

Desarrollar una aplicación con gráficas en DevC++ usando WinBGIm

Introducción

La librería WinBGIm ha sido creada originalmente por Konstantin Knizhnik's en el proyecto winbgi shareware y posteriormente modificada por Mark Richardson y Michael Main; posterior a esto, ya que la librería no tuvo más soporte desde el año 2004, Oswaldo Romero de la Universidad Distrital en Colombia hizo modificaciones menores y añadió unas funciones para la manipulación de imágenes BMP. Esta librería es una versión que emula la creada por la compañía Borland BGI (Borland Graphics Interface) la cual tiene la capacidad para crear gráficos en pantalla en aplicaciones de 16 bits sobre MS-DOS® y Windows®.

Esta librería facilita funciones que permiten escribir en modo gráfico en la pantalla (en realidad en una ventana de Windows) punto a punto, figuras geométricas, copiar y modificar el contenido de un trozo de la pantalla y manejar el ratón.

La ventaja de esta librería es que nos permite crear aplicaciones gráficas con compiladores GNU tales como el ambiente de desarrollo Dev C++ de la misma manera como se crean en el compilador comercial Borland C++, siendo muy utilizada en su tiempo por la facilidad de su uso.

¿Cómo se crea una aplicación?

En Dev-C++, ejecutamos la aplicación y creamos un nuevo proyecto de Consola, como se muestra en la figura 1.

Como nombre de proyecto, escoja una ubicación en donde se creará una carpeta con el mismo nombre; en esta carpeta se guardarán los archivos de código y por supuesto los archivos correspondientes a la librería.

El siguiente paso es copiar los archivos de la librería para poderlos colocar en el proyecto. Estos archivos son (ver figura 2):

- **winbgim.h** : Que es el encabezado (header) donde se definen los prototipos de las funciones de la librería.
- **winbgim.cpp** : Corresponde a la implementación de las funciones.

Luego debemos entrar a las opciones del proyecto (ver figura 3). Se puede hacer de 2 formas:

- presionando la combinación de teclas CTRL+H.
- ir con el puntero del ratón a la sección del Proyecto (Tabulador), y con click derecho seleccionar en el menú que aparece "Opciones del proyecto".

