CFGM SISTEMAS MICROINFORMÁTICOS Y REDES
SISTEMAS OPERATIVOS MONOPUESTO
TEMA 3 - PRÁCTICA 01

Conceptos

CodeBlocks es un entorno de desarrollo libre (GPL: General Public License) para la creación de aplicaciones en lenguaje C y C++ que utiliza el compilador Mingw/GCC. Tanto CodeBlocks como Mingw/GCC son libres y pueden descargarse de forma gratuita en la dirección *http://www.codeblocks.org/downloads/26*. Para descargar ambas herramientas simultáneamente, debe seleccionar la opción de descarga *codeblocks-17.12mingw-setup.exe*. Desde aquí, puede instalar fácilmente el entorno y el compilador conjuntamente.

1. Pasos en la creación de un programa:

Edición. La edición no es más que escribir el código programa propiamente dicho en lenguaje C. Una vez escrito el código, éste debe ser almacenado en un fichero con extensión ".c". Existen otro tipo de ficheros fuente que se denominan ficheros de cabecera (*header*) y cuya extensión es ".h".

Compilación y Enlazado. En general, se suele denominar compilación al conjunto de procesos que transforma el código fuente escrito en C en un fichero ejecutable (".exe"). La compilación se divide en tres etapas: la *precompilación* (en la que se realiza el análisis sintáctico y se prepara el código para ser compilado), la *compilación* propiamente dicha (en la que se genera un código intermedio llamado código objeto) y el *enlazado o linkado* (donde se enlaza el código objeto con las librerías estándar y externas para formar el ejecutable). Si no se producen errores de compilación, como resultado obtenemos un programa con el mismo nombre que el código fuente original pero con extensión "*.exe*". Todo este proceso es ilustrado en la Figura 1.

Ejecución. Una vez compilado el programa, se ejecuta haciendo doble clic sobre el fichero ejecutable generado o directamente desde el entorno de trabajo. En esta fase también se pueden producir errores por mal funcionamiento del programa, denominados errores de ejecución. Estos errores son más difíciles de corregir, ya que el compilador no indica qué error hay ni dónde se produce, siendo necesario en la mayoría de los casos depurar el programa.

Depuración. La mayoría de los entornos de programación permiten depurar programas realizando una ejecución paso a paso y consultar los valores que van tomando las variables, de manera que podemos hacer una traza para detectar dónde y por qué se producen los errores y corregir los mismos.

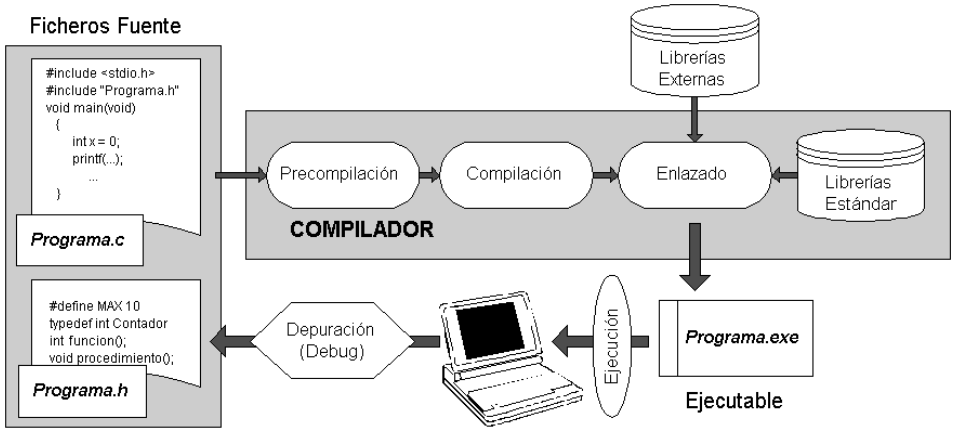

Figura 1: Pasos en la creación de un programa en C

2. El entorno CodeBlocks

El entorno de desarrollo CodeBlocks integra, entre otras, las herramientas:

- Editor orientado al lenguaje C y C++ (resaltando palabras claves, texto autocompletado, etc)
- Compilador C y C++.
- Depurador visual (Debugger), que permite visualizar el contenido de variables.
- Visor de proyectos y ficheros.
- Herramientas complementarias.

