USO Y CONFIGURACIÓN DEL COMPILADOR TURBO C++

Bruno López Takeyas
Instituto Tecnológico de Nuevo Laredo
Reforma Sur 2007, C.P. 88250, Nuevo Laredo, Tamps. México
http://www.itnuevolaredo.edu.mx/takeyas
E-mail: takeyas@itnuevolaredo.edu.mx

Resumen: *Para codificar mejores programas es recomendable utilizar y aprovechar al máximo las herramientas que proporciona el editor y el compilador. Un aspecto muy importante es la depuración de los programas para identificar fácilmente los errores o las áreas sujetas a optimización. Por ello es indispensable saber ejecutar los programas paso a paso, por bloques, poder observar los valores de variables en tiempo real, monitorear variables e incluso alterar sus valores en tiempo de ejecución para realizar las pruebas a dichos programas. A continuación se muestran las herramientas que presenta Turbo C++ para esto.*

1.- Depuración de errores

A continuación se presentan algunas opciones que nos proporciona el compilador para ejecutar programas paso a paso y monitorear los valor de variables para depurar posibles errores en programas.

Trace into (ejecución línea por línea) (F7)

La ventana de Run (ejecución) (Fig. 1.1) contiene varias herramientas para ejecutar programas paso a paso. La herramienta "Trace into" ejecuta paso a paso un programa, es decir, línea por línea de código. Para lograr esto, sólo basta oprimir la tecla F7 cada vez que se requiera ejecutar una línea de código.

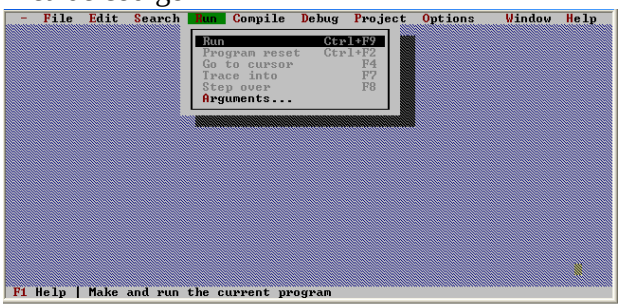

Fig. 1.1.- Ventana de ejecución de programas.

Step over (ejecución por subrutinas o funciones) (F8)

Esta herramienta también ejecuta paso a paso un programa, pero a diferencia del "Trace into" no lo hace línea por línea, sino por subrutinas o funciones; es decir, si en el código se encuentra el llamado a una subrutina, la ejecuta completamente sin llamarla línea por línea. Para lograr esto sólo basta oprimir la tecla F8 cada vez que se requiera ejecutar la siguiente subrutina o función.

Program reset (interrumpir la ejecución paso a paso de un programa) (Ctrl-F2)

Se utiliza esta opción cuando se desea interrumpir la ejecución y depuración de un programa. Basta oprimir la combinación de teclas Ctrl-F2.

Inspect (inspeccionar variables) (Alt-F4)

Turbo C++ muestra varias herramientas para monitorear variables (Fig. 1.2). La opción "Inspect" permite inspeccionar variables, es decir, monitorear su valor, su tipo y la dirección donde se aloja en memoria (Fig. 1.3).

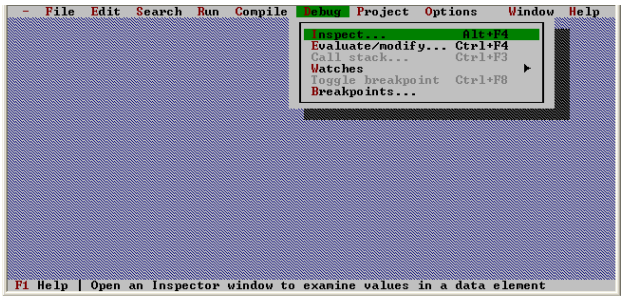

Fig. 1.2.- Depurar.

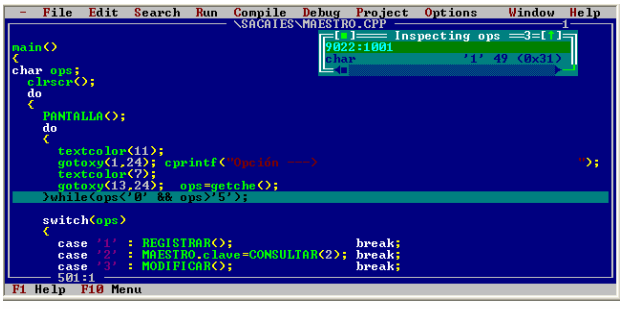

Fig. 1.3.- Inspeccionar variables

Evaluate/Modify (evaluar y modificar variables) (Ctrl-F4)

Con esta herramienta se puede monitorear y alterar el valor de variables (Fig. 1.4) y es útil cuando se desea modificar el valor de una variable en tiempo de ejecución.