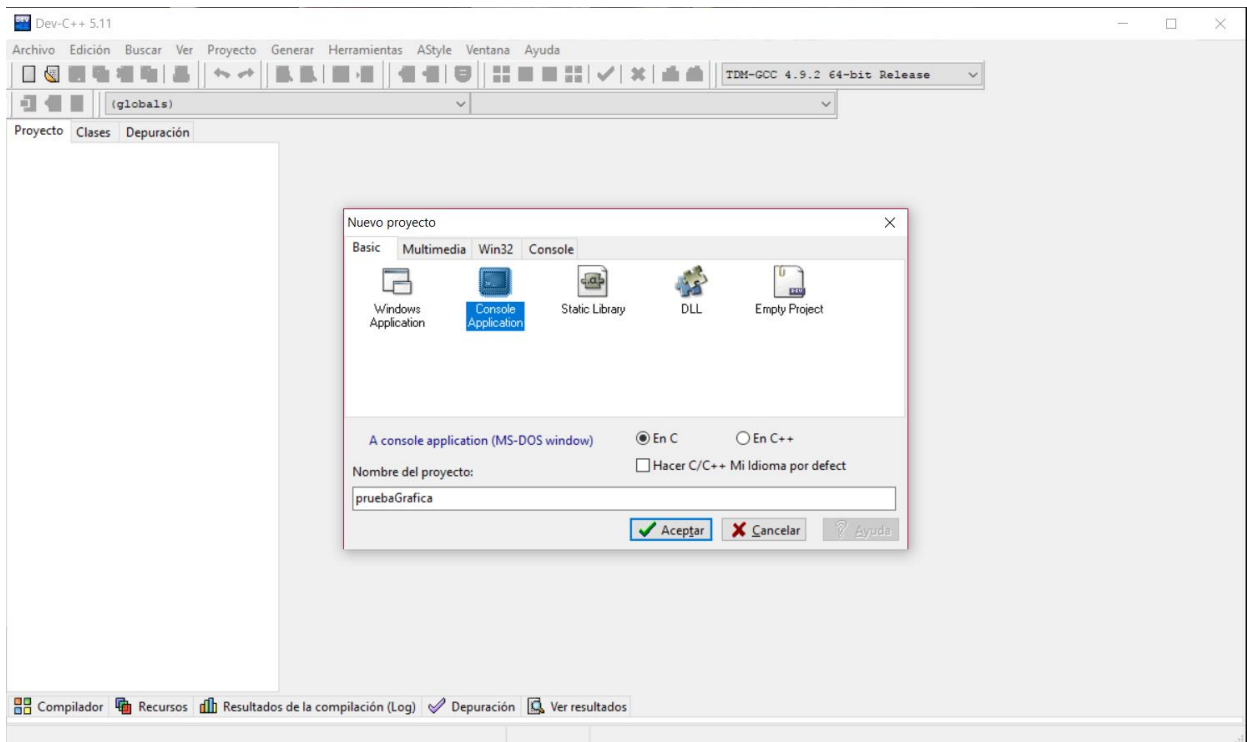


Figura 1. Nuevo proyecto.

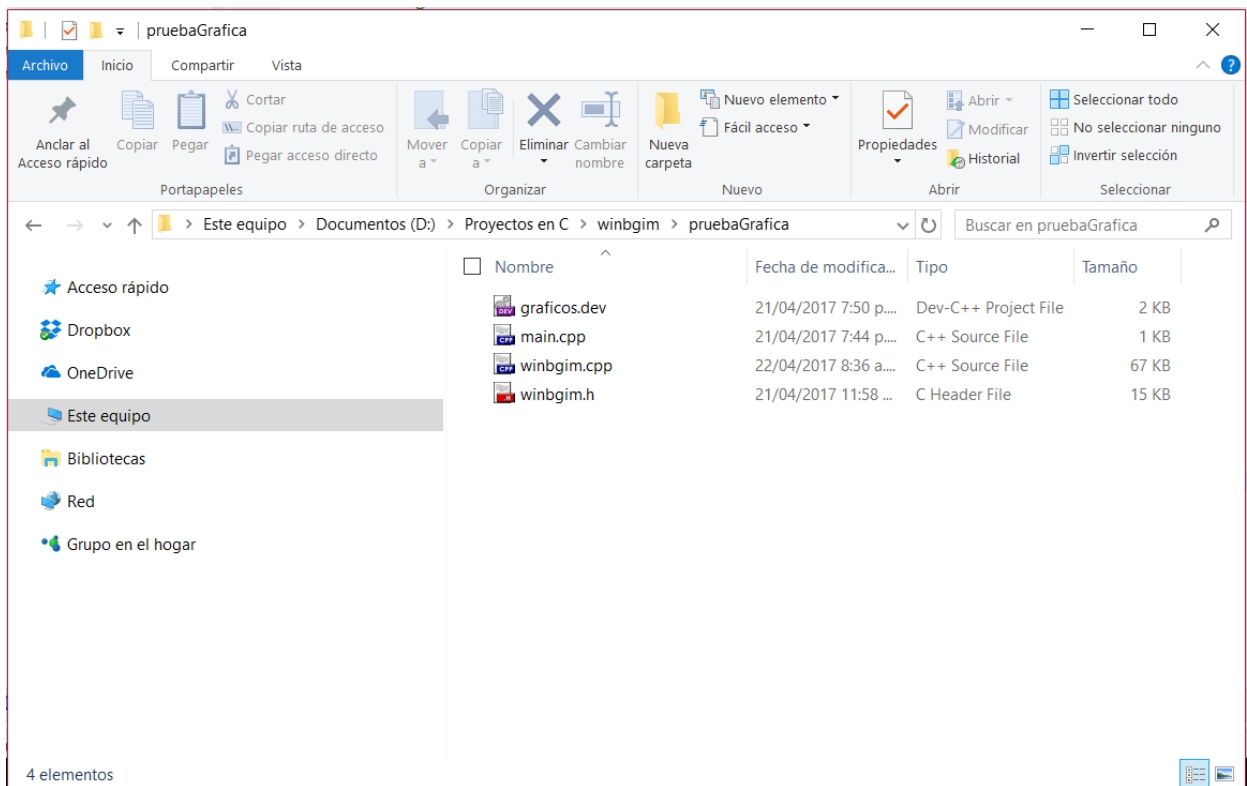


Figura 2. Ejemplo de ubicación de los archivos del proyecto.

