HERRAMIENTA PARA EL DISEÑO Y DESARROLLO DE APLICACIONES EN BASE DE DATOS

TESIS DE GRADO

GERARDO FLORES DELGADO

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO
FACULTAD DE INGENIERIA

MEXICO, 1989

CONTENIDO

INDICE DE FIGURAS

INDICE DE TABLAS

Introducción

El gran desarrollo tecnológico que se ha dado durante la segunda mitad de este siglo en el campo de la electrónica, especialmente en la época de los 70's y la mitad de los 80's, ha hecho posible que el uso de las computadoras se extienda a casi todos los ámbitos de la vida cotidiana del ser humano.

En esta época aparecen en la industria una gran cantidad de compañías que se dedican a la fabricación de equipos de cómputo. Este hecho permite reducir el costo de éstos productos haciendolos cada vez más accesibles.

Por otro lado, el aumento de las actividades comerciales y administrativas aunado al vertiginoso avance experimentado por los medios de comunicación durante los ultimos años, han propiciado que los negocios y empresas requieran de información oportuna y veraz para apoyarse en la toma de decisiones que les permita a su vez operar adecuadamente.

Para organizar de manera eficiente esa información, se ha recurrido cada vez más al uso de los equipos de cómputo desarrollando sistemas de información que generalmente se apoyan en el uso de Base de Datos.

Los sistemas de Base de Datos han tenido una gran aceptación para el desarrollo de aplicaciones en computadora, debido a que proporcionan una manera eficiente para manejar y recuperar grandes volúmenes de información.

Durante el desarrollo de un sistema en Base de Datos, una de las actividades más laboriosas es la que se encarga de diseñar el programa que permite obtener los datos para alimentar al sistema.

Sin embargo, hasta el surgimiento de las microcomputadoras, no se había logrado contar con métodos adecuados para actualizar los datos de un sistema debido a las limitaciones que presentaban los equipos anteriores.

Cuando aparecen en el mercado las computadoras personales (*PC*) se revoluciona el campo de la computación y se abre un mundo de posibilidades para el desarrollo de sistemas.

Entre las caractrerísticas de las *PC's* se encuentra la capacidad de controlar totalmente la pantalla de la computadora. Esta característica ha hecho posible alimentar los datos de un sistema a través de un formato de captura.

Un formato de captura es un método que se usa para leer datos de la pantalla en forma eficiente y segura. Está compuesto por varios campos de lectura y cuenta con las siguientes características:

- Permite validar los datos que se lee en cada campo.
- Se pueden editar los datos que contiene cada campo.
- Proporciona ayuda al capturista sobre el tipo de datos que se deben suministrar en cada campo.
- Permite moverse a través de todo los campos que contiene mientras dura el proceso de captura.
- Limita la entrada de datos a la longitud del campo.

El objetivo de este trabajo consiste en desarrollar una herramienta (Editor de Pantallas **EDP**), que pueda ser usada para diseñar formatos de captura en la pantalla de la computadora de manera fácil y rápida.

Esta herramienta esta compuesta por un editor de pantallas y un generador de código.

El editor de pantallas se usa para definir diseñar y definir formatos de captura sobre la pantalla de la computadora.

El generador de código se encarga de generar un programa que permita aplicar el formato definido en el editor durante el proceso de captura de datos para un sistema.

Uno de los atractivos que proporciona esta herramienta es que su operación no depende de ningún paquete o sistema específico, aunque su uso permite definir formatos de captura que pueden ser aplicados en varios sistemas.

En la siguiente figura se muestra el modelo del Editor de Pantallas.

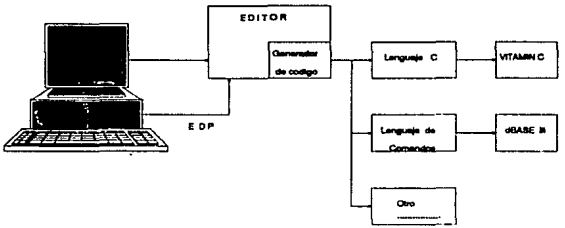


FIG. 1 MODELO DEL EDITOR DE PANTALLAS

Al generador de código se le pueden integrar varios módulos para generar programas que pueden ser usados en diferentes sistemas. Dentro de los alcances de este trabajo sólo se desarrollaron dos módulos. Uno de ellos se encarga de generar programas para definir formatos usando el lenguaje de comandos **dBASE III,** y el otro módulo genera programas en lenguaje de programación *C*, que se apoyan en el uso de una librería de funciones llamada **Vitamin C.**

El programa esta enfocado para ser usado principalmente por el programador de sistemas o de aplicaciones en computadora de tal manera que éste pueda contar con un método para obtener formatos para captura de datos.

Este documento está organizado de la siguiente manera:

El capítulo 1 contiene un análisis de los sistemas similares que se tomaron como base para definir algunas características del programa desarrollado como parte de este trabajo de tesis.

En el capítulo 2 se contempla el diseño y desarrollo del editor de pantallas así como el manual de usuario del programa.

En el capítulo 3 se describen las características y funciones del generador de código y los módulos que tiene integrados, además se incluye el procedimiento a seguir para agregar otros módulos a ésta parte.

El capítulo 4 se incluye la librería de funciones en que se apoya el programa generado en el lenguaje *C*.

Por último en el capítulo 5 se presenta un ejemplo de aplicación para ilustrar la forma en que se pueda aplicar el programa.

Capítulo 1. Antecedentes

En la actualidad muchos paquetes y sistemas comerciales hacen uso de un formato de captura para obtener los datos que manejan. Algunos contienen un formato predefinido, otros permiten al usuario definir sus propios formatos de acuerdo a sus necesidades y también existen paquetes que funcionan para definir formatos de captura exclusivamente para un sistema determinado.

Algunos de los sistemas y paquetes que cuentan con la posibilidad para definir formatos de captura son: el sistema de Base de Datos **dBASE III**, el sistema de Base de Datos **UNIFY**, el paquete **FLASHCODE** y la librería de funciones **Vitamin C**. En este capítulo se presenta un análisis de cada uno de ellos acerca de la opciones que permiten para definir un formato de captura y las características de éstos últimos.

El sistema **dBASE III** dispone de dos opciones para alimentar y actualizar la información de una Base de Datos. Por un lado genera un formato para captura de datos a través de la estructura de la base de datos definida, que se presenta al usuario cada vez que quiera agregar o modificar un registro. Por otro lado, cuenta con la posibilidad de que el usuario defina un formato de captura de acuerdo a sus necesidades. Para definir este formato, el usuario debe escribir un programa usando el lenguaje de comandos de **dBASE III**.

Los tipos de datos que se pueden manejar en este sistema son: cadena de caracteres, números enteros y reales, fechas y textos.

Las características del formato de captura que utiliza este sistema permiten hacer los siguiente:

- asignar valores iniciales a los campos.
- validar y formatear los datos de un campo.
- mover el cursor entre los campos del formato durante la captura.
- editar los datos que contiene cada uno de los campos.

- limitar la entrada de datos a la longitud del campo.

La desventaja que presenta este sistema cuando se utiliza el formato que se basa en la estructura de la base de datos, es que generalmente resulta insuficiente para las necesidades del usuario y éste tiene que emplear una cantidad de tiempo adicional para definir su propio formato de captura.

El sistema de Base de Datos **UNIFY**, también dispone de dos opciones para alimentar y actualizar una base de datos en forma similar a **dBASE III**. Utiliza un formato generado automáticamente a partir de la estructura de la base de datos y además cuenta con un programa Editor de Pantallas (**PAINT**) para definir formatos de captura. El procedimiento que sigue este programa para editar formatos es a través de cuatro modos de operación que funcionan a partir de comandos. Estos comandos especifican las características del formato que se desea definir en la pantalla. De esta manera aunque el usuario no tiene control sobre la pantalla, puede visualizar la forma que va tomando el formato al momento de definirlo.

Los tipos de datos que se pueden manejar en este sistema son: cantidad, binario, fecha, número real y entero, cadena de caracteres,texto y hora.

Las características del formato de captura que proporciona el formato de captura permiten hacer lo siguiente:

- formatear los datos de cada campo.
- editar los datos que contiene cada uno de los campos.
- limitar la entrada de datos a la longitud del campo.

En este tipo de sistema el proceso de captura se lleva a cabo a través de la pantalla de una terminal, por esta razón las características del formato se encuentran limitadas ya que su operación depende de las características propias del equipo de cómputo donde se encuentre instalado.

La desventaja de este programa con respecto al procedimiento que sigue para definir un formato es que requiere mucha interacción con el usuario debido a que durante la especificación del formato es necesario cambiar de un modo de operación a otro y proporcionar los datos de la definición para que ésta se lleve a cabo en la pantalla. Este hecho le resta flexibilidad al programa.

El paquete **FLASHCODE** contiene una Librería de Funciones y un programa integrado por tres módulos: un Editor de Pantallas, un Editor de Ventanas y un Generador de Código.

La Librería de Funciones contiene los programas necesarios para usar ventanas dentro del sistema **dBASE III**. Estos programas se dejan residentes en la memoria de la computadora y pueden ser llamados a través de un comando durante la ejecución del programa.

El Editor de Pantallas permite definir campos para lectura de datos y caracteres de texto directamente en la pantalla de la computadora. Para definir un formato en la pantalla se utilizan menús, formas para captura de datos y comandos asociados a las teclas de función del teclado de la computadora.

El Editor de Ventanas permite definir menús de opciones dentro de una ventana que pueden ser usados desde el sistema **dBASE III**.

El Generador de Código se encarga de generar programas completos en lenguaje de comandos **dBASE III** para capturar y recuperar datos por medio de un formato definido en el editor.

Los archivos de salida que se generan incluyen además del formato de captura: la estructura de una base de datos que contiene los campos definidos, varios menús de opciones, uno o varios archivos de índice para realizar búsquedas rápidas y los mensajes apropiados de ayuda y error.

Los tipos de datos que se pueden manejar y las opciones disponibles durante la captura para este paquete son las mismas que se permiten en el sistema **dBASE III**.

Las desventajas de este paquete son: que solo se pueden generar programas para ser usados en **dBASE III** y que todos los menús y mensajes son desplegados en inglés.

La Librería de Funciones **Vitamin C**, es un conjunto de funciones en lenguaje *C* que se pueden usar para desarrollar diversas aplicaciones, entre ellas se encuentra la posibilidad de definir formatos para capturar y recuperar datos en la pantalla.

