Generación de un Programa Para Realizar Diagramas de Flujo Mediante la Técnica de POO

Ing. Ignacio Raúl Rosas Román
Alumno de la Maestría del CINTEC-IPN.

Desde su aparición, la programación estructurada ha proporcionado a los programadores una metodología de diseño consistente que facilita la elaboración de un programa. Esta metodología se basa en la descomposición de un problema en piezas pequeñas que son más manejables que el problema original. A estas piezas que resultan de la primera división se les denomina "subtareas". El siguiente paso bajo esta metodología es ir descomponiendo cada una de estas subtareas, hasta llegar al punto donde se pueda expresar cada subtarea en comandos del lenguaje de programación que se está utilizando. De lo anterior, se puede observar que la metodología se centra en las acciones que el programa realizará sobre los datos.

Aun cuando los lenguajes de programación estructurada reconocen como objetivo final la reutilización de código, hasta ahora no ha sido posible obtener un programa fácil de actualizarse o modificarse con facilidad.

Por otra parte, la programación orientada a objetos fue uno de los primeros sistemas de programación en que se reconoció que el cambio y la evolución de un programa no sólo son inevitables, sino también deseables. La programación orientada a objetos busca minimizar el impacto del cambio a través de la técnica del encapsulado, la cual se basa en el agrupamiento de procedimientos y datos asociados de un objeto en un solo conjunto.

La idea general de la programación bajo esta metodología se basa en estructurar los programas alrededor de los datos, asociándoles íntimamente los tratamientos que les son específicos, en lugar de hacerlo sobre las funciones, las cuales son modificadas mas frecuentemente.

Desarrollo del programa.

El compilador que se utilizó para generar el programa, es el C++ versión 3.1 de Borland. Este compilador utiliza como plantilla para la construcción de objetos la estructura **class**, dentro de la cual es posible definir datos y funciones. Las funciones de una clase se llaman funciones miembro y son las encargadas de manipular los datos pertenecientes a la clase, así como de definir el acceso que se tiene a ellas.

Para definir las clases que se utilizarán, es necesario analizar primero el problema. Los diagramas de flujo son dibujos bidimensionales, por lo cual todos los dibujos pueden ser definidos dentro de un espacio rectangular. Este espacio rectangular dentro de un sistema de ejes cartesianos, puede ser caracterizado por cuatro variables que definen dos parejas de datos (x, y) distintas dentro del plano. Por esta razón, todos los objetos a dibujar dentro del programa se representarán mediante 5 variables, 4 de ellas están encaminadas a definir el área dentro de la cual se dibujará el objeto y una más para definir el tipo de objeto que se está dibujando. Con esto se logra que los datos que definen los objetos del dibujo sean solamente 5 enteros, en lugar de guardar el mapa de bits asociado a cada objeto. Esta estrategia resulta en un manejo de memoria más eficiente. La declaración de este objeto se encuentra en el archivo FIGURAS.H y se presenta a continuación.

```
class Base
{
 protected:
 int X1, X, Y1, Y, linea;
 public:
 void asigna(int, int, int, int, int);
};
```

Como se observa, la única función miembro dentro de este objeto, es la función *asigna*, la cual tiene como característica ser **pública**, lo cual nos permite utilizar dicha función en caso de que la clase se utilice como bloque de construcción de otro objeto. La tarea que realiza esta función miembro es actualizar las variables que definen al objeto. La importancia de esta clase es fundamental, ya que a partir de ella se definen las clases encargadas de dibujar los objetos en la pantalla y se genera una lista en la cual se almacenan los objetos.

Lo anterior se hace mediante el uso de **clases derivadas**, que tienen la característica de **heredar** las propiedades de la clase a partir de la cual fueron construidas.

En términos generales, la herencia tiene el objetivo de compartir y/o sacar puntos comunes entre los diferentes subconjuntos de problemas. El principio es describir un nuevo objeto sin que se tenga que partir de cero, sino por el contrario, por extensión o especialización de uno o varios objetos que ya existen. A partir de lo anterior, es evidente que las características de herencia de la programación orientada a objetos facilitan la reutilización del código o bien su transición hacia características más complicadas (evolución).

Una clase derivada que se utiliza dentro del programa es la que se describe a continuación:

```
class Elementos: public Base {
 int tipo;
 char *cadena;
 public:
 void entra_datos(int, int, int, int, int, int, char*);
 void sale_datos(int &, int &, int &, int &, int &, int
&, char *&);
 int busca_elemento(int, int);
};
```

La clase llamada "Elementos", tiene como clase básica para su construcción a la clase "Base". Dentro de "Elementos" se definen variables adicionales para describir características que dan la diferencia entre dos objetos similares, como son el ancho de línea o el tipo de carácter a utilizar (variable tipo) y, por otra parte, una dirección que indica donde está la cadena de caracteres asociada a un mensaje. Adicionalmente a los métodos de la clase base, se tienen métodos para actualizar los datos (función entra_datos), leer los datos (función sale_datos) y una función que busca si un punto definido por un par (X, Y) está dentro del área que define un objeto.