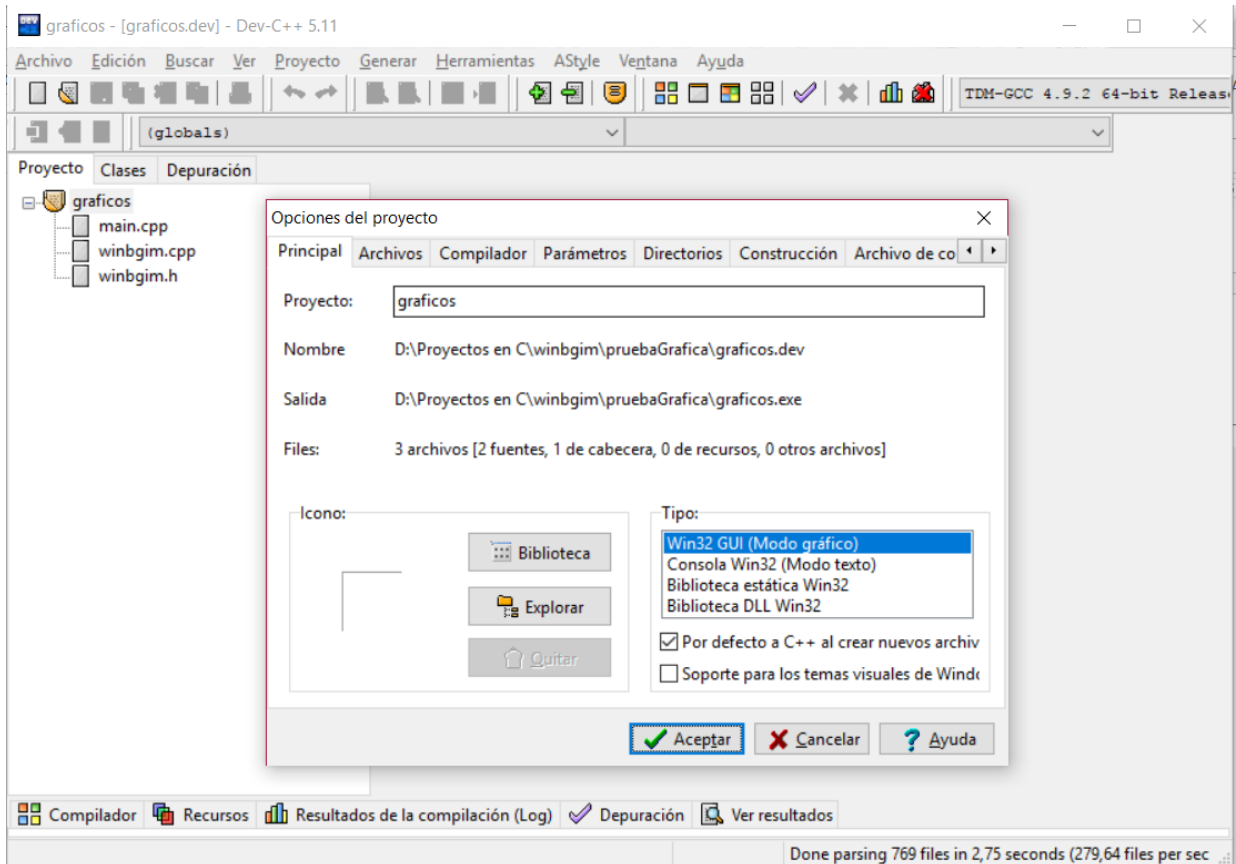


Figura 3. Opciones del proyecto

En este cuadro debemos seleccionar en la sección TIPO, la opción “Win32 GUI (Modo gráfico)”. Luego aceptamos.

¡Listo! Ya podemos empezar con nuestro código.

En el código

Lo primero que tenemos que hacer es vincular los archivos de la librería en el proyecto. Para esto con el puntero del ratón vamos a la sección del Proyecto (Tabulador), y con click derecho seleccionamos en el menú la opción “Agregar al proyecto” como indica la figura 4. Aparecerá un cuadro de diálogo en donde seleccionamos los archivos y aceptamos (ver figura 5).

