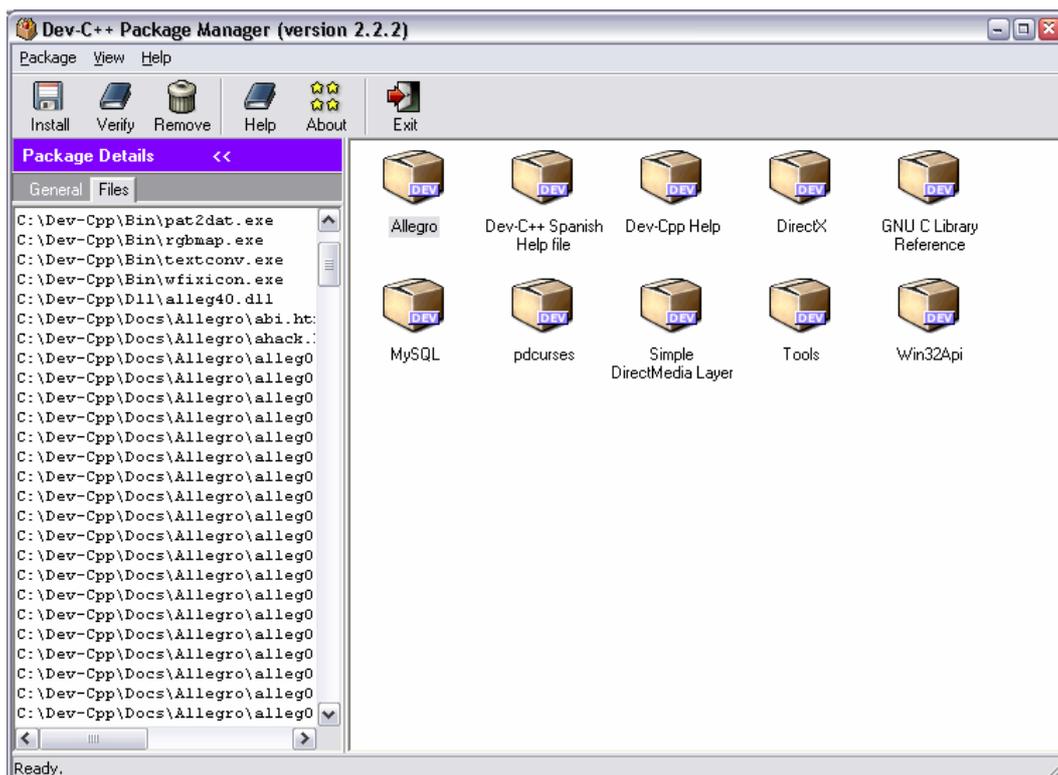
Simplemente tenemos que pulsar el botón “Check for updates”, y en unos instantes nos aparecerá una lista de todas las descargas disponibles. Simplemente marcamos las casillas de las actualizaciones que deseamos descargar, y pulsamos el botón “Download selected” y comenzará la descarga. Al finalizar se instalará todo automáticamente, ya sea una actualización, un dev-pack o un fichero de ayuda (éstos últimos se descargan en el directorio Dev-cpp/docs)



- Package Manager

Con esta opción, se ejecutara una herramienta llamada Package Manager (o pacman a secas ☺) Con ella puedes gestionar todos los DevPacks que tienes en el sistema, pudiendo instalarlos, desinstalarlos o simplemente obtener información acerca de ellos.

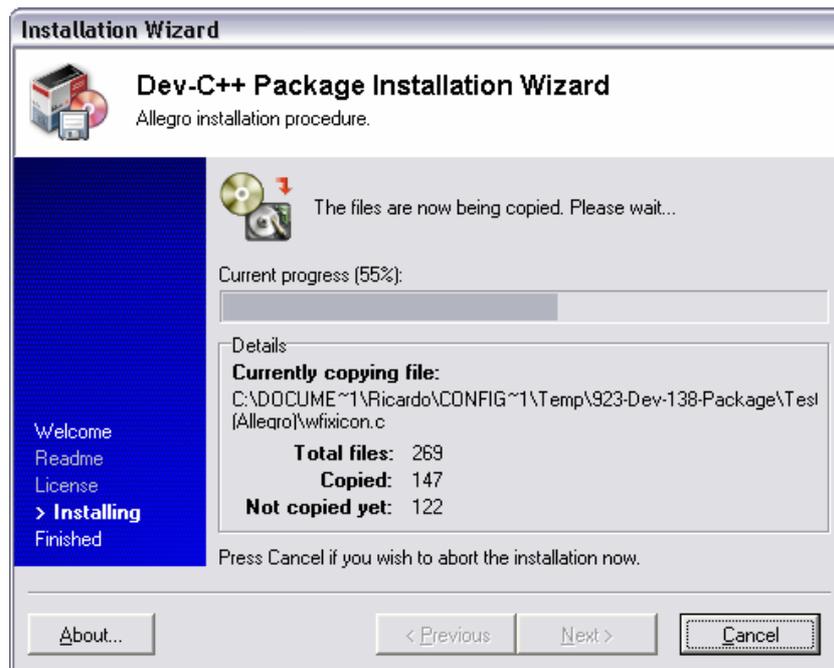
Los DevPacks son archivos que contienen una agrupación de bibliotecas, documentación o utilidades para el Dev-C++. Al instalarlo automáticamente distribuye los ficheros que contiene en los directorios adecuados (las bibliotecas en dev-Cpp/lib, los ejecutables en dev-Cpp/bin...) Hay DevPacks de utilidades, otros de documentación, y otros con bibliotecas, que normalmente también incluyen documentación y plantillas de proyecto. Los DevPacks están ya comprimidos.



Dev-C++ Instalación y primeros pasos.

Aquí tenemos la ventana principal. Se gestiona todo con los iconos de arriba. En la captura está activada la pestaña 'files' Archivos, en este caso del DevPack de la librería Allegro. Puedes ver una lista de todos los archivos que instala el DevPack y donde. En la otra pestaña 'General' encontrarás información acerca del DevPack (lo que contiene, su versión, y referencias) Si quieres deshacerte de la lo que contiene el DevPack simplemente desinstálalo y todo quedará como antes.