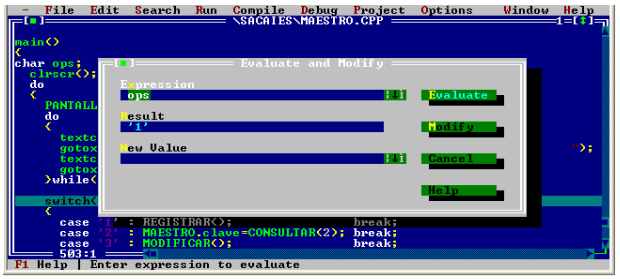

Fig 1.4.- Evaluar/Modificar variables.

Add watch (agregar vista para monitorear variables) (Ctrl-F7)

Con esta herramienta se puede monitorear constantemente las variables (Fig. 1.5) y es útil cuando se desea observar constantemente las variables en tiempo de ejecución. Se puede agregar mas de una variable en una ventana (Fig. 1.6).

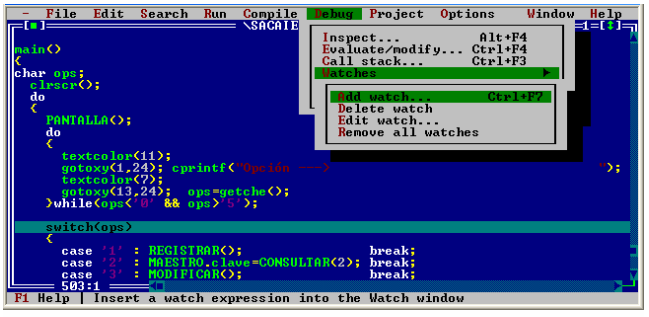

Fig. 1.5.- Vistas de variables.

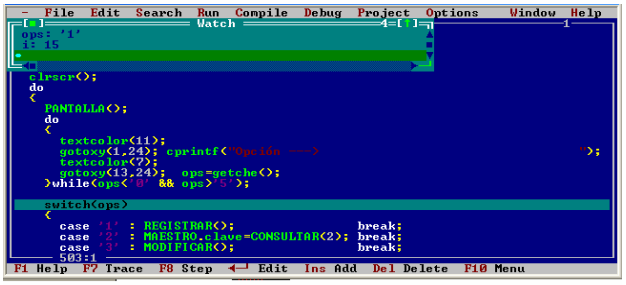

Fig. 1.6.- Ventana de monitoreo de algunas variables.

Toggle breakpoint (agregar/eliminar punto de ruptura) (Ctrl-F8)

Esta opción permite establecer o eliminar puntos de ruptura, es decir, cuando se desea ejecutar un programa paso a paso a partir de alguna línea de código, se establece un breakpoint (se marca la línea de código con una franja roja), para después ejecutar el programa normalmente y cuando vaya a ejecutar la línea marcada, se detiene la ejecución a esperar un "Trace into" o un "Step over" (Fig. 1.7).

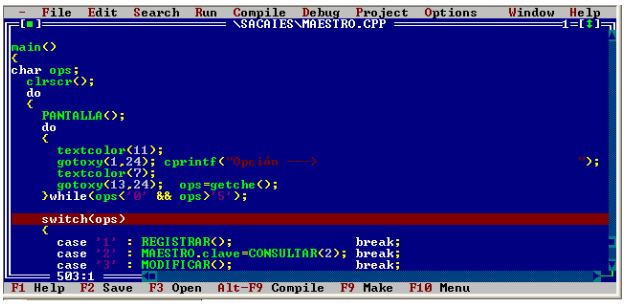

Fig. 1.7. Breakpoint.

2.- CONFIGURACIÓN DEL COMPILADOR

Es importante conocer las opciones de configuración del compilador para optimizar el rendimiento de nuestros programas. En esta sección se verá brevemente las diversas formas de configuración.

Modelos de memoria

La Fig. 2.1 muestra la ventana de configuración de los modelos de memoria del compilador.