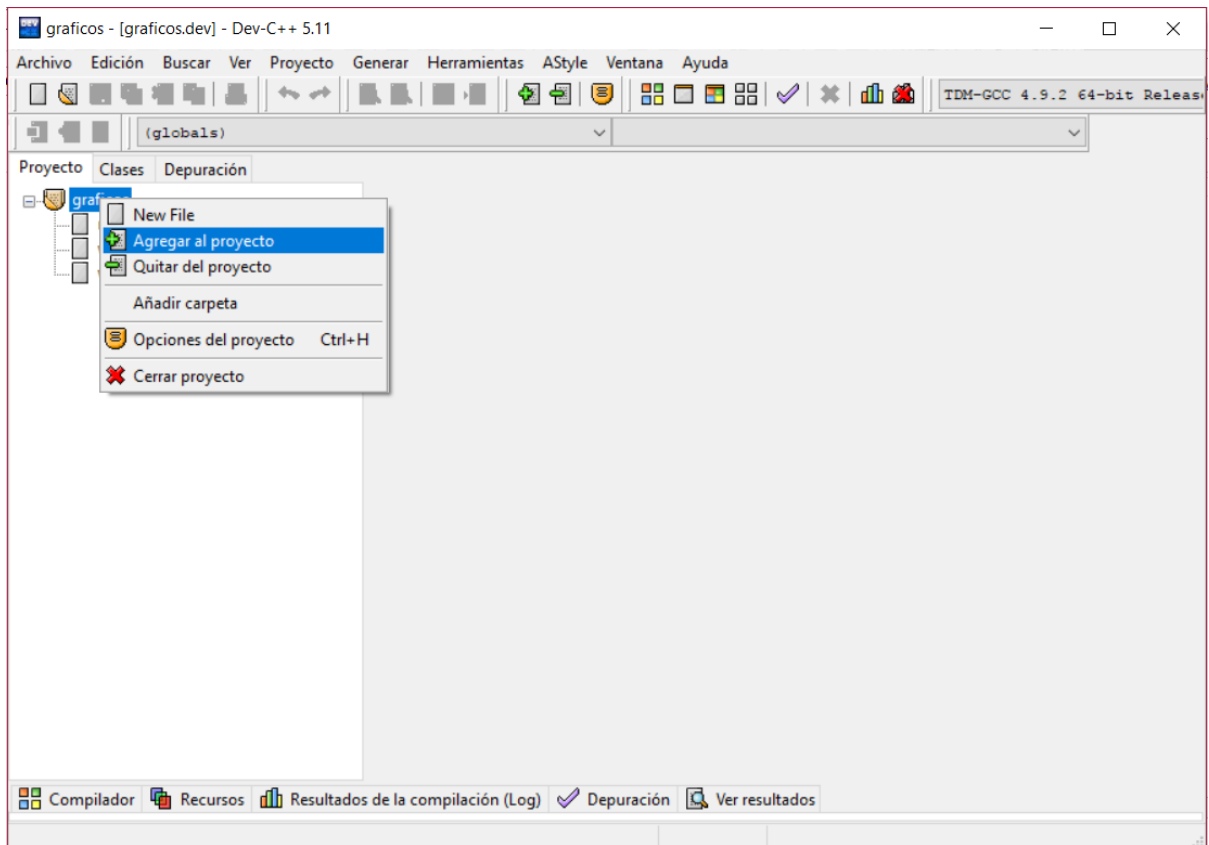


Figura 4. Opción para agregar archivos al proyecto