Para instalar un DevPack puedes utilizar el pacman, o simplemente hacer doble clic en el archivo del DevPack que quieras instalar (extensión .devpack). Te aparecerá entonces un diálogo como este:



En esta captura se muestra el proceso de instalación, bastante detallado. Antes puedes encontrar el readme o la licencia (normalmente GNU ©) Vamos, como un InstallWizard típico de Windows.

Los DevPack se pueden descargar con el vUpdate, que es la opción más recomendada. En este caso, los DevPack se guardarán en el directorio /Dev-cpp/devpacks pero puedes copiarlos, moverlos e instalarlos desde cualquier directorio.

También los hay disponibles en algunas páginas, como puede ser <http://www.bloodshed.net/dev/packages/index.html>

Menú CVS:

Como no uso el CVS no tengo ni idea de cómo utilizarlo, así que acepto cualquier tipo de colaboración.

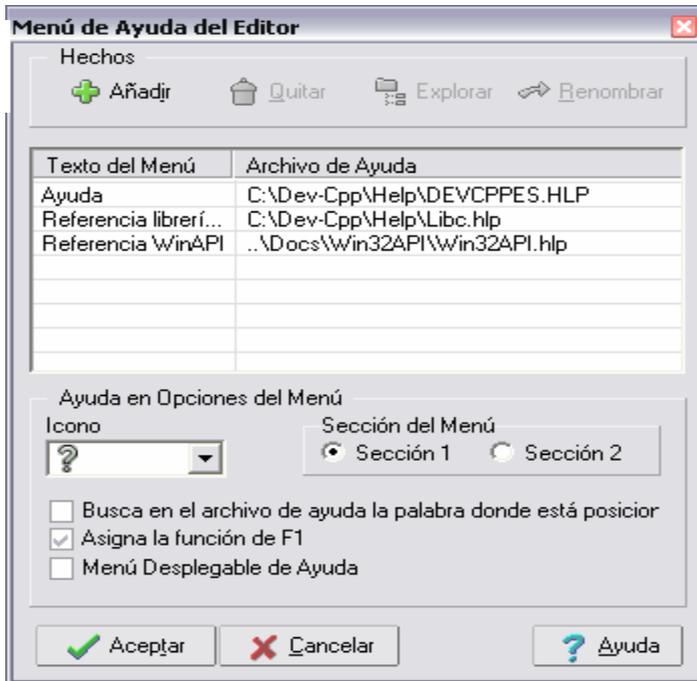
Menú Ventana:

Típico de Windows: seleccionar pantalla completa, conmutar entre ventanas...

Dev-C++ Instalación y primeros pasos.

Menú Ayuda:

Más de lo mismo, pero con una pequeña excepción. Podemos añadir elementos a este menú para tener a mano diferentes ficheros de ayuda o referencia. Para eso hacemos clic en *configurar el menú de ayuda*, y tendremos un diálogo como este:



Podemos añadir nuevos ficheros de ayuda, seleccionar un icono o incluso asignar ese nuevo fichero como el predeterminado a la hora de buscar, asignándole la tecla F1. Yo tengo en mi menú la traducción al español del fichero de ayuda del Dev-C++, una referencia rápida de la librería estándar de C, y la ayuda de la API Win32. Todo descargado con el vUpdate y añadido al menú ☺

Hasta aquí todo lo referente a los menús y opciones del Dev-C++. Como guinda vamos a ver cómo crear un proyecto y ver las opciones que tenemos para la gestión de proyectos.

Creación de proyectos.

Vamos a basarnos en un proyecto ya creado que viene con los ejemplos del Dev-C++. Partiendo de los ficheros fuente, vamos a crear un proyecto semejante al del ejemplo.

Utilizaremos los archivos situados en Dev-Cpp\Examples\FileEditor. Si vas a ese directorio encontrarás un archivo llamada FileEditor.dev. Es el fichero del proyecto. Haz doble clic para verlo si quieres. Lo que nosotros vamos a hacer es generar otro proyecto semejante a ese. Cierra el proyecto (Archivo/cerrar proyecto)

Copia los siguientes archivos del directorio anterior a otro directorio temporal (el que tú quieras): main.h, main.c menu.rc

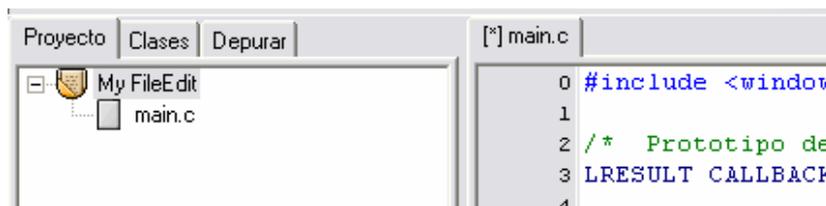
Ahora pulsa en archivo/nuevo/proyecto. Te debería salir un diálogo como este:

Dev-C++ Instalación y primeros pasos.



Selecciona la aplicación para Windows (Windows application) como nombre del proyecto 'My FileEditor' y asegúrate de seleccionar 'En C' para que los archivos tengan la extensión .c. Dev-C++ distingue por defecto un archivo a compilar como C a todo aquel que tenga extensión .c Y a su vez, todos los archivos de extensión .cpp serán compilados como C++. Sin embargo eso se puede cambiar dentro de las opciones del proyecto. Pulsa aceptar y te saldrá un diálogo para poner el directorio donde guardar archivo de proyecto. Guárdalo en el lugar donde copiaste los tres archivos anteriores.