El procedimiento que se sigue para definir un formato de captura con esta librería consiste en escribir un programa en lenguaje *C* usando las funciones adecuadas.

Los tipos de datos que se pueden manejar son: cadenas de caracteres y números enteros y reales en diferentes formas.

Las características del formato de captura que proporciona esta librería permiten hacer lo siguiente:

- asignar valores iniciales a los campos.
- validar y formatear los datos de cada campo.
- desplegar mensajes de ayuda para cada campo.
- editar los datos que contiene cada uno de los campos.
- mover el cursor entre los campos del formato durante la captura.
- limitar la entrada de datos a la longitud del campo.

La desventaja que presenta esta librería es que genera mucho código para los programas y se emplea una gran cantidad de tiempo para desarrollar la aplicación.

Capítulo 2. Editor de Pantallas

Durante el desarrollo de un sistema es conveniente contar con un procedimiento eficiente para obtener los datos que permitan al sistema proporcionar información confiable y consistente.

Uno de los procedimientos que normalmente se utilizan es el de programación. Este procedimiento consiste en diseñar un formato y escribir un programa en algún lenguaje de programación donde se especifiquen las instrucciones para definir el formato. Después, éste se traduce al lenguaje de máquina y se obtiene el código necesario para obtener el formato diseñado.

Este procedimiento puede llegar a repetirse varias veces si se comete algún error o se modifica el diseño del formato, incrementando el tiempo que se utiliza para definir el formato.

Recientemente, gracias al desarrollo de la tecnología de computadoras personales, han surgido otros procedimientos para definir formatos de captura entre los que se encuentra el uso de programas editores de pantalla.

El uso de estos programas permite reducir considerablemente el tiempo que se emplea para elaborar un formato.

No obstante, la mayoría de estos programas sólo pueden ser usados en el sistema para el que fueron definidos.

El programa que se presenta aquí, contiene un editor de pantallas y contempla la posibilidad de generar el código necesario para obtener formatos de captura que puedan ser usados para la lectura de datos.

Este programa, llamado Editor de Pantallas (**EDP**), puede ser considerado como un procedimiento alternativo para diseñar formatos de captura a través del uso directo de la pantalla de la computadora.

Una de sus características importantes es que a pesar de que funciona en forma independiente de cualquier paquete o sistema, puede generar programas en diferentes lenguajes para capturar, actualizar o recuperar datos en la pantalla como: lenguaje *C* y lenguaje de comandos **dBASE III** entre otros.

El procedimiento que se sigue consiste en usar el *editor* para diseñar un formato en la pantalla y por medio del *generador* de código seleccionar el tipo de programa que se desea obtener como salida.

Si se sigue este procedimiento, se puede conseguir un ahorro de tiempo considerable durante el desarrollo de un proyecto o sistema, ya que se estima que es posible obtener el diseño de los formatos más complicadas empleando tan solo unos cuantos minutos.

Este capítulo contiene todos los aspectos relacionados con el **EDP** que incluyen: las consideraciones de diseño, las características del programa, la manera en que se desarrolló y se implantó en la computadora así como las estructuras de datos que se utilizan y el manual de usuario.

2.1 Diseño y Desarrollo del Editor de Pantallas (EDP)

DISEÑO

El **EDP** tiene como funciones permitir la definición y edición de formatos mediante el uso directo de la pantalla de la computadora y proporcionar una salida que sirva como base para escribir un programa que contenga las instrucciones necesarias para aplicar el formato en el proceso de captura y recuperación de datos.

CARACTERISTICAS DEL EDITOR

Para llevarlas a cabo, se apoya en el uso de una *libreria* de funciones, un *descriptor* de la pantalla y una *interfaz* de entrada/salida.

El diagrama de bloques del **EDP** se muestra en la siguiente figura.

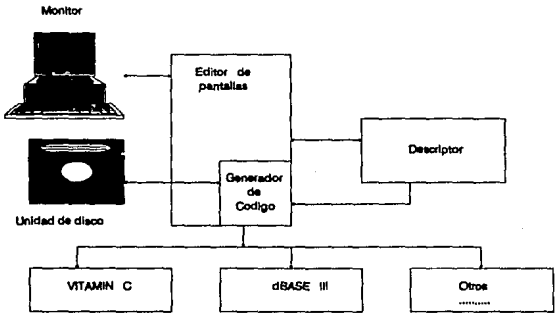


Fig. 2 DIAGRAMA DE BLOQUES DEL PROGRAMA

La *libreria* consiste en un conjunto de funciones que se encarga de controlar el flujo de información que se despliega en la pantalla.

La *interfaz* de entrada/salida permite realizar el manejo de archivos para las operaciones de lectura y escritura a disco. Por medio de ésta, la entrada al programa puede ser proporcionada en forma externa.

La entrada externa consiste en leer un archivo que contiene un *descriptor* de la pantalla definido previamente dentro del *editor* o en crear un archivo nuevo. De otra manera, el *editor* proporciona la entrada internamente creando un *descriptor* que no contiene datos y presentando la pantalla vacia.

El *descriptor* de la pantalla esta compuesto por cuatro estructuras de datos que representan cualquier formato definido en la pantalla. Estas estructuras son: una matriz de datos, un registro de ventanas, una tabla de campos y una tabla de cajas.

A través de estas estructuras y algunas variables de control, el *editor* permite escribir texto, definir ventanas, campos y cajas para definir y editar formatos en la pantalla.

El contenido de la pantalla se representa en la *matriz* de datos o *imágen* de la pantalla. Su dimensión es igual al tamaño lógico de una pantalla de la computadora en modo texto, más una columna por cada renglón para colocar una marca de fin de linea, Es decir, 25 renglones de 81 columnas cada uno.

En lo sucesivo se hará referencia a la *matriz* o *imágen* de la pantalla indistintamente.

Para representar en la *matriz* el contenido de la pantalla, cada elemento de ésta se forma a partir de un tipo y un valor para cada carácter que puede estar presente sobre la pantalla. Estos tipos son: tipo *texto*, tipo *campo*, tipo *caja* y tipo *ventana*. Esta clasificación permite manipular en forma independiente cada una de las definiciones permitidas por el *editor* y proporciona la consistencia adecuada para los datos que contiene la *matriz*. Con esto se mantiene un control de cada elemento de la *matriz* para asegurar que cada localidad no sea ocupada por más de un tipo de carácter a la vez. Es decir que el área o región ocupada por alguna definición en la *matriz* no puede ser ocupada por otra definición.

A cada caracter de la pantalla le corresponde un elemento de la *matriz* y viceversa. Cada elemento es de dos bytes y contiene un número que se forma como se ilustra en la figura 3. En el byte más significativo se mantiene el tipo de caracter, en el menos significativo el valor que le corresponde.

Figura 3. Contenido de cada elemento de la *matriz*.

El valor asociado tiene un significado diferente de acuerdo a su tipo como se muestra en la siguiente tabla.

TIPO	VALOR	SIGNIFICADO
TEXTO	código	Valor del caractér en la tabla ASCII
CAMPO	índice	Indice correspondiente en la tabla de campos
CAJA	índice	Indice correspondiente en la tabla de cajas
VENTANA	nulo	No tiene valor asociado

Tabla 1. Clasificación de caracteres

Cuando el *editor* proporciona la entrada al programa en forma interna se presenta vacia la pantalla y a cada elemento de la *imágen* de la pantalla se le asigna un número que representa a un caracter en blanco.

El rango de valores que puede tomar el valor asociado a cada tipo de caracter se encuentra entre **0** y **255**. Este rango permite representar todos los caracteres incluidos en el código ascii extendido y es posible definir hasta **256** campos y **256** cajas. La única restricción en este sentido sería disponer de la memoria suficiente para mantener los datos necesarios.

CARACTERISTICAS DE LA PANTALLA

Para diseñar un formato sólo se dispone de una pantalla completa para simplificar la operación del programa. El cursor se puede colocar en cualquier lugar de ésta usando para ello todas las teclas disponibles para el movimiento del cursor.

El movimiento del cursor se lleva a cabo libremente. Esto significa que si el cursor se lleva a alguno de los extremos de la pantalla y se sigue avanzando o retrocediendo, éste se moverá hacia otro extremo de la pantalla.

A través del *editor* se pueden escribir y borrar caracteres de texto, definir una ventana de trabajo, campos para lectura de datos y cajas para enmarcar alguna región de la pantalla. El procedimiento que se sigue se lleva a cabo directamente sobre la pantalla. Esta característica permite visualizar continuamente como se va creando el formato.

OPERACION DEL EDITOR

Para facilitar la operación del *editor*, las opciones disponibles se ejecutan a partir de la especificación de una orden o mandato. Cada mandato esta asociado a una de las teclas de función que se encuentran en el la sección de funciones del teclado de la computadora.

Durante la operación realizada por un mandato, siempre se dispone de una opción para cancelar su ejecución. Este hecho permite proteger el contenido de la pantalla contra modificaciones que no se desean.

El programa proporciona la facilidad de consultar en cualquier momento un menú de ayuda que contiene todos los mandatos disponibles dentro del programa.

La ejecución del programa se basa en una función que se encarga de obtener los caracteres presionados en el teclado y analizar si corresponde a una tecla de función o un caracter de texto para llamar a la función adecuada que el efecto esperado. Una vez concluida la operación, se regresa el control a la función principal para obtener y analizar más caracteres.

El *editor* mantiene una línea de estado que se usa para desplegar en la pantalla cierta información que puede ser de utilidad para el usuario. Para controlar ésta línea, se usa un mandato que opera en forma de switch permitiendo desplegar o no dicha información. Más adelante en esta misma sección se describirá el contenido de la línea de estado. Cuando se entra al programa, la línea de estado se presenta en la parte inferior de la pantalla.

DESARROLLO

La pantalla de una computadora generalmente se maneja en forma similar al primer cuadrante *I* de un par de ejes cartesianos. La diferencia estriba en que en la pantalla de la computadora la coordenada *(24,0)* corresponde a la coordenada *(0,0)* en el eje cartesiano. Esto implica que al avanzar sobre el eje de las ordenadas en la pantalla, el valor de la ordenada disminuya en lugar de aumentar.

En el ámbito de la programación se ha establecido como una costumbre nombrar como x a la ordenada y como Y a la abscisa en forma contraria al uso que se le da en Matemáticas.

De esta manera, a cada posición *(x,y)* de la pantalla, se le refiere como una coordenada, donde *x* es el número de renglón y *y* es el número de la columna.