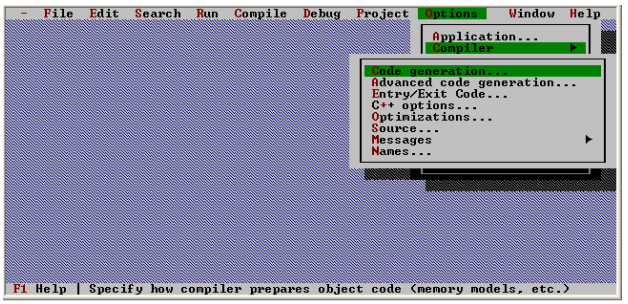

Fig. 2.1.- Configuración de los modelos de memoria.

Existen los siguientes modelos de memoria (Fig. 2.2):

Tiny: Este modelo de memoria usa los registros CS, DS, ES y SS en la misma dirección. Esto significa que se cuenta con 64K de memoria para código, datos y pila.

Small: Se usa este modelo de memoria para aplicaciones de tamaño regular. Los segmentos de código y de datos son diferentes y no se empalman, ya que se cuenta con 64K de memoria para el código y 64K para datos y la pila.

Medium: Este modelo se utiliza cuando se tiene un programa de gran tamaño que no maneja

muchos datos en memoria. Los apuntadores lejanos se utilizan para el código pero no para los datos, es decir, se tienen 64K para datos y pila, pero el código puede ocupar hasta 1MB de memoria.

Compact: Este modelo es lo contrario del modelo Médium y úselo cuando se tenga poco código en el programa pero que maneje una gran cantidad de memoria de datos. En este caso se usan apuntadores lejanos para los datos pero no para el código, ya que éste se limita a 64K mientras que los datos pueden ocupar hasta 1MB de memoria.

Large: Use este modelo solamente para aplicaciones muy grandes. Se usan apuntadores lejanos tanto para datos como para código, otorgándoles 1MB de memoria a cada uno.

Huge: También se usa este modelo para aplicaciones muy grandes, sólo que permite varios segmentos de datos de 64K cada uno, hasta 1MB de código y 64K para la pila.

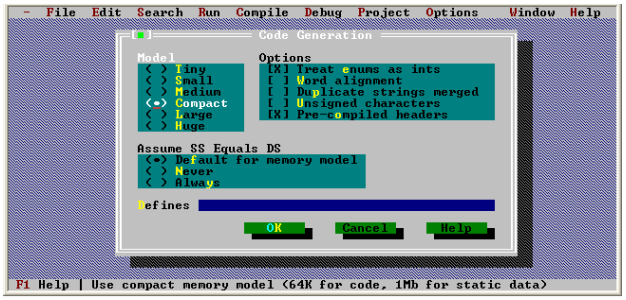

Fig 2.2. - Modelos de memoria

Directorios

La Fig. 2.3 muestra la ventana donde se establecen las rutas de búsqueda de las utilerías del compilador, además se define el subdirectorio donde se grabarán los programas objeto y los ejecutables producto de la ejecución de los programas.

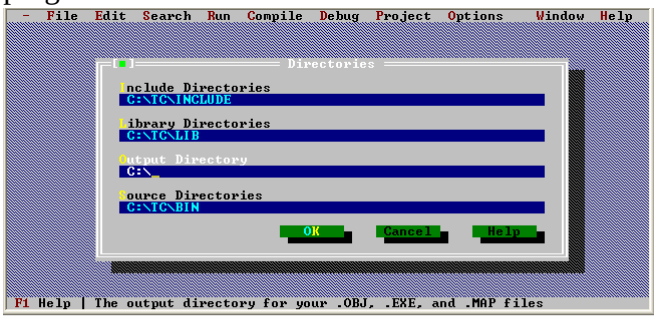

Fig. 2.3.- Directorios.

Guardar configuración

Una vez que se modifica la configuración del compilador, es necesaria grabarla para mantenerla en compilaciones futuras (Fig. 2.4

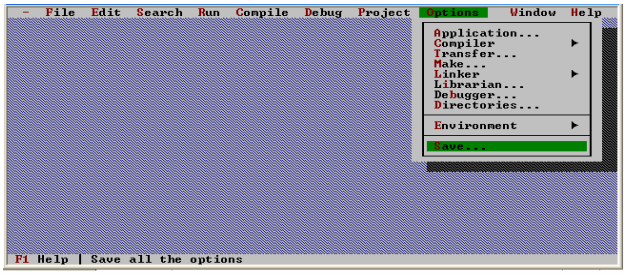

Fig. 2.4. Grabar la configuración del compilador.