Verás que automáticamente se crea un fichero o que acabas de elegir (Windows application) tiene asociada esa plantilla, con lo que por defecto genera ese código. Puedes crear tus propias plantillas, aunque eso se escapa a lo que pretende este documento. (Sólo una pista: son simples ficheros de texto. Ábrelos con cualquier editor, y comprobarás que no es muy difícil averiguar cómo funcionan ☺)



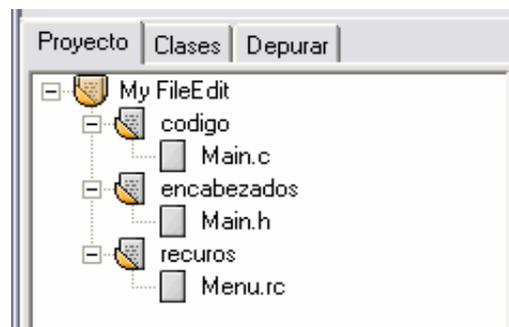
Aunque esta característica es muy útil, ahora mismo no nos interesa, pues ya tenemos los ficheros creados. Por tanto tendremos que eliminar ese archivo del proyecto. Verás que en la pestaña del archivo tiene un asterisco tal que así:[*] Eso significa que el archivo se ha modificado y no se ha salvado. Sólo tienes que cerrar el archivo (botón derecho en la pestaña del archivo y luego cerrar) Eliges no a la opción de salvar y ya está.

Es la hora de incluir nuestros propios ficheros en el proyecto. Pulsa en el icono de agregar ficheros (también lo tienes en el menú proyecto/Añadir a proyecto) En el diálogo que te sale selecciona los tres ficheros de antes (main.c main.h menu.rc) y dale a aceptar. Ya tienes los ficheros en el proyecto (fíjate en el explorador de proyectos)

Dev-C++ Instalación y primeros pasos.



Pero imagínate que tu próximo proyecto consta de 12 ficheros de encabezado, 20 de código y 7 de recursos... quedaría un poco desordenado ¿no? No hay problema, porque podemos definir carpetas para cada archivo o grupo de archivos. Para ello haz clic derecho en el nombre del proyecto que aparece en el explorador de proyectos, selecciona 'añadir carpeta' y eliges el nombre. En este caso crea tres carpetas : 'codigo' 'encabezados' y 'recursos'. Luego arrastra a cada una de ellas los ficheros, main.c main.h y menu.rc, a la carpeta que le corresponda. Debería quedar así:



Ahora compilamos el proyecto. Pulsa el icono o en el menú. No debería dar ningún error. Una vez hecho esto vete al directorio donde guardaste el proyecto. Ahora deberías tener más ficheros, pero el que más nos importa es My FileEdit.exe. Si lo ejecutas verás un editor de ficheros de texto muy limitado.

El archivo ocupa 36Kb (36.224 bytes para ser exactos) Y no hemos puesto ninguna optimización... quizá se pueda mejorar eso... y el icono es un poco triste, tendremos que cambiarlo también, ¿no te parece? Pues para eso vamos a menú proyecto/opciones de proyecto.

Nos aparecerá un diálogo con varias pestañas. Estamos en la *principal*, donde podemos definir el nombre del proyecto y su tipo (Windows, consola, DLL...) Y también podemos seleccionar el icono. Pulsamos en el botón 'biblioteca' o si tenemos algún icono gracioso podemos pulsar el botón 'explorar' para buscarlo en el disco duro. Elige uno original -hay que ser un poco imaginativo ☺ - y acepta. La próxima vez que reconstruyamos el proyecto, nuestro ejecutable tendrá el icono elegido.

En la pestaña *archivos* podremos referirnos a cada uno de los archivos de nuestro proyecto de forma individual, por ejemplo para definir cuál de ellos tiene prioridad en la compilación, o si algunos de ellos deben forzarse a ser compilados como C++, o incluso si queremos que alguno de ellos no se compile. Esto sólo funcionará para ficheros de código, ya que son los únicos que se compilan.

Las siguientes tres pestañas (*compilador*, *parámetros* y *directorios*) Son idénticas a las del menú Herramientas/opciones de compilador. Vamos entonces a probar unas optimizaciones. Ve a compilador/optimización y selecciona 'la mejor optimización'. Acepta y recompila todo el proyecto (Ejecutar / reconstruir todo) Miramos el ejecutable de nuevo y vemos que ahora tiene un icono, pero ¡ocupa un poco mas! Es posible que algunas optimizaciones hagan que el código "engorde", eso es algo normal (sin pasarse,

Dev-C++ Instalación y primeros pasos.

pasar de 30Kb a 1Mb sería pasarse) Pero bueno, nosotros queremos código compacto aunque sea un poco más lento (y si no lo queréis lo siento, pero no me desmontéis el tutorial ☺) Volvemos a opciones de proyecto, y vamos a la pestaña *parámetros*. Donde pone *linker* añadimos *-s*, aceptamos y recompilamos todo de nuevo. Vaya, no está mal, hemos pasado a 15Kb, menos de la mitad ☺

Si nuestro proyecto utilizara bibliotecas o encabezados específicos, deberías incluirlos en su lugar correspondiente en la pestaña *directorios*.

En la pestaña *construcción* podemos especificar el directorio donde se genera el ejecutable una vez compilado, vinculado y generado dicho ejecutable; el directorio donde se guarda el código objeto, una vez compilados los archivos; así como el nombre de archivo que tendrá el ejecutable.

En la pestaña *archivo* podemos introducir nuestros propios comandos al Makefile del proyecto. Si no sabes qué es eso no te preocupes, cuando lo necesites te enterarás, simplemente recuerda que éste es el lugar a donde ir.

Por último, en la pestaña *información*, podemos incluir cierta información sobre el proyecto, como la versión del programa. Esto se incluye en un fichero de recursos generado por Dev-C++

Bueno, hemos creado nuestro primer proyecto con éxito, lo hemos compilado, e incluso optimizado. Es hora de pasar a un asunto más serio: la depuración del proyecto.