La posición del cursor sobre la pantalla se controla por medio de dos variables que forman una coordenada *(r_pos,c_pos)*, *r_pos* contiene el renglón y *c_pos* la columna. Estas variables permiten también tener acceso inmediato al elemento correspondiente en la *imágen* de la pantalla.

Cuando el usuario pulsa una de las teclas definidas para movimiento del cursor (para mayor información consulte la sección correspondiente al movimiento del cursor en el **Manual de Usuario**) el programa verifica que éste no se salga de los límites de la pantalla. Para controlar estos límites, se usan cuatro variables que forman dos coordenadas. La coordenada mínima *(r_min,c_min)* y la coordenada máxima *(r_max,c_max)*

donde se puede colocar el cursor. El valor de éstas permite mover el cursor a través de las líneas **1-25** y las columnas **1-80**.

La edición de ventanas, cajas y campos, se apoya también en el uso de cuatro variables que forman un par de coordenadas, la coordenada inicial *(r_arr,c_izq)* y la coordena final *(r_aba,c_der)*. A través de estas variables es posible ubicar la región que ocupa una definición tanto en la pantalla como en su *imágen*.

El uso de todas estas coordenadas se ilustra en la siguiente figura.

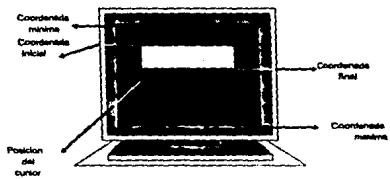


Fig. 4 Coordenadas para controlar la pantalla

El ambiente del *editor* se controla a través de una variable que funciona como un registro de banderas. Cada uno de los bits de éste registro se utiliza para controlar una variable de tipo lógico. El formato de éste registro se muestra en la figura 5. Su contenido y funcionamiento se describe en la siguiente sección.

0	FC	FM	FC	OC	AD	IF	IB	FD	BD	V	E	MI	M	F	TM

Figura 5. Registro de banderas

El *descriptor* de la pantalla se construye y se mantiene a través de cuatro operaciones de edición fundamentales: de ventanas, de cajas, de campos y de texto.

VENTANAS

El uso de ventanas representa una forma de mantener organizada la información que se despliega en la pantalla. En la computadora nor-

malmente sólo se tiene lugar para una página de información que ocupa toda la pantalla. Con el uso de ventanas es posible llenar la pantalla de información y después encimar toda o parte de ella con más información dentro de una ventana. En el momento que se termina de usar una ventana, se restaura la región de la pantalla que quedó *abajo* de ésta. Para mantener organizada la pantalla cuando se usan ventanas, muchas veces se coloca un marco a su alrededor.

Muchos sistemas tienen contemplado durante su operación el manejo de ventanas. Por esta razón el *editor* cuenta con la posibilidad de abrir una ventana, de tal manera que la definición de un formato quede contenida dentro de ésta. Para simplificar el uso del *editor* y usar en forma independiente cada formato, sólo se permite abrir una ventana de trabajo por cada formato que se desee definir.

El *editor* de ventanas realiza dos operaciones: agregar o eliminar la definición de una ventana.

Agregar

Para agregar la definición de una ventana, el *editor* permite dibujarla sobre la pantalla y colocarla en la posición deseada usando las teclas para movimiento del cursor. La coordenada inicial se toma de la posición del cursor al momento de especificar la operación y la coordenada final se va ajustando internamente mientras se va dibujando el marco en la pantalla. Los demás atributos que definen a una ventana en el EDP, se pueden consultar en el manual de referencia que se encuentra en la *sección 2* de este capítulo.

Mientras se están definiendo los atributos de la ventana, ésta se va desplegando en la pantalla de tal manera que el usuario pueda visualizar la forma que va tomando.

Al momento de definir la ventana se realizan los siguientes cambios en la pantalla y el *descriptor* de la pantalla:

- Las coordenadas y atributos seleccionados se mantienen en el registro de ventanas.
- Se forma un número con el tipo de caracter (VENTANA) y su valor asociado (0) (para mayor claridad véase la Figura 6 y la Tabla 1) que se asigna a cada elemento de la *imágen* de la pantalla que corresponde con el marco de la ventana.
- En la pantalla se presenta la ventana con los atributos que se seleccionaron.

Estos cambios se ilustran en la siguiente figura.

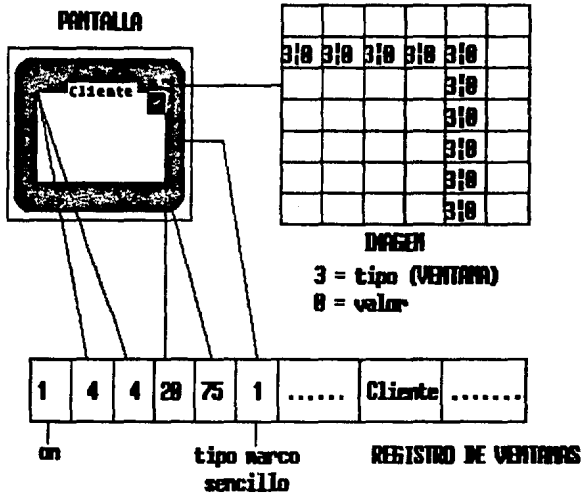


Figura 6. Cambios en el *editor* cuando se define una VENTANA.

Cuando se acepta la definición, el contenido de las coordenadas inicial y final (que ubican la posición de la ventana en la pantalla) se asigna a las coordenadas mínima y máxima, respectivamente (que definen los límites de la pantalla) y el cursor se coloca en la primera posición dentro de la ventana.

Lo anterior provoca que desde ese momento el tamaño de la pantalla se tome con respecto a las coordenadas de la ventana y sólo se pueda utilizar el área contenida dentro del marco.

De esta manera las definiciones que se hagan posteriormente quedarán ubicadas automáticamente con respecto a las coordenadas de la ventana y no a las de la pantalla.

Eliminar

La operación para eliminar una ventana borra totalmente la definición tanto de la pantalla como del *descriptor*, reemplazando la región que ocupa el marco por caracteres en blanco. Esto permite que esa región pueda der ocupada por otra definición o caracteres de texto.

Antes de realizar la operación, el programa solicita al usuario que ésta sea confirmada para prevenir que la definición de la ventana sea eliminada involuntariamente.

Cuando se elimina la ventana, se restaura el valor inicial de las coordenadas mínima y máxima para permitir que el cursor pueda moverse nuevamente a través de toda la pantalla.

CAMPOS

La posibilidad de definir campos es una de las operaciones primordiales del *editor*.

Las características o atributos que representan a un campo para lectura de datos, son diferentes y varían en cada paquete, librería o Base de Datos que se encuentran en el mercado. En nuestro caso, los atributos que representan a un campo se seleccionaron a partir de un análisis de algunos programas comerciales que se juzgó conveniente (para mayores detalles consulte el capítulo I de éste trabajo). Este hecho responde a la pretensión de usar el Editor de Pantallas para diseñar formatos de captura en general, para cualquier sistema que necesite leer datos de la pantalla en forma organizada y eficiente. Estos atributos son: una coordenada *(x,y)* para ubicar la posición dentro de la pantalla, el nombre del campo o la variable donde se debe guardar el dato leído, el tamaño del campo en caracteres, el tipo de dato y una mascara para validar cada caracter del campo.

Para controlar el número de campos que se definen, se usa un contador de campos que al mismo tiempo sirve como índice para obtener el elemento que le corresponde en la tabla de campos cuando se agrega una definición.

La edición de campos realiza cuatro operaciones: agregar, desplazar, modificar y eliminar la definición de un campo.

Las operaciones de desplazamiento, modificación y eliminación requieren que el cursor sea colocado dentro del área de lectura del campo que se desea afectar.

Agregar

Para agregar un campo, el *editor* toma como coordenada inicial la posición del cursor al momento de especificar la operación, se presenta un formato en la pantalla donde se deben especificar los demás atributos del campo y después de llenar el formato, se abre una ventana en la pantalla del tamaño del campo que puede ser desplazada hacia la posición donde se desea colocar el campo. Durante el desplazamiento se ajusta la coordenada inicial automáticamente.

Cuando se agrega un campo, se efectúan los siguientes cambios en el *editor*:

- Se incrementa el contador de campos y los atributos se guardan en la tabla correspondiente en el lugar indicado por el contador.
- En la *imágen* de la pantalla, se registra en cada elemento que queda comprendido dentro del área de lectura del campo, el número formado con el tipo de carácter (CAMPO) y el valor del contador que representa el índice que le corresponde a la definición en la tabla de campos (para mayor claridad véase la Figura 3 y la Tabla 1).
- En la pantalla de la computadora se despliega, a partir de la coordenada inicial, una cadena de caracteres del tamaño del campo en color inverso para representar el área de lectura del campo. Esta cadena se forma a partir de la letra que indica el tipo de campo.

Estos cambios se ilustran en la siguiente figura.

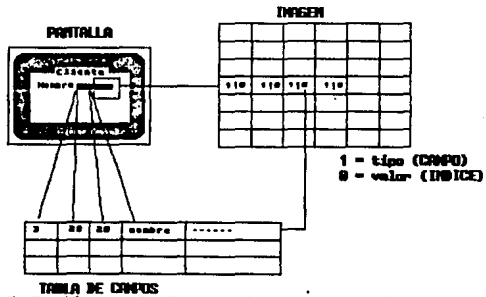


Figura 7. Cambios en el *editor* cuando se define un CAMPO.

Desplazar

La operación de desplazamiento, permite trasladar de un lugar a otro la posición de lectura de un campo sobre la pantalla.

Para hacer el desplazamiento, se abre una ventana en la pantalla del tamaño del campo, que puede moverse para colocar el campo en la posición que se desee. El campo sólo puede ser colocado en algún lugar de la pantalla que no interfiera con otra definición o caracteres de texto.

Si la operación concluye adecuadamente, se actualiza la nueva posición tanto en la tabla de campos como en la *imágen* de la pantalla.

Modificar

La operación para modificar un campo, permite cambiar sus atributos. Para realizar esta operación, se presenta en la pantalla un formato que permite modificar los atributos actuales del campo. Cuando se proporcionan las modificaciones deseadas, se efectúan los siguientes cambios en el *editor*:

- La tabla de campos se actualiza con los nuevos valores.
- Si se modifica el tamaño, se abre una ventana de acuerdo al nuevo tamaño para desplazar el campo a otro lugar si es necesario. En este caso se actualiza la nueva posición del campo en la *imágen* de la pantalla.