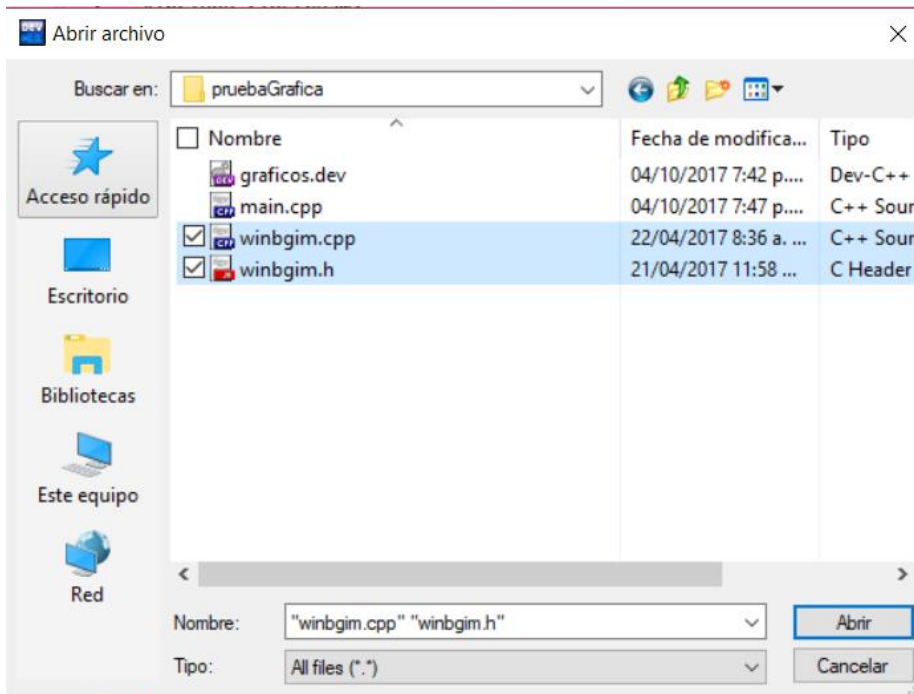
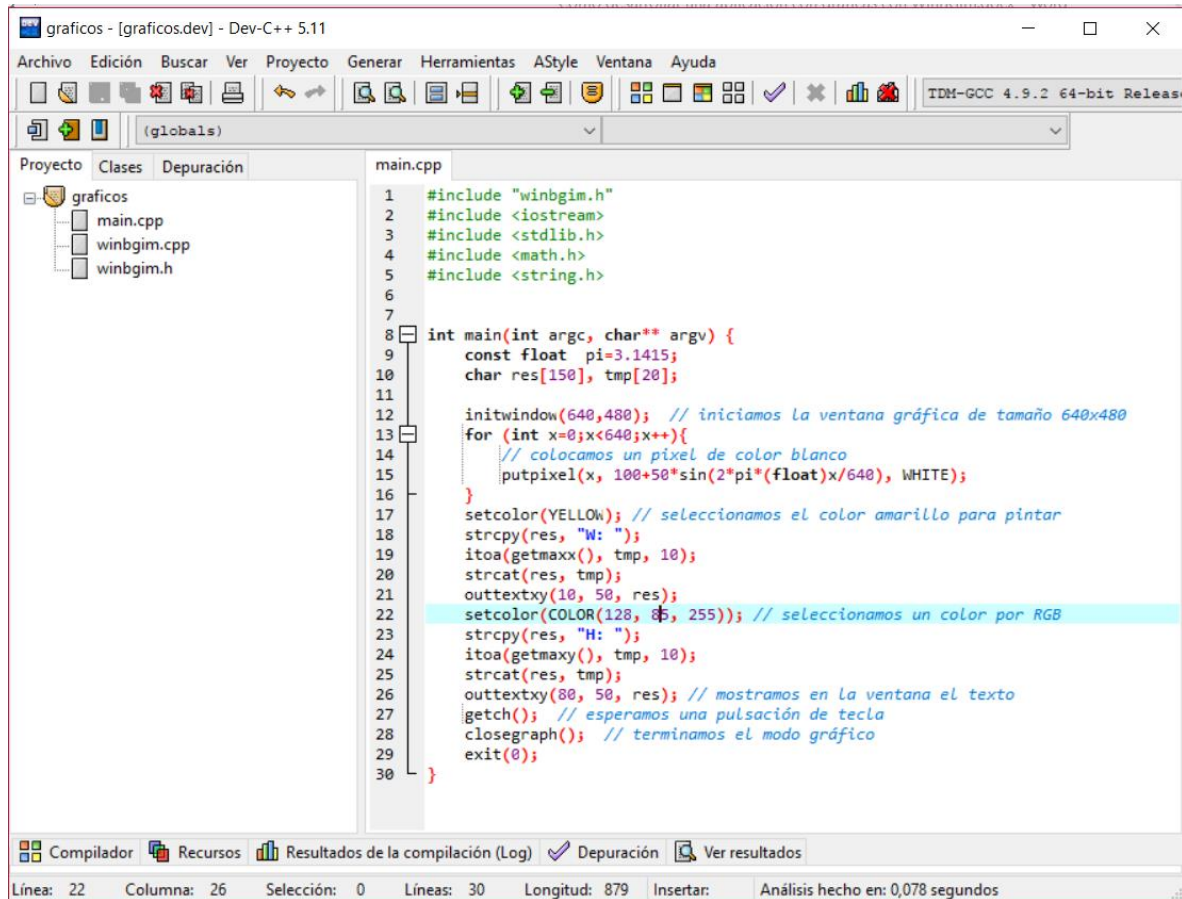