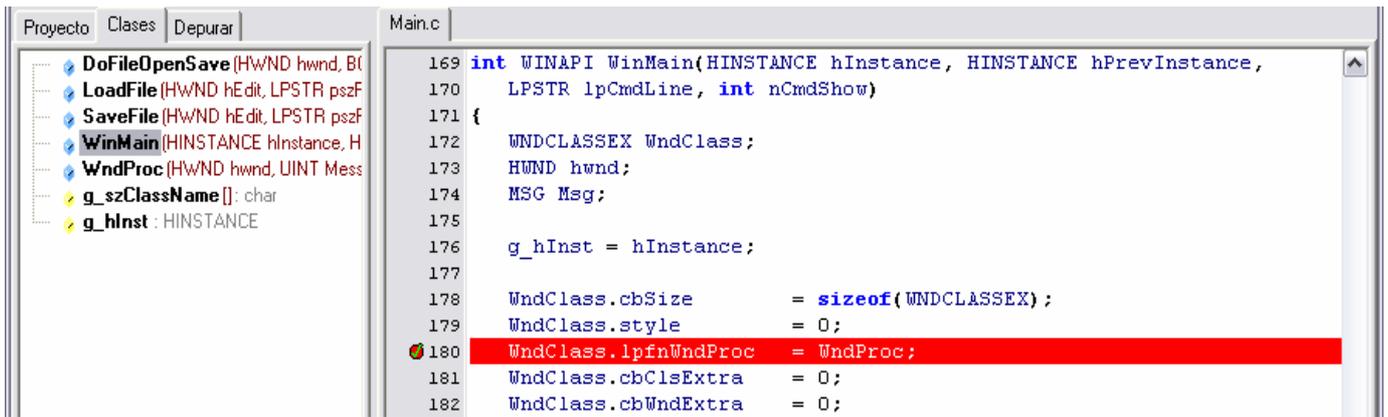
Uso del depurador.

Para iniciarnos en la depuración de un programa, partiremos de nuestro proyecto anterior 'My FileEdit' Primero tenemos que decirle al compilador que genere la información necesaria para que el depurador pueda mostrarnos el programa y sus elementos en tiempo de ejecución. Dicho de otras palabras tenemos que hacer un ejecutable con información de depuración. Para eso abrimos el proyecto y sus propiedades. En la pestaña *compilador / linker* activamos la opción 'generar información de debug'. También vamos a las opciones de optimización y desactivamos todas. Quitamos también el argumento *-s* del linker en la pestaña *parámetros*. Por último reconstruimos TODO el proyecto.

Ahora el ejecutable generado será bastante más grande, del orden de 337Kb. Esto es así porque toda la información necesaria para la depuración está incluida en el .exe. Ni que decir tiene que no es nada recomendable distribuir un programa con información de depuración, puesto que, además de ocupar más, se ejecuta de una manera más lenta.

Muy bien, una vez hecho esto podemos comenzar. Antes de pulsar el botón *depurar* tienes que poner un breakpoint para decirle al depurador que cuando llegue a ese lugar debe detenerse. Si no haces esto el programa se ejecutará sin pausa alguna, lo que no nos sirve de mucho. Por ejemplo abrimos el archivo *main.c*, vamos a la línea 180 (Dentro de la función *WinMain*, usa el explorador de clases para buscarla) y pulsa CTRL-F5 (o menú *depurar / establecer punto de ruptura*). Debería quedar algo así:

Dev-C++ Instalación y primeros pasos.



```
169 int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,
170 LPSTR lpCmdLine, int nCmdShow)
171 {
172     WNDCLASSEX WndClass;
173     HWND hwnd;
174     MSG Msg;
175
176     g_hInst = hInstance;
177
178     WndClass.cbSize = sizeof(WNDCLASSEX);
179     WndClass.style = 0;
180     WndClass.lpfnWndProc = WndProc;
181     WndClass.cbClsExtra = 0;
182     WndClass.cbWndExtra = 0;
```

Ahora pulsa F8 para comenzar la depuración. Verás que la línea cambia de rojo a azul y que además tiene una flechita azul a la izquierda. Eso significa que el punto de ejecución del programa se encuentra en esa línea, o lo que es lo mismo: La siguiente instrucción que se ejecutará será la de la línea azul. De esto se deduce que al establecer un breakpoint, el programa se parará antes de ejecutar la instrucción de la línea donde lo hemos establecido.

También deberías observar que se ha activado la pestaña de depuración del área 4 (abajo del todo ☺) y tienes varios comandos a tu disposición.

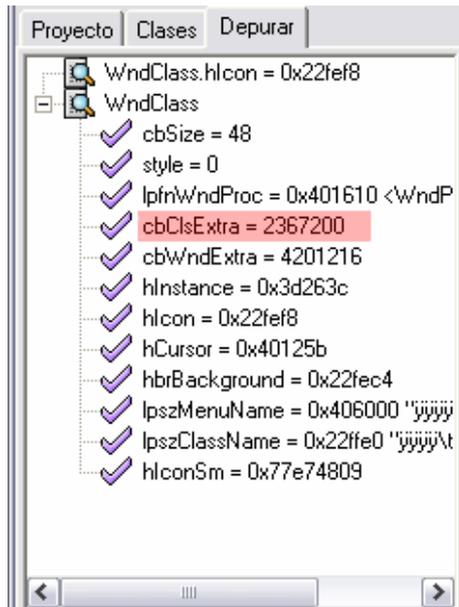
Vamos a aprender a observar el valor de las variables en tiempo de ejecución. Para eso pulsamos F4 o pulsamos el botón “Añadir watch”(automáticamente se activará la pestaña *Depurar* del explorador de clases. Introducimos “WndClass.lpfnWndProc” (sin las comillas) Es justo la variable que se inicializará dentro de dos instrucciones (la siguiente a la línea azul) Vemos que la variable tiene un valor aleatorio (una variable no inicializada apunta a memoria, por lo que puede tener cualquier valor) Vamos a ejecutar el programa paso a paso. Para eso pulsamos en *Siguiente paso* Vemos que ahora la línea siguiente se pone azul (la anterior sigue roja para indicarnos que existe un breakpoint)