Eliminar

La operación para eliminar un campo, borra totalmente la definición tanto del *descriptor* como de la pantalla. Las modificaciones que se llevan a cabo se describen a continuación:

- El área ocupada por el campo tanto en la pantalla como en la *imágen*, se reemplaza por una cadena de caracteres en blanco para permitir que esa región pueda volver a ser ocupada.
- La tabla de campos se compacta para evitar que queden huecos. Este procedimiento implica actualizar en la *imágen* de la pantalla los índices de los campos afectados por la compactación para mantener la consistencia del *descriptor* de la pantalla.

Antes de realizar esta operación, el programa solicita al usuario que la operación sea confirmada para prevenir que la definición del campo sea eliminada involuntariamente.

Cuando se elimina un campo, el contador de campos es decrementado en una unidad.

CAJAS

Una caja puede estar formada por un marco, una linea horizontal o una linea vertical. Esta característica puede resultar útil para separar parte del contenido de un formato y atraer la atención del operador hacía alguna región específica. Los caracteres que forman parte de una caja, son los únicos que pueden quedar encimados entre sí sin afectar la consistencia de la *imágen* de la pantalla.

Para controlar el número de cajas que se definen, se usa un contador que al mismo tiempo sirve como índice para obtener el elemento que le corresponde en la tabla de cajas cuando se agrega una definición.

La edición de cajas realiza tres operaciones: agregar, desplazar y eliminar la definición de una caja.

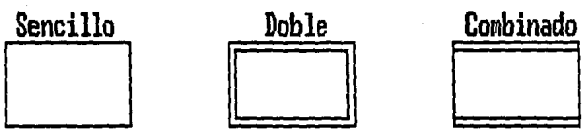
Las operaciones de desplazamiento y eliminación requieren que el cursor sea colocado dentro del marco de la caja que se desea afectar.

Agregar

Para agregar una caja, el *editor* toma como coordenadas inicial la posición del cursor al momento de especificar la operación y permite al usuario dibujar el marco y colocarlo en la posición deseada con el uso de las teclas para movimiento del cursor. Durante este proceso, el valor de las coordendas inicial y final se va ajustando automáticamente.

Para seleccionar el marco de la caja se cuenta con tres opciones: sencillo, doble o combinado. En la siguiente figura se ilustran los tipos de marco disponibles.

Figura 8. Tipos de marco disponibles.



Cuando se agrega una caja se realizan los siguientes cambios en el *editor*:

- Se incrementa el contador de cajas y el tipo de marco y las coordenadas se guardan en la tabla correspondiente en el lugar indicado por el contador.
- En la *imágen* de la pantalla, se registra en cada elemento que corresponde con el marco de la caja, el número formado con el tipo de carácter (CAJA) y el valor del contador que representa el índice que le corresponde a la definición en la tabla de cajas (para mayor claridad véase la Figura 3 y la Tabla 1).
- En la pantalla de la computadora se dibuja el marco de la caja en la región definida.

Estos cambios se ilustran en la siguiente figura.

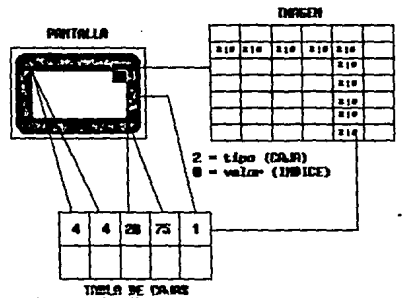


Figura 9. Cambios en el *editor* cuando se define un CAJA.

Desplazar

La operación de desplazamiento, permite trasladar de un lugar a otro la posición de un marco sobre la pantalla.

Para hacer el desplazamiento, se abre una ventana en la pantalla del tamaño del marco, que puede moverse para colocar la caja en su nueva posición. El marco sólo puede ser colocado en alguna región de la pantalla que no interfiera con el área señalada para la lectura de un campo o caracteres de texto.

Si la operación concluye adecuadamente, se actualiza la nueva posición tanto en la tabla de cajas como en la *imágen* de la pantalla.

Eliminar

La operación para eliminar una caja, borra totalmente la definición tanto del *descriptor* como de la pantalla. Las modificaciones que se llevan a cabo son las siguientes:

- Cada elemento de la región ocupada por el marco de la caja tanto en la pantalla como en la *imágen*, es reemplazado por el número que representa un caracter en blanco.
- La tabla de cajas se compacta para evitar que queden huecos. Este procedimiento involucra actualizar en la *imágen* de la pantalla los índices de cada elemento que forma parte del marco de una de las cajas afectadas por la compactación.
- En la pantalla se refresca el marco de todas las cajas definidas. Este procedimiento es necesario porque podría darse el caso de que el marco de la caja eliminada estuviera encimado con el marco de otras cajas.

TEXTO

La edición de texto en el *editor* permite la edición de caracteres y la edición de líneas.

Durante la edición de texto sobre una línea, cada una es tratada en forma independiente, es decir, la edición realizada en una línea no afecta a las demás.

En la edición de caracteres, para insertar o borrar un carácter se debe realizar un desplazamiento hacia adelante o hacia atrás de los caracteres que se encuentran a la derecha o izquierda de la posición del cursor, respectivamente.

De esta manera, es posible detectar alguno de los siguientes casos durante la edición de caracteres en el *editor*:

- Cuando en la línea donde queremos insertar o borrar un carácter se encuentre definida el área para la lectura de un campo o el marco de una caja, sólo habrá desplazamiento de caracteres mientras no se llegue a los límites de la pantalla, el límite del área señalada para la lectura del campo o el marco de la caja, para evitar que el campo sea desplazado involuntariamente o que se causen daños al marco de la caja.

- Cuando el cursor se encuentra dentro del área de lectura de un campo o sobre el marco de una caja no es posible escribir o borrar caracteres.

La edición de líneas cuenta con la posibilidad de insertar o eliminar una línea.

En forma similar a la edición de caracteres, la inserción y borrado de líneas implica el desplazamiento hacia abajo o hacia arriba de las líneas que se encuentran debajo de la posición del cursor, respectivamente. En el *editor*, esta operación provocaría el desplazamiento de los campos o las cajas que estuvieran definidas en la parte de abajo, llevándonos a la situación de afectar las definiciones.

Para evitar lo anterior y hacer posible la edición de líneas, una línea dentro del *editor* se define como el área de un renglón de la pantalla que se encuentra entre:

- los límites de la pantalla
- el marco de una caja o dos marcos
- cualquier combinación de los anteriores

El tamaño de una línea es el número de caracteres de la línea sin incluir las delimitaciones.

Esto nos lleva a concluir que en un momento dado, no todas las líneas del *editor* tienen el mismo tamaño, lo cual también las hace diferentes.

De acuerdo con lo anterior, en la edición de líneas sólo se desplazarán aquellas líneas que se encuentran debajo del cursor que tengan el mismo tamaño o sean iguales a la línea afectada.

Si al insertar una línea, en las líneas involucradas estuviera definido un campo, éste será desplazado y su posición actualizada por el *editor*, pero si la línea se quiere eliminar, la operación no tendrá efecto. La razón por la cual sucede lo anterior es porque el *editor* cuenta con un mandato para cambiar de posición un campo, pero no dispone de un mandato para recuperar un campo eliminado.

Esto hace posible la edición de texto en el *editor* al mismo tiempo que protege las definiciones que contiene un formato.

Para apoyar la operación de inserción y eliminación de una línea en el *editor*, se usa una variable para indicar el último renglón de la pantalla que contiene caracteres de texto. Esto permite efectuar con mayor ra-

pidez la operación ya que sólo se desplazan las líneas que contienen caracteres de texto no tomándose en cuenta las líneas en blanco.

De aquí en adelante se describen otras operaciones que apoyan el funcionamiento del **EDP.**

ENTRADA/SALIDA

El manejo de archivos forma parte de la interface de entrada/salida, y se encarga de realizar las operaciones necesarias para guardar y recuperar formatos de captura a través de un archivo en disco, así como de crear los archivos donde se escribe el código de salida generado por el programa.

Para ejecutar estas operaciones el usuario debe especificar el nombre de un archivo.

La información que se guarda en un archivo que contiene la definición de un formato se escribe de la siguiente manera:

- Se calcula un número con el contenido de la *imágen* de la pantalla que se escribe al principio del archivo, para poder identificar a la hora de usarlo nuevamente, que contiene un formato que pertenece al **EDP** o que el archivo no haya sido alterado fuera del **EDP.**
- Se escribe el registro de ventanas.
- Se escribe el contador de campos. Si su valor es mayor o igual que cero, significa que se encuentran campos definidos y se procede a escribir la tabla de campos. De lo contrario la tabla no se escribe debido a que se encuentra vacia.
- Se escribe el contador de campos. Si su valor es mayor o igual que cero, significa que se encuentran cajas definidos y se procede a escribir la tabla de cajas. De lo contrario la tabla no se escribe debido a que se encuentra vacia.
- Se escribe la *matriz* o *imágen* de la pantalla y la variable que indica el número de la última línea en la pantalla que contiene caracteres de texto .

Para cargar un formato de captura en la pantalla primero se verifica el número que identifica que el archivo pertenece al *editor* o que la pantalla no fué alterada fuera del **EDP.** Si el resultado de la verificación es positiva, se lee el resto de la información en el *descriptor* de la pantalla y en las variables de control en el mismo orden en que fueron

escritas en el archivo. Después se despliega en la pantalla el contenido del *descriptor* de la pantalla.

LINEA DE ESTADO

La línea de estado tiene como función desplegar en la pantalla cierta información que se mantiene en el *descriptor* de la pantalla que puede ser de utilidad para el usuario.

Para realizar su función, se basa principalmente en el contenido de la *imágen* de la pantalla, el registro de banderas y la posición del cursor sobre la pantalla.

La información que se despliega en la línea de estado es la siguiente:

- Si el contenido de la pantalla ha sido modificado y no se ha salvado en un archivo.
- El nombre del archivo que contiene el formato definido en la pantalla.
- Si la definición está contenida o no dentro de una ventana definida por el usuario.
- El número de la caja donde se encuentra colocado el cursor.
- El nombre del campo donde se encuentra colocado el cursor.
- El modo de inserción durante la edición de caracteres en la pantalla.
- La posición del cursor sobre la pantalla.
- La tecla que se debe pulsar para consultar todos los mandatos del programa.