Las clases "figuras", también son clases derivadas a partir de la clase base. Esta clase solamente define una función miembro adicional que se encarga de dibujar una figura específica dentro de los límites definidos por los dos puntos de la clase "Base".

Por otra parte, es necesario definir un objeto que permita la interacción del usuario con el programa. Dicha interacción se realizará básicamente a partir de la clase MOUSE, que se encarga de recibir los eventos que provienen de este dispositivo. El uso del programa se hace casi enteramente a través del Ratón, a excepción de los mensajes para el texto. Es por esta razón que aparecerá un mensaje indicando la falta de este dispositivo en caso de no estar instalado en el equipo y el acceso al programa será negado.

La última clase que compone al programa, es la clase "Ambiente", la cual se encarga de realizar todas las tareas asociadas con la edición y manejo de los objetos gráficos definidos anteriormente. Los datos que maneja esta clase son principalmente instancias de la clase "Elementos" y variables que definen características diversas del área de trabajo que se utiliza (monitor). El manejo del monitor utiliza una resolución de 640 X 480 pixeles bajo el modo de video VGA en color o blanco y negro.

Estructura del programa.

El primer procedimiento que se ejecuta es la función "dibuja_pantalla" de la clase ambiente, que es la encargada de comprobar la existencia del Ratón, así como de inicializar el modo de video y dibujar el ambiente de trabajo para el usuario, el cual se puede observar en la **figura 1**.

El siguiente paso consiste en el uso de la función "maneja _icono", que es la encargada de detectar los mensajes provenientes del mouse para verificar si el usuario desea activar alguna de las funciones disponi-

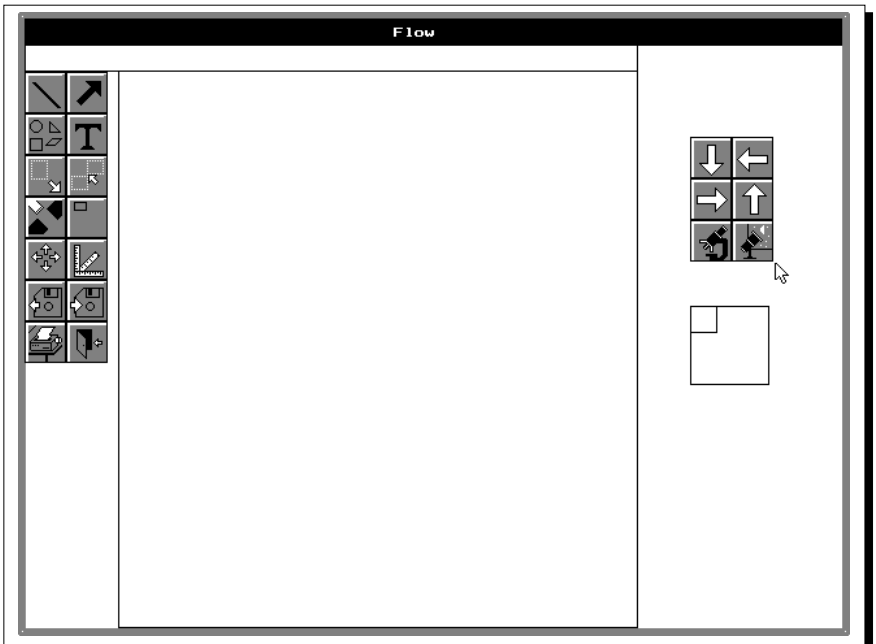

Figura 1. Ambiente de trabajo para el usuario

bles en el programa. Para hacer esto, simplemente se posiciona el apuntador del mouse sobre el ícono de la función deseada y se presiona el botón izquierdo.

Las funciones miembro de la clase ambiente pueden ser agrupadas en tres tipos:

1) Funciones que dibujan los objetos.
2) Funciones que editan los objetos en la pantalla.
3) Funciones que escriben o leen de archivos los diagramas generados en el paquete.