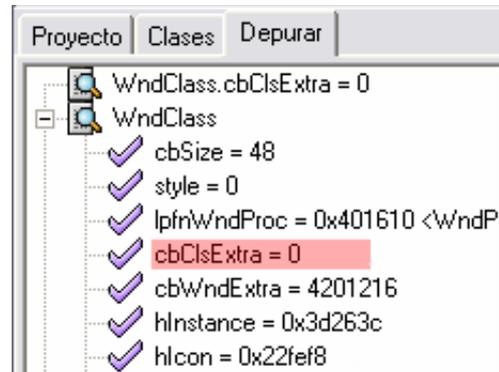
```
180 WndClass.lpfnWndProc = WndProc;
181 WndClass.cbClsExtra = 0;
```

Pero fíjate en el área de depuración del explorador de clases. ¡Ya no tenemos nuestra variable! Bueno, no es del todo correcto, no ‘sólo’ tenemos nuestra variable, sino que tenemos todos los campos de los que consta el struct WndClass. El depurador es muy inteligente y nos lo muestra todo ☺

Dev-C++ Instalación y primeros pasos.



Pulsamos de nuevo en *Siguiente paso* y veremos que la variable se actualiza al llegar a la línea en la que se inicializa. Ahora `WndClass.cbClsExtra` vale 0



La diferencia entre los botones *siguiente paso* y *avanzar paso a paso*, es que el segundo se introduce en las funciones y el primero no. Por ejemplo si la siguiente línea a ejecutar es:

```
A = miFunción()  
B = 5
```

Si pulsamos *siguiente paso* pasaremos a la línea `B = 5;` El código de la función `miFunción()` se ejecuta, por supuesto, pero no accedemos a él. Sin embargo si pulsamos *avanzar paso a paso*, nos meteremos en la función `miFunción()` y continuaremos ejecutando las instrucciones dentro de la función. Cuando la función retorne, volveremos a la línea `B = 5.`

El botón *saltar paso* (continuar) reanuda la ejecución normal del programa hasta que nos topemos con otro breakpoint, o bien el programa finalice. Como podemos definir los breakpoints que queramos en cualquier momento, vamos a buscar la línea 160, en la que debería haber una función llamada `PostQuitMessage(0);` Establecemos un breakpoint en esta línea y le damos al botón *saltar paso* (continuar).

Parece que no pasa nada, ¿verdad? Ha salido la ventana del editor de texto y podemos interactuar con ella. Como no podemos hacer mucho vamos a cerrarla.

¡Ups! Parece que hemos regresado al código y la ejecución del programa se vuelve a detener. Esto ocurre porque hasta que no llegamos a la línea en la que hemos situado el breakpoint el programa continúa igual, como si no estuviéramos depurando. Un programa de Windows no es una excepción. Hemos hecho un breakpoint en una función que se ejecuta cuando cerramos la ventana, por eso hasta entonces podíamos interactuar normalmente con la aplicación. Podemos dar por finalizada nuestra clase introductoria a la depuración y por tanto el programa. Pulsamos el botón *Parar ejecución*.

Dev-C++ Instalación y primeros pasos.

FAQ (Preguntas más frecuentes)

1) Cuando ejecuto el programa la ventana de MS-DOS se abre y se cierra muy rápido, o no se abre en absoluto. ¿Cómo puedo ver los resultados de mi programa?

Que la ventana de comandos del Windows haga eso es normal: ejecuta el programa y finaliza, y lógicamente al finalizarse la ventana se cierra. Para que puedas ver los resultados de tu programa tienes dos opciones:

-Ejecuta el programa directamente desde línea de comandos.

-Haz que el programa no finalice hasta que el usuario pulse una tecla.

Esto se hace de dos maneras:

- Poniendo **getchar()**; (si usas C) o **std::cin.get()**; (si usas C++) al final del programa, justo antes del **return**. Con esto el programa espera a que el usuario pulse una tecla antes de continuar, con lo que consigues tu propósito. Necesitas incluir *stdio.h* (en C) o *iostream* (en C++).
- utilizar la función estándar **system("pause")** Esto tiene el inconveniente de que no es portable: sólo funcionará en Windows. Necesitarás incluir *stdlib.h* (en C) o *cstdlib* en (C++)

- Utilizando un comando personalizado del menú herramientas para ejecutarlo. Pulsa en el menú herramientas/configurar herramientas. Pulsa el botón añadir y te saldrá un diálogo, complétalo así:

Nombre: ejecutar en consola de comandos (o el que tú quieras ponerle)

Programa: c:\windows\system32\cmd.exe

Carpeta de Trabajo: <PROJECTPATH>

(<PROJECTPATH> es una macro, la puedes incluir seleccionándola en la lista de "macros disponibles" y pulsando el botón "añadir macro")

Parámetros: /c "<EXENAME>" && PAUSE

(<EXENAME> es una macro, se añade igual que la anterior.)

Una vez hecho esto, cada vez que quieras ejecutar un programa de consola, vete al menú herramientas / ejecutar en consola de comandos -como ves se ha añadido esta opción nueva al menú. Así ejecutarás el programa en una consola de comandos, pero no se cerrará al finalizar, sino que imprimirá el mensaje "pulse una tecla para continuar..." al estilo de Visual C++

2) Vale, he puesto la primera opción de la pregunta 1) ;;pero el programa se sigue cerrando igual!! ¿Qué hago?

Eso es debido a que anteriormente has pedido al usuario que introduzca algo por teclado (o al menos por el dispositivo de entrada estándar) y en el buffer de entrada ha quedado registrado algún carácter. Normalmente es el carácter de fin de línea, (o digamos de una forma más clara, el carácter que se genera cuando pulsas ENTER) que algunas funciones de entrada no retiran del buffer. Eso implica que tu anterior **getchar()**; o **std::cin.get()**; recoge dicho carácter del buffer, con lo que una vez cumplida su función el programa continúa y finaliza. La solución es volver a poner otro

Dev-C++ Instalación y primeros pasos.

`getchar()`; o `std::cin.get()`; justo detrás del otro Si necesitas utilizar más de 3 seguidos para que esto funcione entonces es probable que debas revisar tu programa y el modo en que tratas la E/S.