Para obtener la información que se despliega, se lleva a cabo el siguiente procedimiento:

- El nombre del formato y la posición del cursor se toman directamente de las variable globales que se usan para este propósito.
- Se verifica el bit correspondiente en el registro de banderas, para saber si el archivo ha sido modificado y si ésta siendo usando el modo de inserción.
- Se consulta en el registro de ventanas si está "encendida" la bandera que indica si se encuentra definida una ventana.
- De la *imágen* de la pantalla se toma el elemento que corresponde con el carácter que esta sobre la posición del cursor y se analiza su tipo:

Si es tipo CAMPO, se toma el nombre del campo de la tabla a partir del índice del campo que se encuentra en el del byte menos significativo del elemento.

Si es tipo CAJA, el número de ésta se toma directamente del byte menos significativo del elemento.

Si es tipo TEXTO y se encuentran cajas definidas, se analiza la tabla de cajas para saber si la posición del cursor queda contenida dentro de alguna de éstas y así obtener su número. El control de la línea de estado funciona en forma de switch (encendido/apagado) para desplegar o no su contenido en la pantalla.

Esta línea se puede desplegar en la parte inferior o en la parte superior de la pantalla. Si el cursor se trata de colocar sobre la línea donde se despliega el estado, el programa mueve su posición a la otra parte de la pantalla. Esta característica de la linea de estado permite el uso de toda la pantalla para definir cualquier formato.

En la siguiente figura se muestra el contenido de la línea de estado cuando comienza la ejecución del programa.

Figura 10. Información inicial de la línea de estado.

INICIALIZAR EL EDITOR

La inicialización del *editor* permite definir más de un formato de captura durante una misma corrida del programa.

Para ello cuenta con una opción que se encarga de inicializar el *editor*. Cuando se ejecuta se limpia la pantalla de la computadora, se inicializan las estructuras del *descriptor* de la pantalla así como todas las variables globales que controlan tanto la operación como el ambiente del EDP. El cursor se coloca al inicio de la pantalla y todo queda listo para diseñar un nuevo formato en *editor*.

2.2 Estructura de Datos

Como ya se mencionó en la sección anterior, un formato queda definido a través de un *descriptor* de la pantalla.

En esta sección se describe cada una de las estructuras de datos que forman parte del *descriptor*, así como las variables que se utilizan para controlar la operación del programa, haciendo uso del formato que se utiliza en los Manuales del **Sistema Operativo UNIX.**

Estas declaraciones se encuentran especificadas para lenguaje *C* que es el que se empleó para desarrollar el **EDP.**

Declaración de variables del Editor de Pantallas:

Registro de ventanas

NOMBRE

ventana

SINOPSIS

```
struct    sventana      {
char      flag;         /*BANDERA */
short     x1, y1, x2, y2;  /* UBICACION */
short     marco;        /* MARCO */
char      atr[MAXATR];  /* ATRIBUTOS */
char      tit[MAXTIT];  /* TITULO */
char      ctrl[MAXCTRL]; /* CONTROL */

} ventana;
```

FUNCION

Esta estructura se encarga de reservar la memoria necesaria para mantener los atributos que definen a una ventana. En la siguiente figura se muestra el registro de ventanas.

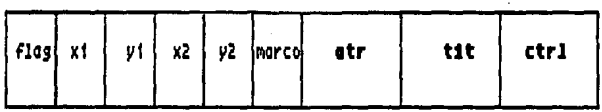


Figura 11. Registro de ventanas

Donde :

- BANDERA

se usa como variable lógica para indicar si se encuentra o no definida una ventana.

- UBICACION

contiene las dos coordenadas (*x1,y1*) y (*x2,y2*) que permiten ubicar el área de la pantalla ocupada por la ventana.

- MARCO

es un entero que contiene el tipo de marco seleccionado.

- ATRIBUTOS

es una cuerda de MAXATR caracteres formada con los atributos de la ventana.

- TITULO

es una cuerda de MAXTIT caracteres que contiene el encabezado que se incluye en el marco superior de la ventana.

- CONTROL

es una cuerda de MAXCTRL caracteres que contiene los atributos que definen las características de operación de la ventana.

Tabla de cajas

NOMBRE

caja

SINOPSIS

```
struct    scaja          {
short     x1, y1, x2, y2;   /* UBICACION */
short     tipo;             /* TIPO */
} caja;
```

FUNCION

Esta estructura se utiliza para formar una tabla donde se mantienen los atributos que definen a una caja. El tamaño

de la tabla se indica a través de la constante _CAJAS. En la siguiente figura se muestra la tabla de cajas.

	POSICION				TIPO
_CAJAS					

Figura 12. Tabla de cajas

Donde :

- UBICACION

contiene las dos coordenadas (*x1,y1*) y (*x2,y2*) que definen la región para dibujar el marco de la caja sobre la pantalla.

- TIPO

es un entero que contiene el tipo de marco seleccionado.

Tabla de campos

NOMBRE

campo

SINOPSIS

```
struct    scampo          {
short     x1, y1;         /* POSICION */
short     longC;          /* TAMAÑO */
char      nombre[MAXNOM]; /* NOMBRE */
char      pict[MAXLCMPO]; /* MASCARA */
char      tipoC;          /* TIPO */
} campo;
```

FUNCION

Esta estructura se utiliza para formar una tabla donde se mantienen los atributos que definen a un campo. El tamaño de la tabla se indica a través de la constante _CAMPOS. En la siguiente figura se muestra la tabla de campos.

	POSICION		TAMAÑO	NOMBRE	MASCARA	TIPO
_CAMPOS						

Figura 13. Tabla de campos

Donde :

- POSICION

contiene la posición (*x1,y1*) de lectura del campo sobre la pantalla.

- TAMAÑO

es un entero que contiene la longitud del campo en caracteres.

- NOMBRE

es una cadena de MAXNOM caracteres que contiene el nombre de la variable de memoria donde se debe depositar el dato leídos en la pantalla sobre el campo.

- MASCARA

es una cadena de caracteres de MAXLCMPO que contiene el formato donde se especifica el tipo de dato que puede tomar cada carácter que forma parte del campo.

- TIPO

es un carácter que contiene el tipo de dato que contiene el campo.

Matriz

NOMBRE

imágen

SINOPSIS

```
short    (* IMAGEN)    /* IMAGEN de la pantalla en memoria */
         [MAXCOLU+1];
```

FUNCION

Con esta estructura se forma una *matriz* donde se representa el contenido de la pantalla. La dimensión de esta *matriz* es de [MAXRENG] [MAXCOLU + 1] elementos de tipo entero que corresponde al tamaño de la pantalla.

Variables de control

NOMBRE

r_pos, c_pos

SINOPSIS

```
short    r_pos, c_pos;
```

FUNCION

contienen la coordenada (*r_pos,c_pos*) que indica la posición del cursor en la pantalla y se utilizan como índice para obtener el elemento que le corresponde en la *matriz*.

NOMBRE

r_min,c_min,r_max,c_max

SINOPSIS

```
short    r_min, c_min,
         r_max, c_max;
```

FUNCION

contienen la coordenada mínima (*r_min,c_min*) y la coordenada máxima (*r_max,c_max*) para controlar el movimiento del cursor sobre la pantalla.

NOMBRE

r_arr,c_izq,r_aba,c_der

SINOPSIS

```
short    r_arr, c_izq,
         r_aba, c_der;
```

FUNCION

contienen las coordenadas inicial (r_arr,c_izq) y final (r_aba,c_der) para ubicar la región que ocupa una definición sobre la pantalla.

NOMBRE

ulinea

SINOPSIS

```
short    ulinea;
```

FUNCION

es un entero que indica el número de la última línea en la pantalla que contiene caracteres de texto.

NOMBRE

ncampos

SINOPSIS

```
short    ncampos;
```

FUNCION

es un entero que contiene el número de campos definidos y también se usa como índice para la tabla de campos.

NOMBRE

ncajas

SINOPSIS

```
short    ncajas;
```

FUNCION

es un entero que contiene el número de cajas definidas y también se usa como índice para la tabla de cajas.

NOMBRE

edpArchivo

SINOPSIS

```
char     edpArchivo[MAXFILE];
```

FUNCION

es una cuerda de caracteres donde se mantiene el *path* o ruta del archivo que contiene la definición de un formato en edición.

NOMBRE

mandato

SINOPSIS

```
int     mandato;
```

FUNCION

es un entero que contiene el último mandato especificado.

NOMBRE

edpPrmpt

SINOPSIS

```
char    *edpPrmpt[];
```

FUNCION

es un apuntador que contiene la dirección del área de memoria donde se encuentran todos los mensajes que se despliegan sobre la pantalla.

NOMBRE

slin

SINOPSIS

```
short   slin;
```

FUNCION

es un número entero que indica el renglón donde se despliega la línea de estado.

NOMBRE

edpFlags

SINOPSIS

```
unsigned short          edpFlags;
```

FUNCION

es un registro de banderas que se utiliza para controlar el ambiente del programa. Cada uno de los bits de este registro controla una variable del programa. En la siguiente figura se muestra el formato del registro

REGISTRO DE BANDERAS

0	FC	FM	FE	OC	AD	IF	IB	FD	BD	V	E	MI	M	F	TM

Donde :

- TM

se enciende cuando se especifica al programa el switch para usar la configuración a color

- F

se enciende cuando se especifica un archivo como parámetro al programa

- M

se enciende cuando se modifica la pantalla

- MI

se mantiene encendida cuando se usa el modo de inserción para escribir caracteres

- E

se enciende cuando se despliega la línea de estado

- V

se mantiene encendida cuando el *descriptor* de la pantalla se encuentra vacio

- BD

se mantiene encendida cuando hay cajas definidos

- FD

se mantiene encendida cuando hay campos definidos

- IB

se mantiene encendida cuando el cursor se encuentra colocado dentro de una caja

-IF

se mantiene encendida cuando el cursor se encuentra colocado dentro del área de lectura para un campo

-AD

se enciende cuando se especifica la opción para generar un archivo con la declaración de los campos definidos

-OC

se enciende cuando se especifica la opción del para ordenar la tabla de campos

-FE

se enciende cuando se elige la opción que permite generar un formato para captura de datos

-FM

se enciende cuando se elige la opción que permite generar un formato para modificar datos.

-FC

se enciende cuando se elige la opción que permite generar un formato para consultar datos.

La memoria necesaria para obtener las estructuras de datos del *descriptor* de la pantalla, se solicita en forma dinámica al comienzo del programa. Con esto se evita que esta memoria se incluya dentro del archivo ejecutable y ocupe espacio adicional.

Las estructuras y constantes que se usan para hacer algunas de las declaraciones se encuentran especificadas en el archivo "edp.h". Todas las variables globales usadas por el programa, se encuentran declaradas en los archivos "edpmem.c" y "edpext.h".