El primer tipo de funciones tiene la característica de presentar varias opciones de dibujo al usuario, las cuales aparecen en el lado derecho de la pantalla (ver **figura 2**). Estas opciones se pueden escoger presionando el botón izquierdo del ratón sobre ellas. Una vez hecho esto, la figura seleccionada aparecerá en video invertido para indicar cuál fue seleccionada. Para proceder con el dibujo, primero se marca un cuadro con el ratón, lo cual se realiza de la siguiente forma: primero se presiona el botón izquierdo en un punto y, sin soltar el botón, se desplaza el dispoditivo para formar un rectángulo; hecho esto se debe soltar el botón. Cuando se suelta el botón, la figura se dibuja automáticamente, y se puede volver a dibujar otra figura. En caso de que se desee seleccionar otra figura, se presiona el botón derecho del mouse y la figura seleccionada que aparecía en video inverso, vuelve a su estado original; llegado a este punto, se puede seguir el procedimiento descrito anteriormente para seleccionar otra figura y dibujarla, o bien presionar el botón derecho del mouse una vez más para salir de esa función.

Las funciones que operan de esta forma son:

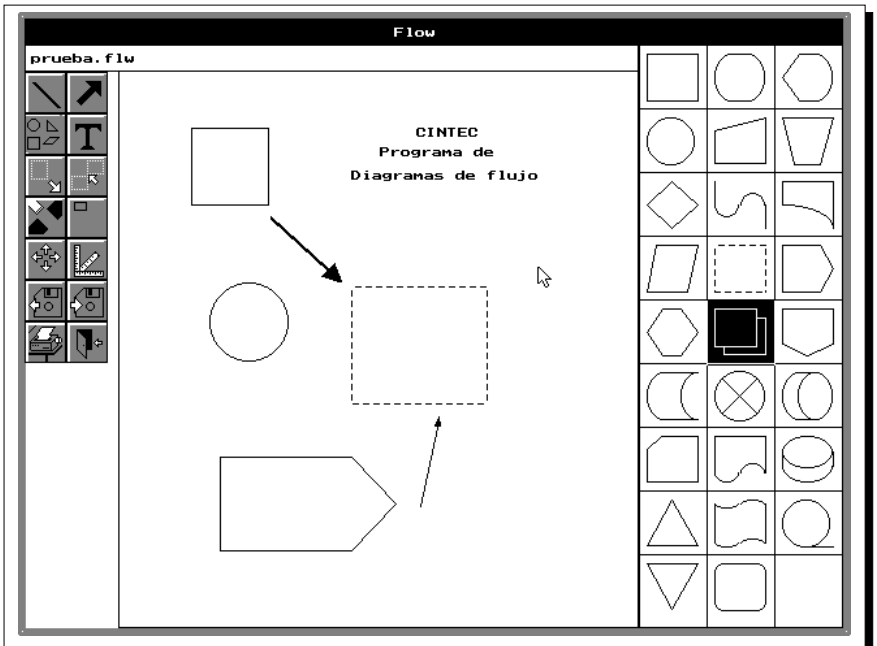

Figura 2. Opciones de dibujo del programa (lado derecho)

1) Dibuja_figuras.
2) Dibuja_lineas.
3) Dibuja_texto.
4) Dibuja_flechas.

El segundo grupo de funciones realizan la tarea de editar los objetos que fueron dibujados con el primer grupo de funciones. Las opciones de edición son: Copiar, Borrar, Mover y escalar objetos. Para su uso, primero se debe activar la opción de selección, una vez activada la opción simplemente se presiona el botón izquierdo del ratón para seleccionarlo, y dicho objeto aparecerá resaltado para indicar que fue seleccionado. Para salir de la opción de selección se presiona el botón derecho. Los objetos seleccionados por el método anterior pueden ser copiados, movidos o borrados, dependiendo de la función utilizada.

Las funciones que realizan las tareas anteriores son:

1) Selecciona_figuras.
2) Escalar_objetos.
3) Mover_objetos.
4) Copiar_objetos.
5) Borrar_objetos.

Por último se tienen las funciones del grupo 3, que son las encargadas de guardar en disco la información y de imprimir el dibujo. La rutina que maneja la impresión del dibujo está hecha para un dispositivo de impresión EPSON FX-80 o compatible.

Las funciones que realizan esta tarea son:

1) Salva_dibujo.
2) Carga_dibujo.
3) Imprime_dibujo.

Nota: El código fuente completo de esta aplicación se encuentra disponible en el CINTEC para cualquier persona que lo requiera.

Bibliografía

[1] Greg Voss. *"Programación orientada a objetos: Una introducción"*. Ed. McGraw Hill México D.F, 1994.

[2] Loren Heini. *"Power graphics using Turbo C"*. Ed. Adison Wesley.

[3] Clayton Walnum. *"Borland C++ power programming"*. Editorial QUE.

[4] Bjarne Stroustrup. *"The design and evolution of C++"*. Ed. Adison Wesley.

[5] Star Micronics 1000 User´s Guide.