3a) Dice que no encuentra la biblioteca conio.h. Pero si la usaba con mi/s anterior/es compilador/es. ¿Porque éste no la incluye?

3b) El compilador me da el error [Linker error] undefined reference to `clrscr' ¿que pasa?

Ante todo, estos errores vienen al utilizar la biblioteca conio (CONsole Input/Output) la cual no es una biblioteca estándar. (Es decir, no se encuentra en todos los compiladores ni en todas las plataformas) Fue creada por Borland para sus compiladores y entornos DOS (consola de comandos). No es estándar porque la E/S a la consola del sistema es dependiente del sistema operativo (de hecho el estándar ni siquiera contempla que exista una pantalla o teclado, simplemente habla de dispositivos estándar de E/S) El Compilador sólo incluye las bibliotecas estándar, ni una más. ¿Que hacer entonces? Hay varias opciones:

La primera sería olvidarse de esa biblioteca y aprender a vivir sin ella. Puede hacerse, no es más que cuestión de acostumbrarse.

Otra opción sería implementar tú mismo las funciones utilizando la API del SO. Vale, es algo rollo, pero aprenderás más. Te aseguro que no es muy complicado. Es una buena opción.

Si realmente no puedes vivir sin esa biblioteca hay otros medios.

Si no te gusta meterte en berenjenales, puedes hacer una cosa. El Dev-C++ incluye en realidad cierto "soporte" a esa biblioteca. Hay un fichero conio.c con la implementación de algunas -en realidad MUY pocas- funciones de la biblioteca original. Para poder utilizarla necesitas incluir este fichero en tu proyecto y compilarlo con él. Ese fichero se encuentra en el directorio `\dev-cpp\include`. Por supuesto también tienes que incluir conio.h

Por último tienes una biblioteca conio para el Dev-C++ un poco más desarrollada y con el código fuente (por si algún día quieres saber como funciona) en el siguiente link: <http://c.conclase.net/conio/index.php>

4) El compilador me da el error ``main' must return `int' return type for `main' changed to `int'`

Bueno, a partir de este punto olvida TODO lo que hayas visto u oído acerca del valor de retorno de la función `main()`. La función `main()` siempre retornó, retorna y retornará `int`, tanto en C como en C++ (principalmente porque aun en C++, `main()` se considera una función de C). Siempre fue así en el estándar, el problema es que muchos compiladores se olvidaron de que `main()` no es una función común -o al menos relajaron esa restricción- y permitían cualquier tipo de retorno. Los compiladores que siguen el estándar ahora no lo permiten, y dan o bien advertencias, o, como en este caso, errores. La solución es fácil: declara `main()` como `int main()` y pon un `return 0;` al final de tu programa.

5) He compilado un programa 'hola mundo' en C++ ¡y el código ocupa 439Kb! ¿No es un poco exagerado? Si lo compilo utilizando `printf()` el código ocupa sólo 22Kb...

Bueno si, es exagerado, pero no hay manera de evitarlo. Es un problema del minGW, que no enlaza las operaciones de E/S de C++ con la biblioteca en tiempo de ejecución de Windows, sino que lo hace estáticamente -parece ser que por problemas legales. Esto hace que el propio código objeto de la biblioteca se incluya en el

Dev-C++ Instalación y primeros pasos.

ejecutable con lo que su tamaño aumenta considerablemente. La parte buena de este asunto es que este aumento no es proporcional, sino constante: un programa el doble de grande que un 'hola mundo' (en código) no va a ocupar el doble, sino un poco más.

Puedes reducir el tamaño del ejecutable añadiendo la opción **-s** como parámetro al linker (enlazador) tal y como vimos anteriormente. También obtienes el mismo resultado ejecutando el comando 'strip.exe nombre_ejecutable', que se encuentra en el directorio /bin del Dev-C++ Estas opciones te dejarán el 'hola mundo' en unos 200Kb.

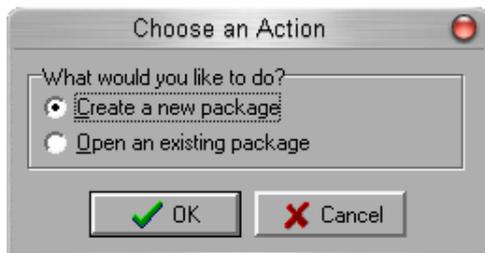
Si realmente necesitas mantener el tamaño de código muy pequeño deberías utilizar la E/S de C, o bien utilizar un programa compresor de ejecutables como por ejemplo el upx (<http://upx.sourceforge.net/>) que también es de código libre.

Dev-C++ Instalación y primeros pasos.