2.3 Manual de usuario

El **EDP** es un programa que se usa para diseñar formatos para captura de datos sobre la pantalla de la computadora y generar el código de

salida necesario para aplicar el formato diseñado. En esta sección se incluye el manual de usuario del programa.

La operación del programa se lleva a cabo a través de la especificación de mandatos. Cada mandato se encuentra asociado a alguna de las teclas de función del teclado.

Para hacer una definición, solo se necesita colocar el cursor de la pantalla en el lugar deseado, especificar un mandato y proporcionar las características de la definición ya sea llenando algún formato o seleccionando las opciones de un menú.

Durante el llenado de esos formatos se pueden ocupar las teclas que se encuentran en la siguiente tabla.

TECLA	ACCION
→	Avanzar el cursor un carácter
←	Retroceder el cursor
|	Avanzar al campo siguiente
|	Regresar al campo anterior
Supr/Del	Borrar el carácter sobre el cursor
Retro/BackSpace	Borrar el carácter detrás del cursor
Ins	Seleccionar el modo de edición
Intro/Enter/Return	Avanzar al campo siguiente o terminar la captura desdesde el último campo del formato
Esc	Cancelar la captura

Tabla 2. Teclas disponibles para llenar un formato de captura.

La operación de cada uno de los mandatos se controla principalmente con el uso de dos teclas. La tecla correspondiente al mandato especificado se usa para indicar que se desea continuar con la ejecución del mandato o la selección de otros parámetros. La tecla [Esc] se usa para cancelar la operación. Si se pulsa ésta tecla, el diseño del formato no sufre ninguna alteración.

Se dispone de un menú de ayuda para consultar todos los mandatos del programa.

Durante la edición se maneja una linea de estado donde se despliega información considerada de interés para el usuario.

El manejo de archivos permite salvar el contenido de la pantalla en un archivo y recuperarlo posteriormente.

Para ejecutar el programa sólo es necesario proporcionar su nombre (**EDP**) desde el **Sistema Operativo**. Junto con éste se pueden especificar opcionalmente dos parámetros. En uno (-c) se indica al programa el uso de la configuración para monitor a color. El otro parámetro se usa para especificar un archivo que contiene una pantalla definida anteriormente dentro del *editor* o el nombre de la pantalla inicial.

La sintaxis que reconoce el programa al momento de ejecutarse es la siguiente:

edp [-c] [path]

donde,

edp es el nombre del programa

[-c] el parámetro para seleccionar monitor a color *[opcional]*.

[path] la ruta completa del archivo que contiene el formato que se desea cargar *[opcional]*

Los parámetros pueden ser proporcionados en cualquier orden.

Todos los mandatos disponibles dentro del **EDP** serán tratados en alguna parte dentro de esta sección agrupados de acuerdo a la función que desempeñan.

2.3.1 Descripción del teclado de funciones

MOVIMIENTO DEL CURSOR

El cursor (carácter '_' pulsante) indica la posición en la pantalla a partir de la cual cualquier operación efectuada dentro del *editor* tendrá efecto.

El cursor se mueve sobre la pantalla usando las teclas para movimiento lento y movimiento rápido del cursor.

Para mover el cursor en forma rápida dentro de una línea, se colocaron marcas en la pantalla cada 8 columnas (8,16, 24, ..., 72). De esta manera se puede mover el cursor de una marca a otra pulsando una tecla.

Las teclas que se pueden usar para mover el cursor se muestran en la siguiente tabla.

TECLA	ACCION
	MOVIMIENTO LENTO
→	Avanza el cursor una posición a la derecha
←	Avanza el cursor una posición a la izquierda
↑	Avanza el cursor 1 línea hacia arriba
↓	Avanza el cursor 1 línea hacia abajo
Tab	Mueve el cursor a la siguiente marca
sTab	Mueve el cursor a la marca anterior
Intro/Enter/Return	Mueve el cursor al comienzo de la siguiente línea
	MOVIMIENTO RAPIDO
Inicio/Home	Coloca el cursor al comienzo de la línea
Fin/End	Coloca el cursor al final de la línea
RePág/PgUp	Coloca el cursor en la primera posición de la pantalla
AvPág/PgDn	Coloca el cursor en la última posición de la pantalla
[Control/Ctrl][→]	Mueve el cursor a través de los campos definidos hacia adelante
[Control/Ctrl][←]	Mueve el cursor a través de los campos definidos hacia atrás

Tabla 3. Teclas disponibles para el movimiento del cursor

DESCRIPCION DEL TECLADO DE FUNCIONES

El teclado generalmente contiene una sección donde se encuentran las teclas con las etiquetas F1, F2, ..., F10. A esta sección se le conoce comúnmente como el Teclado de Funciones.

En el EDP tanto el teclado de funciones de la parte baja como algunos de la parte extendida se encuentran asociadas a un mandato específico. Para seleccionar una función del teclado extendido, se deben presionar simultaneamente las teclas [Cambio/Shift], [Control/Ctrl] o [Alt] y la función deseada. Las teclas para seleccionar la parte extendida se simbolizan como s/, c/^ y a, respectivamente.

En la siguiente tabla se muestran las teclas de función definidas y el mandato asociado a cada una de ellas.

TECLA	MANDATO
F1	Muestra el menú de ayuda

TECLA	MANDATO
F2	Despliega la linea de estado
F3	Define una VENTANA
F4	Define una CAJA
F5	Define un CAMPO
F6	Inserta una LINEA en blanco
F7	Salva la PANTALLA
F8	Genera Código
F9	Inicializa el editor
F10	Abandona el EDITOR
sF4	Desplaza una CAJA
sF5	Desplaza un CAMPO
sF7	Recupera una PANTALLA
^F3	Elimina una VENTANA
^F4	Elimina una CAJA
^F5	Elimina un CAMPO
^F6	Elimina una LINEA
aF5	Modifica un CAMPO

Tabla 4. Teclado de funciones

MENU DE AYUDA

El menú de ayuda permite consultar los mandatos del programa y la tecla de función asociada a cada uno de ellos.

El mandato se especifica pulsando la tecla **[F1]**. Cuando esto sucede, se despliega un menú en la parte superior de la pantalla que contiene varias opciones que permiten mostrar los mandatos en del **EDP** en forma agrupada como se muestra en la siguiente figura.

Figura 14. Opciones del menú de ayuda

Para mover el cursor entre las opciones se usan las teclas [-] o []. Para seleccionar una, se debe pulsar la tecla [Intro/Enter/Return]. Otra manera de hacer la selección en forma rápida consiste en pulsar la letra de cada opción que se encuentra en mayúscula ya sea desde el menú o estando dentro de alguna opción.

Los mandatos agrupados en cada opción se muestran en la siguiente tabla.

Archivos	Campos	Edición	Marcos	Ventanas	Otros
Salvar F7	Definir F5	Insertar Ins	Definir F4	Definir F3	Ayuda F1
Recuperar sF7	Mover sF5	Ell car Del	Mover sF4	Borrar ^F3	Estado F2
	Borrar ^F5	Ell car← BkSp	Borrar ^F4		Generar C. F8
	Cambiar sF5	Ins Linea F6			Inicializar F9
		Ell Linea sF6			Salir F10

Tabla 5. Agrupación de mandatos del programa

Durante la operación del mandato se utiliza la tecla [Esc] para salir de una opción y la tecla [F1] para abandonar el menú.

2.3.2 Edición de texto

La edición de texto permite insertar, borrar y reemplazar caracteres en la pantalla y borrar e insertar lineas.

Para llevar a cabo estas operaciones y al mismo tiempo proteger la definición de campos y cajas sobre la pantalla, se han establecido las siguientes reglas:

Durante la inserción y borrado de caracteres:

- La operación no se lleva a cabo si el cursor se encuentra sobre el marco de una caja o dentro del área para lectura de un campo.
- Sólo puede haber desplazamiento de caracteres mientras no se llegue a:

Los límites de la pantalla

El marco de una caja

El límite de un campo de lectura

Durante la inserción y borrado de lineas:

- No se puede insertar una linea en el último renglón de la pantalla
- No se puede editar una línea si el cursor esta sobre un marco
- Una línea dentro del *editor* se define como el área de un renglón de la pantalla que se encuentra entre:

Los límites de la pantalla

El marco de una caja o dos marcos

Cualquier combinación de éstos

- Durante la inserción sólo puede haber desplazamiento de lineas hasta llegar a:

El límite inferior de la pantalla

El marco de una caja

- Si al insertar una línea, en las líneas que es necesario desplazar estuviera definido un campo, el campo será desplazado y su posición será actualizada por el *editor*.
- No se puede eliminar una linea que contiene campos de lectura.

La escritura de caracteres se lleva a cabo a partir de dos modos de operación: modo de reemplazo o modo de inserción.

Para seleccionar el modo se utiliza la tecla [Ins] en forma de switch. El modo por omisión es el reemplazo.

Las teclas que se usan para borrar caracteres son: [Supr/Del] para borrar hacia adelante y [Retro/BackSpace] para borrar hacia atrás.

La edición de líneas se especifica a través de dos mandatos: [F6] para insertar una línea en blanco y [^F6] para eliminar una línea.

2.3.3 Edición de ventanas

Algunos sistemas cuentan con la posibilidad de manejar ventanas para obtener y desplegar información sobre la pantalla. En el *editor* se puede definir una ventana de tal manera que la definición de un formato que contenida dentro de ella.

Para permitir que cada formato que se define dentro del *editor* pueda ser utilizado de manera independiente y para simplificar la operación del programa, sólo se permite definir una ventana.

Los atributos que definen a una ventana se tomaron principalmente de la librería de funciones **Vitamin C**. Por esta razón se recomienda utilizar esta opción cuando el formato que se vaya a definir se apoye en el uso de esta librería.

La edición de ventanas tiene definidas dos operaciones: agregar y eliminar la definición de una ventana.

Agregar

Antes de agregar una ventana, se sugiere colocar el cursor en un lugar apropiado, ya que el *editor* permite dibujarla a partir de la posición del cursor al momento de especificar el mandato.

Durante el tiempo que en que se lleva a cabo la edición de una ventana se dispone de la tecla [Esc] para cancelar el proceso.

Para agregar o definir una ventana, se debe pulsar la tecla [F3].