Como se muestra en la Figura 2, CodeBlocks presenta el aspecto de una aplicación común Windows, con la barra de menús, la barra de botones y varios paneles de acción. El panel principal es el de edición, donde se escribe el código C de los diferentes programas. A la izquierda encontramos el panel de proyecto para acceder a cada una de las partes del programa. Por último, en la parte inferior tenemos diferentes paneles en los que se presenta toda la información sobre la compilación, errores y depuración del programa.

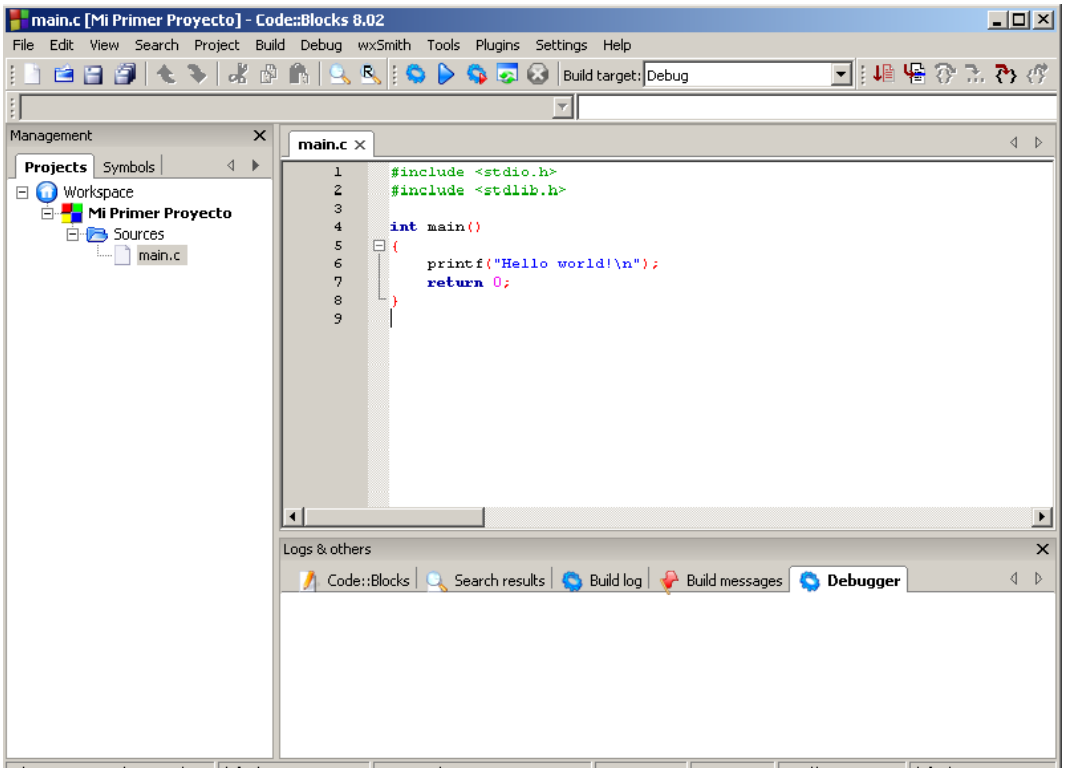

Figura 2: Vista simple de CodeBlocks

3. Creación de un proyecto y edición del código

En el entorno de desarrollo *CodeBlocks* para poder crear un ejecutable a partir de un programa fuente en C es necesario primero crear un proyecto. En esta asignatura se crearán proyectos de consola que son aquellos que se ejecutan en una ventana de símbolo del sistema. Cuando se crea un proyecto, *CodeBlocks* genera un fichero con extensión "*.cbp*" que define el proyecto y otros ficheros que en principio se obviarán.

Para crear un nuevo proyecto, simplemente accedemos a *File*□*New Project* en la barra de menús. A continuación se selecciona *Console Application* como se muestra en la Figura 3. Al hacer clic en el botón *Go* aparece una ventana en la que debe seleccionar el lenguaje de programación (C/C++), y en las sucesivas ventanas debe indicar el nombre del proyecto, el directorio donde se va a guardar el proyecto y el tipo de configuración (Debug/Release).