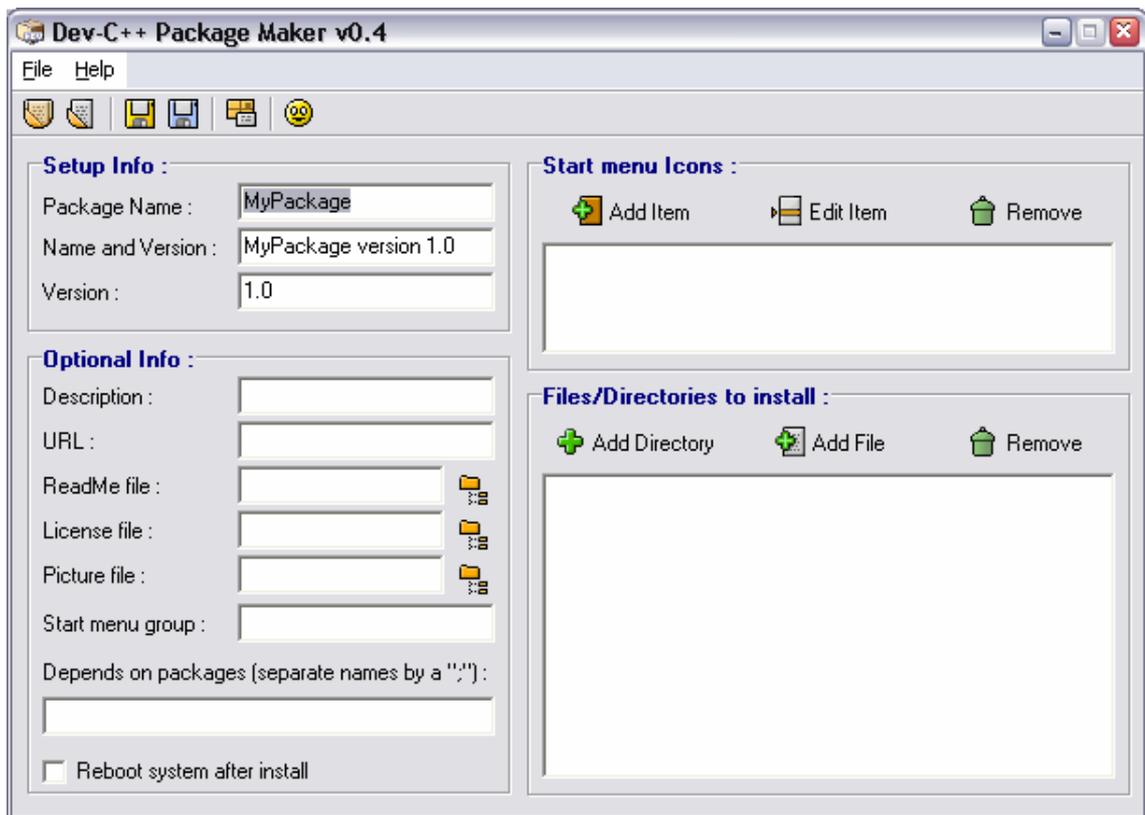
APÉNDICE: Crear tus propios dev-packs.

Con la nueva versión del Dev-C++ viene incluida una utilidad para crear y abrir los devpacks. Éstos ficheros no son más que un simple fichero TAR para agrupar todos los ficheros que lo componen, y comprimidos con BZIP2. La diferencia es la estructura de directorios, y un fichero especial que incluye toda la información. Éste fichero reside dentro del propio devpack y es un simple fichero de texto con la extensión .devPackage

Si ejecutamos Dev-Cpp\PackMaker.exe obtendremos un cuadro de diálogo como este:



donde seleccionaremos "Create a new package" para hacer nuestro propio devpack, y obtendremos el siguiente diálogo:

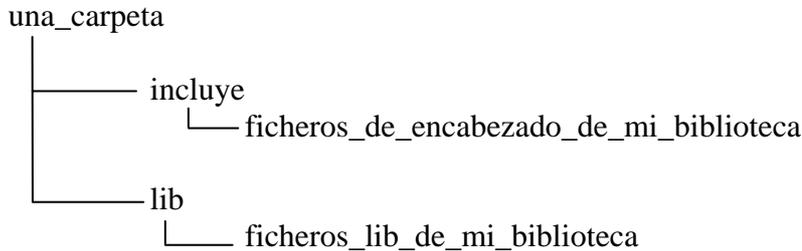


Toda la información que introducimos en los campos de la izquierda es la información que se mostrará cuando instalemos el dev-pack. Incluso podemos definir nuestro propio grupo en el menú inicio, así como los iconos que queremos que aparezcan en él. Por supuesto estos iconos tendremos que añadirlos nosotros, y decirle al programa que los utilice (por medio del recuadro "Start menú icons")

Lo más importante sin embargo es el recuadro "Files/directories to install" Aquí definiremos qué archivos y directorios se crearán cuando se instale el devpack, y dónde. Podemos elegir varias ubicaciones, pero parece ser que sólo la primera funciona, al menos en esta versión:

Dev-C++ Instalación y primeros pasos.

<app>\ Indica que todo lo que haya en la carpeta que hemos seleccionado será instalado *en el directorio de la aplicación*. Es decir, que la carpeta que hemos seleccionado, no se creará, sino que se creará todo lo que esta contenga. Por tanto, dentro de esta carpeta tenemos que definir una estructura de directorios, que podría ser algo así para el caso de una biblioteca:



Esto crearía un directorio incluye y un directorio lib en el directorio de instalación del dev-c++ y copiaría en ellos los ficheros.

Una vez está todo listo, pulsamos el botón “Build package” (el 5º empezando por la izquierda ☺) y ya tenemos nuestro Dev-Pack, como reza el programa, listo para su chequeo y distribución.

Sin embargo el programa no deja descomprimir el dev-pack, aunque se puede hacer con cualquier utilidad de descompresión, como por ejemplo WinRar. (<http://www.winrar.com>) Debido a esto, y a que en anteriores versiones no disponía de este programa, he creado dos ficheros Batch para comprimir y descomprimir dev-packs en línea de comandos. Ya no uso el de comprimir, ya que para *crear* un Dev-Pack es mucho más cómodo el programa anterior –lo recomiendo, aunque sólo sea para generar el fichero .devPackage. Sin embargo para descomprimir sigo usando la línea de comandos, me parece más cómodo.

Aquí tenéis los dos, funcionan al menos en WindowsXP, no los he probado en ningún otro sistema, así que si encontráis algún fallo comunicádmelo por favor.

Dev-C++ Instalación y primeros pasos.