Al momento de ejecutar el mandato, el *editor* verifica si ya se definió una ventana previamente. Si es así, se despliega un mensaje en la pantalla para advertirnos de este hecho y se cancela la operación del mandato. De lo contrario, se debe seguir la secuencia que se describe:

Usando las teclas para movimiento del cursor:

1. Dibujar el marco de la ventana y pulse [F3] para terminar.

2. Colocar el marco en el lugar deseado y pulsar [F3]

3. Seleccionar el tipo de marco y pulsar [F3].

4. Seleccionar el tipo de efecto deseado para abrir la ventana y pulsar [F3].

posteriormente se debe,

5. Escribir un título para la ventana y pulsar [F3].

6. Seleccionar las características de operación y pulsar [F3].

Cuando se define una ventana la posición del cursor en la pantalla es referida con respecto a las coordenadas de ésta. Esto significa que a partir de ese momento cualquier definición posterior quedará dentro de la ventana.

Eliminar

Para eliminar la definición de una ventana se deben pulsar simultaneamente las teclas [^F3].

En ese momento se despliega el siguiente mensaje en la pantalla:

```
Pulse [s] [n] o [Esc]
Desea eliminar la ventana (s/n)? _
```

Con esta acción previa se evita eliminar una ventana en forma involuntaria. Si se pulsa la tecla [s], se elimina la definición y el marco de la ventana se borra de la pantalla.

A partir de ese momento el cursor se puede colocar nuevamente en cualquier posición de la pantalla.

2.3.4 Edición de cajas

La edición de cajas es una característica que puede ser útil para enmarcar parte del contenido de la pantalla y atraer la atención del operador hacia algún lugar específico de ella.

Para la edición de marcos se dispone de tres operaciones: agregar, desplazar y eliminar la definición de una caja.

Durante el tiempo que en que se lleva a cabo la edición de un marco se dispone de la tecla [Esc] para cancelar el proceso.

La edición de cajas permite que el marco de una o varias cajas queden encimados entre sí, sin embargo, no se puede colocar un marco sobre el área de un campo o sobre algún carácter o línea de texto.

Antes de especificar el mandato para desplazar o eliminar una caja, es necesario seleccionarla colocando el cursor dentro del marco de la que se desea afectar.

Agregar

Antes de agregar una caja, es recomendable colocar el cursor en un lugar apropiado ya que el *editor* permite dibujarla a partir de la posición del cursor al momento de especificar el mandato.

Para agregar o definir una caja, se debe pulsar la tecla [F4].

Si después se desea cancelar la operación del mandato, se debe pulsar la tecla [Esc].

La secuencia que se debe seguir es la siguiente:

Usando las teclas para movimiento del cursor:

1. Dibujar el marco de la caja y pulsar [F4].

2. Colocar el marco en el lugar deseado y pulsar [F4].

3. Seleccionar el tipo de marco y pulsar [F4].

Desplazar

Para desplazar el marco de una caja se deben pulsar simultaneamente las teclas [F4] y hacer lo siguiente:

Usando las teclas para movimiento del cursor se debe colocar el marco en el lugar deseado y pulsar las teclas [F4] nuevamente.

Eliminar

Para eliminar la definición de una caja se deben pulsar simultaneamente las teclas [^F4].

En ese momento se despliega el siguiente mensaje en la pantalla:

```
Pulse [s] [n] o [Esc]
Desea eliminar la caja (s/n)? _
```

Con esta acción previa se evita eliminar una caja en forma involuntaria. Si se pulsa la tecla [s], se elimina la definición y el marco de la caja se borra de la pantalla.

2.3.5 Edición de campos

La posibilidad de definir campos para lectura de datos, es una de las características principales del *editor*. Se pueden definir tantos campos como sean necesarios en cualquier lugar de la pantalla a través de un procedimiento rápido y sencillo.

Durante el tiempo que en que se lleva a cabo la edición de un campo se dispone de la tecla [Esc] para cancelar el proceso.

Antes de especificar el mandato para desplazar, modificar o eliminar un campo, es necesario seleccionarlo colocando el cursor dentro del área de lectura del que se desea afectar.

La edición de campos contempla cuatro operaciones: agregar, desplazar, modificar y eliminar la definición de un campo:

Agregar

Para definir un campo se debe pulsar la tecla [F5] y seguir la siguiente secuencia:

1. Llenar el formato de captura que se presenta en la pantalla con los atributos del campo y pulsar la tecla [F5].

2. Con las teclas para movimiento del cursor colocar el campo en la posición deseada y pulsar [F5].

Las teclas disponibles para llenar el formato se muestran en la tabla 2 y en la siguiente tabla se describen los atributos de un campo.

ATRIBUTO	ESPECIFICACIONES
nombre	Nombre de la variable donde se va a depositar el dato que se lea en el campo.
tipo	Tipo de la variable que contiene el campo. Consulte la tabla 7 para los tipos válidos.
longitud	Tamaño del campo en caracteres (longitud máxima 70)
formato	Tipo de cada uno de los carácter del campo. Depende del sistema donde se va a leer el campo.

Tabla 6. Atributos de un campo

TIPO	DESCRIPCION
C	Carácter
L	Lógico
S	Cuerda de caracteres (tipo por omisión)
E	Numero entero
D	Número real

Tabla 7. Tipos de campo disponibles

Desplazar

Para desplazar un campo se deben pulsar simultaneamente las teclas [F5] y hacer lo siguiente:

Usando las teclas para movimiento del cursor se debe colocar el marco en el lugar deseado y pulsar las teclas [F5] nuevamente.

Para modificar la definición de un campo se deben pulsar simultaneamente las teclas [aF5] y hacer lo siguiente:

1. Modificar los atributos del campo que se presentan en un formato sobre la pantalla y pulsar las teclas [aF5].

2. Si se modifica el tamaño, el programa ajusta el campo y permite desplazarlo a otro lugar. Si es necesario coloque el campo en su nueva posición y pulse las teclas [aF5] nuevamente.

En ese momento el programa hace los ajustes el campo de acuerdo a los atributos modificados.

Eliminar

Para eliminar la definición de una campo se deben pulsar simultaneamente las teclas [^F5].

En ese momento se despliega el siguiente mensaje en la pantalla:

```
Pulse [s] [n] o [Esc]
Desea eliminar el campo (s/n)? _
```

Con esta acción previa se evita eliminar un campo en forma involuntaria. Si se pulsa la tecla [s], se elimina la definición y el área de lectura del campo se borra de la pantalla.

Una vez que el campo ha sido eliminado no existe forma de recuperarlo, por esta razón se recomienda tener mucho cuidado con el uso de este mandato.

2.3.6 Manejo de archivos

El manejo de archivos permite salvar el contenido de la pantalla y recuperarlo en otro momento.

Estas opciones resultan útiles para proteger la definición de un formato contra la interrupción inesperada de energía eléctrica o para conservar la definición y volver a usarla posteriormente.

Al ejecutar uno de estos mandatos el usuario debe proporcionar el nombre de un archivo. Para realizar la operación en forma rápida, el programa propone al usuario el nombre del archivo más recientemente utilizado.

Antes de grabar el contenido de la pantalla en un archivo, se escribe al principio de éste una marca para poder identificar después que el formato corresponde con el que usa el EDP. De esta manera se evita tratar de leer archivos que no fueron creados dentro del *editor*.

Salvar

Para salvar la definición de un formato en un archivo, se deben seguir los siguientes pasos:

1. Pulsar la tecla [F7].

2. Proporcionar el nombre del archivo donde se desea salvar el contenido del formato definido.

3. Pulsar la tecla [Intro/Enter/Return] para grabar el archivo en el disco o pulsar la tecla [Esc] para cancelar la operación.

Si al intentar grabar el formato se encuentra que el archivo ya esta creado, se despliega el siguiente mensaje en la pantalla:

```
Pulse [s] [n] o [Esc]
El archivo existe. Desea actualizarlo (s/n) ? _
```

Si la respuesta es afirmativa, el contenido del archivo se actualiza. De lo contrario, la operación será cancelada.

Recuperar

Para recuperar en la pantalla un archivo que contiene un formato definido anteriormente en el EDP se deben seguir los siguientes pasos:

1. Pulsar simultaneamente las teclas [sF7].

Si el contenido de la pantalla ha sido modificado y no se ha salvado, se despliega el siguiente mensaje en la pantalla:

```
Pulse [s] [n] o [Esc]
Los cambios no se han grabado. Continuar (s/n) ? _
```

Si la respuesta es afirmativa, el contenido del archivo se actualiza. De lo contrario, la operación será cancelada.

Si la repuesta es afirmativa se procede a cargar el archivo en la pantalla perdiéndose definitivamente los cambios efectuados al formato anterior. En caso contrario, el programa regresa el control al *editor* para que la pantalla pueda ser salvada antes de cargar el archivo.

Si el archivo especificado no corresponde a la definición de un formato creado dentro del EDP, éste no podrá ser cargado.

2. Proporcionar el nombre del archivo donde se desea salvar el contenido del formato.

3. Pulsar la tecla [Intro/Enter/Return] para grabar el archivo en el disco o pulsar la tecla [Esc] para cancelar la operación.

Otra forma de recuperar un formato consiste en especificar el nombre del archivo como parámetro al momento de ejecutar el **EDP.**

2.3.7 Línea de estado

La línea de estado es un renglón de la pantalla donde se despliega alguna información que puede ser de utilidad para el usuario.

El control de la línea de estado funciona en forma de switch (encendido/apagado). Para desplegar o no su contenido en la pantalla se debe pulsar la tecla [F2].

Esta línea se puede desplegar en la parte inferior o en la parte superior de la pantalla. Si el cursor se trata de colocar sobre la línea donde se despliega el estado, el programa cambia su posición. Esta característica de la línea de estado permite usar toda la pantalla para definir un formato.

La información que se puede consultar en la línea de estado se muestra a continuación:

y se interpreta de la siguiente forma:

- Si el contenido de la pantalla ha sido modificado y no se ha salvado en un archivo, se despliega un asterisco (*).
- El nombre del archivo que contiene el formato definido en la pantalla. Si se encuentra ausente, significa que la definición no ha sido salvada.
- VENT: SI/NO se utiliza para indicar si la definición está contenida o no dentro de una ventana definida por el usuario.
- Si el cursor se mueve sobre el marco de una caja o se encuentra dentro del área definida por ésta, se despliega la palabra CAJA con su número correspondiente.
- Cuando el cursor se mueve dentro de un campo de lectura, se despliega el nombre con el que esta definido.
- Se muestra cuando se esta usando el modo de inserción para la edición de caracteres. Si esta ausente, significa que se esta usando el modo de reemplazo.
- La tecla asociada con el menú de ayuda.