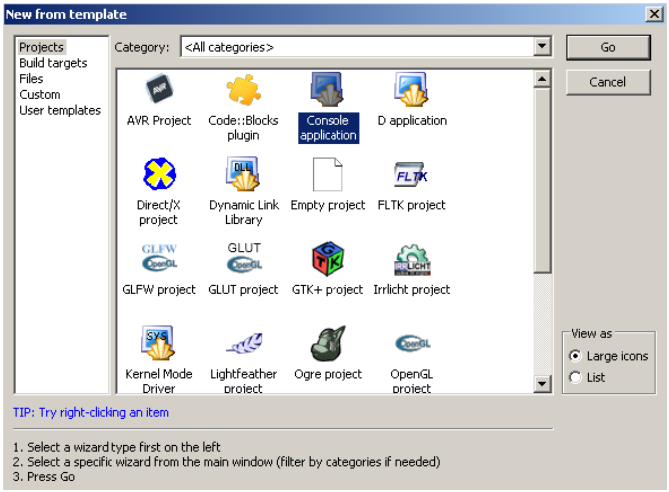

Figura 3: Creación de un proyecto

De esta manera se crea un proyecto con un fichero fuente llamado *main.c* que contiene el siguiente código:

```
#include <stdio.h>
#include <stdlib.h>

int main(){
	printf("Hello world!\n");
	return 0;
}
```

Si se desea crear un proyecto vacío, es decir, sin incluir el fichero fuente *main.c* anterior, simplemente hay que acceder a *Project→New project* en la barra de menús y en la ventana de la Figura 3 seleccionar *Empty project*. Una vez creado el proyecto debe crear un fichero fuente con la opción *File→New→ Empty file* de la barra de menús. Una vez que indique el directorio donde desea guardar el fichero y el nombre del fichero con extensión "*.c*" deberá incluir el archivo en el proyecto activo pulsando con el botón derecho del ratón encima del nombre del proyecto y seleccionando *Add files...*

4. Compilación y enlazado

Una vez que el programa ha sido escrito, podemos compilarlo y "*linkarlo*" para generar el ejecutable. Para ello simplemente basta con acceder al menú *Build→Build*, o pulsar *Ctrl+F9*, o pulsar el botón *Build* en la barra de botones como muestra la Figura 4.

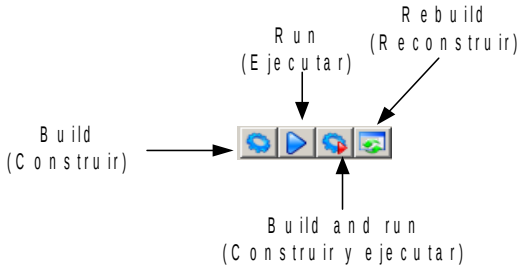

Figura 4: Botones de Compilación, enlazado y ejecución

Si el programa presenta errores, éstos son mostrados en un panel inferior llamado Build log, no se genera el ejecutable y automáticamente se resalta el error en el panel de edición. Si el programa no presenta errores de compilación ni enlazado, se genera un fichero ejecutable con el mismo nombre que el fichero fuente y en la carpeta del proyecto bin/Debug o bin/Release dependiendo del tipo de configuración que hayáis seleccionado.

5. Ejecución

El programa puede ejecutarse desde el propio entorno mediante la opción *Build*→*Run*, o pulsando *Ctrl+F10*, o haciendo clic sobre el botón *Ejecutar* de la barra de botones mostrada en la Figura 4. La salida del programa es mostrada en una ventana MS-DOS independiente a la que podemos acceder mediante la barra de inicio de Windows.

También es posible ejecutar el programa desde fuera del entorno como una aplicación más instalada en el sistema.

6. Depuración

Como se ha comentado anteriormente, un programa no puede ser ejecutado hasta que todos los errores de compilación hayan sido corregidos, aunque esto no garantiza el buen funcionamiento del mismo. Cuando un programa funciona (se ejecuta) pero no realiza su cometido de manera correcta, decimos que se producen errores de ejecución. Estos errores no son detectados por el compilador, por lo que es necesario ejecutar el programa paso a paso para realizar una traza del mismo y localizar el error. A esta fase se le denomina depuración (*debug*). *CodeBlocks* incorpora herramientas de depuración a las que podemos acceder mediante el menú *Debug*.

Breakpoints (Puntos de ruptura)
Antes de arrancar el modo depuración, es necesario establecer previamente al menos un punto de ruptura (*breakpoint*) en alguna línea del programa para que el depurador se detenga al llegar a él. Para ello, se debe colocar el cursor en la línea en la que se quiera establecer el breakpoint y simplemente acceder al menú *Debug*→*Toggle breakpoint* o pulsar *F5*. En ese momento aparece al principio de la línea un punto de color rojo, lo que indica que el punto de ruptura se ha fijado. Para deshacerlo, se repite la misma acción.