ExpandDevpak.bat

```
@ECHO OFF

@cls
@COLOR 0F

ECHO.
ECHO -----
ECHO      Para utilizar este .bat se necesitan las utilidades
ECHO      TAR.EXE y BZIP2.EXE incluidas en el devpak "Tools Devpak"
ECHO      disponible a traves de la herramienta de actualizacion
ECHO      automatica del Dev-C++
ECHO -----
ECHO.
@PAUSE
ECHO.
@CLS

IF "%1" == "" GOTO ERROR1
IF NOT EXIST %1.devpak GOTO ERROR2
IF NOT EXIST C:\DEV-Cpp\bin\NUL GOTO NOTEXISTDEFAULTPATH

@SET PATH = %PATH%;C:\DEV-Cpp\bin\

IF EXIST ./%1/NUL GOTO DIREXIST

:EXEC
ECHO.
ECHO Desempaquetando...
@COPY %1.DevPak C%1.DevPak > NUL
@REN C%1.DevPak %1.tar.bz2
ECHO.
ECHO Descomprimiendo...
@C:\DEV-CPP\BIN\BZIP2 -d %1.tar.bz2
@C:\DEV-CPP\BIN\TAR -xvf %1.tar
@DEL %1.tar
GOTO ENDSUCC

:NOTEXISTDEFAULTPATH
IF "%2" == "" GOTO ERROR3
IF NOT EXIST %2\bin\NUL GOTO ERROR4
@SET PATH = %PATH%;%2
GOTO EXEC

:DIREXIST
ECHO ATENCION! El directorio '%1' ya existe, si continua los ficheros que
ECHO contenga se sobrescribieran.
ECHO Pulse CTRL-C para detener el proceso
@PAUSE
GOTO EXEC

:ERROR1
ECHO USO: Expand_Devpak.bat nombredevpak [Directorio_instalacion_DevC++]
ECHO (El nombre del devpak sin extension)
GOTO END

:ERROR2
ECHO No se ha encontrado el fichero '%1.DevPak'.
ECHO Asegurese de introducir el fichero sin extension.
GOTO END

:ERROR3
ECHO No se ha encontrado el directorio 'C:\Dev-Cpp' !
ECHO Por favor, especifique el directorio de instalacion del entorno Dev-C++
ECHO como segundo parametro del .bat
GOTO END

:ERROR4
ECHO El directorio '%2\bin' especificado no existe!
GOTO END

:ENDSUCC
ECHO.
ECHO Operacion finalizada con exito!

:END
```

Dev-C++ Instalación y primeros pasos.

MakeDevPak.bat

```
@ECHO OFF

@cls
@COLOR 0F

ECHO.
ECHO -----
ECHO      Para utilizar este .bat se necesitan las utilidades
ECHO      TAR.EXE y BZIP2.EXE incluidas en el devpak "Tools Devpack"
ECHO      disponible a traves de la herramienta de actualizacion
ECHO      automatica del Dev-C++
ECHO -----
ECHO.
@PAUSE
ECHO.
@CLS

IF "%1" == "" GOTO ERROR1
IF EXIST .\%1.DevPak GOTO ERROR0
IF NOT EXIST .\%1\NUL GOTO ERROR2
IF NOT EXIST C:\DEV-Cpp\bin\NUL GOTO NOTEXISTDEFAULTPATH

@SET PATH = %PATH%;C:\DEV-Cpp\bin\

:EXEC
ECHO.
ECHO Empaquetando...
ECHO.
@C:\DEV-CPP\BIN\TAR -cvf %1.tar %1
ECHO.
ECHO Comprimiendo...
ECHO.
@C:\DEV-CPP\BIN\BZIP2 -z %1.tar
@REN %1.tar.bz2 %1.DevPak > NUL
GOTO ENDSUCC

:NOTEXISTDEFAULTPATH
IF "%2" == "" GOTO ERROR3
IF NOT EXIST %2\bin\NUL GOTO ERROR4
@SET PATH = %PATH%;%2
GOTO EXEC

:ERROR0
ECHO El fichero '%1.DevPack' ya existe en el directorio.
ECHO Por motivos de seguridad no se sobrescribira, asi
ECHO que debera borrarlo manualmente.
GOTO END

:ERROR1
ECHO USO: Make_DevPak.bat nombredevpak [Directorio_instalacion_DevC++]
ECHO (El nombre del devpak sin extension)
GOTO END

:ERROR2
ECHO No se encuentra el directorio './%1!'
GOTO END

:ERROR3
ECHO No se ha encontrado el directorio 'C:\Dev-Cpp' !
ECHO Por favor, especifique el directorio de instalacion del entorno Dev-C++
ECHO como segundo parametro del .bat
GOTO END

:ERROR4
ECHO El directorio '%2\bin' especificado no existe!
GOTO END

:ENDSUCC
ECHO.
ECHO Operacion finalizada con exito!

:END
```

Dev-C++ Instalación y primeros pasos.

Bibliografía y temas de copyleft.

Bueno de momento esto es todo. Aquí hay unos cuantos links a algunas páginas interesantes acerca del Dev-C++ y el compilador, que son básicamente en las que me he basado para aprender a utilizar un poco el Dev-C++ y hacer este documento. Todas están en inglés -esa es una de las razones por las que me animé a hacer esto. ☺

- Página de Bloodshed Software (creadores del Dev-C++)
<http://www.bloodshed.net/>
- Página oficial del Dev-C++
<http://www.bloodshed.net/dev/devcpp.html>
- Página oficial del compilador MinGW
<http://www.mingw.org/>
- Página del manual del compilador GCC (del cual está basado el MinGW)
<http://www.fsf.org/software/gcc/onlinedocs/>
- Página del código de Dev-C++ (Desarrollado en Delphi, y si, es paradójico ☺)
Incluye foros y lista de correo.
<http://sourceforge.net/projects/dev-cpp/>
- FAQ oficial del Dev-C++
<http://www.bloodshed.net/dev/faq.html>
- Aditsu's Unofficial Dev-cpp FAQ
<http://www14.brinkster.com/aditsu/dev-cpp-faq.html>

Respecto a los derechos de autor, puedes distribuir este documento siempre que no cambies sus contenidos, se distribuya sin ánimo de lucro y mantengas los datos del autor. Si quieres lo puedes poner en una página web, sólo te pido que me enviaras un correo para avisarme. Es simplemente por saber hasta donde puede llegar este documento, y además, dado que no tengo página web, si saco nuevas versiones es la única manera que tengo para poder avisar de los cambios.

Comentarios, críticas –constructivas o destructivas- correcciones o muestras de apoyo (de desánimo también, pero menos ;)) a
zheo_@hotmail.com

Última modificación 30 Diciembre 2003