Figura 5. Cuadro de diálogo para seleccionar archivos a vincular.

Luego de vincular nuestros archivos de la librería, seleccionamos el archivo en donde encontraremos la función `main()`, o en donde vayamos a usar las funciones gráficas, e incluimos el archivo de encabezado `winbgim.h` con la directiva `#include "winbgim.h"`, como se ve en la figura 6.

Luego de esto, es usar las funciones que provee la librería de acuerdo a nuestra imaginación.



```
1 #include "winbgim.h"
2 #include <iostream>
3 #include <stdlib.h>
4 #include <math.h>
5 #include <string.h>
6
7
8 int main(int argc, char** argv) {
9     const float pi=3.1415;
10    char res[150], tmp[20];
11
12    initwindow(640,480); // iniciamos la ventana gráfica de tamaño 640x480
13    for (int x=0;x<640;x++){
14        // colocamos un pixel de color blanco
15        putpixel(x, 100+50*sin(2*pi*(float)x/640), WHITE);
16    }
17    setcolor(YELLOW); // seleccionamos el color amarillo para pintar
18    strcpy(res, "W: ");
19    itoa(getmaxx(), tmp, 10);
20    strcat(res, tmp);
21    outtextxy(10, 50, res);
22    setcolor(COLOR(128, 85, 255)); // seleccionamos un color por RGB
23    strcpy(res, "H: ");
24    itoa(getmaxy(), tmp, 10);
25    strcat(res, tmp);
26    outtextxy(80, 50, res); // mostramos en la ventana el texto
27    getch(); // esperamos una pulsación de tecla
28    closegraph(); // terminamos el modo gráfico
29    exit(0);
30 }
```

Figura 6. En la primera línea incluimos la librería para usar las funciones en el código.

Algunas rutinas o funciones

Rutinas de inicialización

`void initwindow(int ancho, int alto);`

Esta rutina hay que llamarla antes que cualquier otra de esta biblioteca, ya que inicializa el sistema gráfico y crea una ventana gráfica del tamaño especificado en los parámetros. El sistema de referencia empieza en las coordenadas 0,0, que representa la esquina superior izquierda, por lo que el eje y crece hacia la parte de abajo de la ventana.

void closegraph(void);

Al acabar de trabajar en el modo gráfico, hay que llamar a esta rutina, que libera toda la memoria reservada por el sistema gráfico y devuelve la pantalla al modo en el que estaba antes de inicializar el modo gráfico.

Rutinas de escritura

void cleardevice(void);

Esta rutina borra la pantalla gráfica rellenándola del color de fondo y mueve la posición de dibujo al origen de la pantalla, la posición (0,0).

void line(int x1, int y1, int x2, int y2);

Esta rutina dibuja una línea entre los puntos (x1, y1) y (x2, y2), con el color, el estilo de línea y el ancho actuales. No mueve la posición de escritura.

void lineto (int x, int y)

Traza una línea desde la posición actual de cursor hasta x, y.

void circle (int x, int y, int radius);

Esta rutina dibuja un círculo en el color de dibujo actual cuyo centro está en la posición (x, y) y de radio r.