Ejecución paso a paso
Una vez puesto un punto de interrupción, podemos ejecutar el programa en modo depuración, línea a línea, para comprobar su correcto funcionamiento. Para arrancar el modo depuración se accede en el menú a *Debug*→*Start* o se pulsa la tecla *F8*. Cuando la depuración ha empezado, al principio de la línea de ejecución aparece la punta de una flecha de color amarilla.

Cuando se comienza una ejecución en modo depuración, aparecen nuevas opciones en el menú *Debug*. Estas opciones son:

- *Next line (F7):* ejecuta la siguiente línea de código. Si la siguiente línea es una función, la ejecuta sin entrar en ella.
- *Step into (Shift + F7):* ejecuta la siguiente línea de código. Si esa línea es una llamada a otra función, el programa entrará en esa función.
- *Step out (Shift + Ctrl + F7):* Sale de la function actual.
- *Continue (Ctrl + F7):* Salta hasta el siguiente punto de ruptura.
- *Run to cursor (F4)*: Ejecuta el programa y se detiene donde esté el cursor en ese momento.
- *Stop debugger:* Detiene la ejecución en modo depuración.
- *Edit watches*: Nos permite añadir y eliminar variables de la ventana de observación.

Ventana de Observación
Para ver los valores de las variables, podemos utilizar la ventana *watches* que está junto al panel de proyecto. Esta ventana se abre pulsando en la pestaña watches. En esta ventana sólo aparecerán aquellas variables que hayamos añadido mediante la opción *Edit watches*. Además en esta ventana se pueden añadir, editar y eliminar variables pulsando el botón derecho del ratón.

Experimentos

E1. Cree un programa que muestre por pantalla el mensaje "*Mi primer programa en C*". Pruebe las diferentes formas de compilar y ejecutar el mismo. Introduzca errores intencionadamente para comprobar el funcionamiento del compilador.

E2. Complete el siguiente programa para que pida dos números por teclado y muestre el valor de éstos junto a la suma por pantalla.

```
#include <stdio.h>
void main(void){
          int a, b, suma;
          printf ("\nIntroduzca dos numeros a y b: ");
          scanf ("%d%d", ???????, ??????);
          printf ("\n a vale: ?????? y b vale ??????", a, b);
          suma = ??????;
          printf ("\nLa suma de a y b vale: %d\n", suma);
}
```

E3. Amplíe el programa del experimento anterior para que tome dos números enteros por teclado y muestre los valores introducidos junto a la suma, la resta, la multiplicación y la división de los mismos.

E4. El siguiente programa tiene varios errores, tanto de compilación como de ejecución. Escríbalo tal y como está. Luego corrija los errores para que funcione.

```
void main(void){
     char car;
     int i;
     float f;
     scanf("Introduzca un caracter: %c", &car);
     scanf("Introduzca un entero: %d", &i)
     scanf("Introduzca un real: %f", &f);
     printf("El caracter es:", car);
     printf("El entero es:" i);
     prin("El real es:", f);
 }
```

Ejercicios

EJ1 (20 min). El siguiente programa declara dos variables, les asigna un valor diferente a cada una y las muestra por pantalla. Después intercambia los valores de las variables y las vuelve a mostrar. Escriba, compile y ejecute el programa. ¿Funciona correctamente? Ejecútelo en modo depuración y corríjalo.

```
void main(void){
          int x = 10, y = 20;
          printf("x vale %d e y vale %d\n", x, y);
          x = y;
          y = x;
          printf("Después del intercambio, x vale %d e y vale %d\n", x, y);
}
```

EJ2 (20 min). El siguiente programa calcula el área de un rectángulo cuyos datos se leen desde el teclado. ¿Qué errores encuentra en el mismo?

```
include <stdio.h>
void main (void){
       int base, altura;
       printf ("\nCálculo del área de un rectángulo");
       printf ("\nPor favor, introduzca base y altura: ");
       /* Leer base y altura */
       scan ("%c%c", &base, &altura);
       /* Calcular e imprimir área
       area = base*altura;
       printf ("\n\nEl área del rectángulo es: %d\n", area);
}
```

EJ3 (20 min). Escriba un programa que tome por teclado los coeficientes (*a*, *b* y *c*) de una ecuación de segundo grado y la resuelva.

$$ax^2+bx+c \implies x=\frac{-b\pm\sqrt{b^2-4ac}}{2a}$$