Otras rutinas de dibujo

Estas otras rutinas dibujan diversas figuras geométricas:

void arc (int x, int y, int stangle, int endangle, int radius);

void bar(int izq, int arriba, int derecha, int abajo);

void bar3d (int left, int top, int right, int bottom, int depth, int topflag);

void ellipse (int x, int y, int stangle, int endangle, int xradius, int yradius);

void fillellipse (int x, int y, int xradius, int yradius);

*void fillpoly (int numpoints, int *polypoints);*

void pieslice (int x, int y, int stangle, int endangle, int radius);

void rectangle (int left, int top, int right, int bottom);

void sector (int x, int y, int stangle, int endangle, int xradius, int yradius);

void moveto (int x, int y);

Esta rutina mueve la posición de dibujo a la coordenada (x, y).

void outtext(char cad[]);

Esta rutina escribe una cadena de caracteres en pantalla, usando el tipo de letra, la dirección y el tamaño actuales.

void putpixel(int x, int y, int color);

Esta rutina escribe un punto del color especificado en la posición (x,y).

Rutinas de lectura

int getbkcolor (void); int getcolor (void);

getcolor devuelve el color de dibujo actual. Ése es el color en el que escriben los puntos cuando se dibujan líneas y demás figuras. getbkcolor devuelve el color de fondo actual.

int getch (void);

Esta rutina lee un carácter de la pantalla gráfica sin esperar que se pulse la tecla de nueva línea. Para las teclas extendidas, primero se lee el valor 0 y, en una segunda lectura, se lee el valor correspondiente a cada tecla. Por ejemplo:

```
#define KEY_HOME    71 // tecla Inicio
#define KEY_END      79 // tecla Fin
#define KEY_UP       72 // cursor hacia arriba
#define KEY_LEFT     75 // cursor hacia la izquierda
#define KEY_RIGHT    77 // cursor hacia la derecha
#define KEY_DOWN     80 // cursor hacia abajo
#define KEY_F1       59 // tecla de funcion F1
#define KEY_F2       60 // tecla de funcion F2
```

int getmaxx (void); int getmaxy (void);

Estas rutinas devuelven el valor máximo de x y de y, respectivamente, para la pantalla gráfica.

unsigned getpixel (int x, int y);

Esta función devuelve el color del pixel de la coordenada (x,y).

int getx (void); int gety (void);

Estas rutinas devuelven, respectivamente, la coordenada x y la coordenada y de dibujo.

Rutinas de actualización de parámetros gráficos

void setcolor (int color); void setbkcolor (int color);

Estas rutinas establecen el nuevo color de dibujo y el nuevo color de fondo, respectivamente.

void setfillstyle (int pattern, int color);

Esta rutina establece los nuevos parámetro y color de relleno.

void setlinestyle (int linestyle, unsigned upattern, int thickness);

Esta rutina establece el estilo de dibujo de las líneas.

Rutinas varias

void delay (int miliseg);

Esta rutina hace que el programa se pare durante el número de milisegundos que se pasa como parámetro.

int kbhit (void);

Esta rutina devuelve 0 si no se ha pulsado ninguna tecla desde la última lectura y un valor distinto de 0 en caso contrario.

Rutinas del mouse

int mousex(void)

Retorna la coordenada x del Mouse relativa a la esquina superior izquierda

int mousey(void)

Retorna la coordenada y del Mouse relativa a la esquina superior izquierda

Los colores en Windows BGI

Hay dos maneras de referirse a los colores en WinBGI. La primera es usar la paleta tradicional de 16 colores, numerados del 0 al 15 y que tienen sus nombres propios:

BLACK	BLUE	GREEN	CYAN
RED	MAGENTA	BROWN	LIGHTGRAY
DARKGRAY	LIGHTBLUE	LIGHTGREEN	LIGHTCYAN
LIGHTRED	LIGHTMAGENTA	YELLOW	WHITE

La otra manera es formar un color indicando los tonos de rojo, azul y verde con valores entre 0 y 255. Esto se hace con la macro **COLOR(r, g, b)**, donde los tres parámetros indican el tono de cada color.

Referencias

1. Borland BGI Graphics emulation for the MingW (GCC port) compiler, en: <http://winbgim.codecutter.org/>
2. Borland Graphics Interface (BGI) for Windows, en: <http://www.cs.colorado.edu/~main/cs1300/doc/bgi/bgi.html>