Cuando comienza la ejecución del programa el *editor* presenta la línea de estado encendida.

2.3.8 Uso del generador de código

El *generador* de código se encarga de generar un programa que permite usar el formato definido en la pantalla del *editor*.

Tiene integrados dos módulos para seleccionar el tipo del programa de salida que se desea obtener.

Estos módulos son: uno para generar programas en lenguaje *C* y otro para generar programas en lenguaje de comandos dBASE III.

Cada módulo cuenta con las siguientes opciones para generar el programa de salida:

- Se puede seleccionar el orden en que se desean leer los campos definidos. El orden puede ser tomado con respecto a su posición dentro de la pantalla o con respecto a la secuencia en que se definieron.
- Es posible crear un archivo con la declaración de las variables que se usan para leer los campos definidos.
- Se pueden incluir en el mismo archivo tres tipos de formatos: para capturar, para modificar los datos de un registro que ya existe y para consultar los datos de un registro que ya existe.

Para ejecutar el *generador* de código, se debe pulsar la tecla [F8]. Cuando se pulsa esta tecla, se presenta en la pantalla el formato que se muestra en la figura 15.

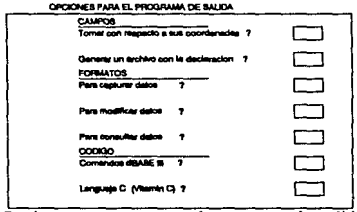


Figura 15. Opciones para generar el programa de salida

Para llenar el formato se pueden usar las teclas que se muestran en la Tabla 2. La manera en que se deben seleccionar las opciones es la siguiente:

1. Se deben marcar cada una de las opciones que se desean escribiendo una equis *X* y pulsar la tecla [Intro/Enter/Return]. Si no se desea elegir una opción sólo se debe pulsar la tecla [Intro/Enter/Return].

El nombre del archivo que contiene el programa de salida se forma a partir del nombre del archivo que contiene la definición del formato. Si el formato no tiene asignado un archivo, se despliega un mensaje en la pantalla solicitando el nombre de un archivo para el programa de salida. Si este es el caso, proporcione el nombre del archivo y pulse la tecla [Intro/Enter/Return].

En la siguiente tabla se muestra las características del archivo de salida creado por el *generador* de código.

Opción	Lenguaje C		dBase III	
	Salida	Nombre	Salida	Nombre
Declaración	*header*	formato.h	archivo	formato.mem
Programa	archivo	formato.c	archivo	formato.prg
Captura	función	Aformato()	procedure	Aformato
Modificación	función	Mformato()	procedure	Mformato
Consulta	función	Cformato()	procedure	Cformato

Tabla 8. Características del archivo de salida

Al terminar de ejecutarse el *generador* de código se regresa el control del programa al *editor*.

2.3.9 Inicializar y abandonar el programa

INICIALIZACION EL PROGRAMA

El mandato para inicializar el programa permite definir más de un formato durante una misma corrida del programa. Para ejecutar este mandato se debe pulsar la tecla [F9].

Cuando se especifica este mandato, se limpia la pantalla y todas las estructuras de datos que mantiene el programa regresan a sus valores iniciales.

Por esta razón el programa solicita que la operación sea verificada por el usuario antes de realizarla. Para esto se despliega el siguiente mensaje en la pantalla:

```
¡Cuidado! Después de realizar esta operación el programa será inicializado
          y se perderá totalmente el contenido de la pantalla.
Pulse [s] [n] o [Esc]
Desea continuar (s/n) ? _
```

Esta acción protege al usuario contra la especificación involuntaria de este mandato. Si la respuesta es afirmativa se inicializa el programa, de lo contrario, se cancela su ejecución.

ABANDONAR EL PROGRAMA

Para abandonar la ejecución del programa se puede pulsar la tecla [F10] o la tecla [Esc].

Antes de terminar la ejecución del programa se lleva a cabo lo siguiente:

1.- Si la pantalla no se ha salvado se despliega el siguiente mensaje:

```
Pulse [s] [n] o [Esc]
La pantalla no se ha salvado. Desea salvarla ? _(s/n)? _
```

- Para salir sin salvar la pantalla se debe pulsar la tecla [n].
- Si desea cancelar la ejecución del mandato se debe pulsar la tecla [Esc].
- Para salvar la pantalla antes de salir, los pasos a seguir son los siguientes:

Pulsar la tecla [s]. El programa solícita el nombre de un archivo proponiendo el más recientemente proporcionado. Pulsar la tecla [Intro/Enter/Return] para aceptar el nombre que se propone o escribir el nombre del archivo donde se desea salvar el contenido de la pantalla y pulsar la tecla [Intro/Enter/Return].

2. Si la pantalla ya se encuentra salvada, se despliega el siguiente mensaje solicitando que la operación sea confirmada:

```
Pulse [s] [n] o [Esc]
Desea abandonar el programa (s/n)? _
```

Para abandonar el programa se debe pulsar la tecla [s]. En caso contrario, se puede pulsar cualquier tecla y la operación será cancelada.

Antes de abandonar el programa, la información que se encontraba en la pantalla antes de ejecutarlo se restaura.

2.4 Implantación

La implantación del *editor* se llevo a cabo en una computadora personal compatible (*PC*). No obstante, está contemplada la posibilidad de introducir el programa en otro tipo de equipos de cómputo como sistemas **UNIX**. Esto fué posible gracias al uso de las herramientas empleadas que cuentan con versiones desarrolladas para trabajar en este tipo de equipos.

MODULOS

El programa esta compuesto por varios módulos, cada uno de éstos contiene un conjunto de funciones que cumplen con una tarea específica y se encuentran separados en varios archivos.

Algunos módulos funcionan en forma independiente y son de uso general o compartido dentro del programa. En la siguiente tabla se listan los módulos que forman parte del programa y el nombre de los archivos donde se encuentran.

Módulo	Archivo	Objetivo/Contenido
Memoria	**edpmem.c**	**Reservar memoria para las estructuras de datos y variables globales**
Mensajes	**edpmsg.c**	**Manejar los mensajes del programa**
Ayuda	**edpayuda.c**	**Manejar el menú de ayuda**
Edición de texto	**edpedit.c**	**Editar texto**
Edición de ventanas	**edpvent.c**	**Editar ventanas**
Edición de campos	**edpcampo.c**	**Editar campos**
Edición de cajas	**edpcajas.c**	**Editar cajas**
Funciones generales	**edpmisc.c**	**Proporcionar las funciones de uso general**
Archivos	**edpfiles.c**	**Proporcionar la interface de entrada/salida del programa**
Principal	**edp.c**	**Controlar la ejecución del programa**
Estado	**edpedo.c**	**Controlar el despliegue de la línea de estado**
Generador de código	**edpcode.c**	**controlar las opciones para generar código**
Lenguaje *C*	**edpcdvc.c**	**Generar programas para Vitamin C**
Comandos dBASEIII	**edpcddb.c**	**Generar programas para dBASE III**
	edp.h	**Definir los valores constantes del programa**

Módulo	Archivo	Objetivo/Contenido
	edpext.h	Declarar las variables globales
	edpdef.h	Declarar las estructuras de datos del programa
	edpfunc.h	Declarar las funciones del programa y sus parámetros
	makefile	Archivo de especificaciones para la utilería de mantenimiento

Tabla 9. Archivos del sistema

La organización del *editor* a través de módulos proporciona varias ventajas que repercuten en el desarrollo del programa. Estas ventajas son:

- Es más fácil trabajar con un programa dividido en varios archivos pequeños, que con un programa contenido en un archivo más grande.
- Durante la compilación del programa, se reparte la carga del compilador porque no tiene que considerar todo el sistema cuando solo se esta tratando una parte del mismo.
- Cuando se detecta alguna falla o error es relativamente más fácil aislar y encontrar el problema.

Por estas razones, el tiempo que se tiene que emplear en la implantación y el desarrollo del programa se reduce en forma considerable con respecto a la opción de emplear solo un archivo para el programa.

LENGUAJE

El programa está escrito en lenguaje *C* y las razones por las que se escogió son:

- es un lenguaje sencillo
- se genera un código pequeño y eficiente
- los programas se ejecutan con rapidez
- se pueden escribir programas *portables* con relativa facilidad
- muchos sistemas y Base de Datos están escritos en este lenguaje
- se dispone de una gran variedad de herramientas para usarse con este lenguaje

COMPILADOR

Para compilar cada uno de los módulos del programa se empleó la versión **5.1** del compilador de *C* de **Microsoft Co.**

Algunas de las ventajas que se pueden señalar sobre el uso de este compilador son:

- La librería de funciones que tiene es muy completa
- Proporciona un ambiente de programación que permite reducir el tiempo de desarrollo.
- Genera código que puede ser "portado" a otros compiladores o equipos
- El programa generado se ejecuta con rapidez
- Dispone de varias utilerías para mantenimiento de programas

Para compilar el programa se uso el modelo de memoria largo que permite direccionar toda la memoria (1 M) de la computadora para el código y los datos del programa.

LIGADOR

Para ligar los módulos del programa y las librería utilizadas se escogió la versión 2.24 del ligador PLink86 Plus de Phoenix Technologies Ltd.

Este ligador genera un archivo ejecutable de menor tamaño que otros ligadores y contempla la posibilidad de manejar **overlay's.**

El manejo de **overlay's** es una técnica que se utiliza para obtener un ahorro en la memoria que se necesita para procesar un programa.

Para utilizar esta técnica en forma eficiente, se requiere que el diseño del programa este integrado por módulos que funcionen de manera independiente o en su defecto separarlo en módulos de uso general y módulos independientes.

De esta forma, el manejo de **overlay's** se encarga de mantener en la memoria de la computadora los módulos compartidos y un sólo un módulo independiente a la vez, según lo vaya requiriendo la ejecución del programa.

Esta fue también una de las razones por las que se decidió dividir el **EDP** en módulos.

La forma en que éstos se encuentran organizados se muestran en la siguiente tabla.

Módulos de uso general.

- Principal
- Memoria
- Mensajes
- Estado
- Funciones generales

Módulos independientes.

- Ayuda
- Edición de texto
- Edición de ventanas
- Edición de cajas
- Edición de campos
- Manejo de archivos
- Generador de código
- Módulo para **Vitamin C**
- Módulo para **dBASE III**

Tabla 10. Organización de los módulos del **EDP.**

Las especificaciones para llevar a cabo el proceso de ligado se encuentran en el archivo "edp_plnk.lnk" que se proporciona como entrada al ligador.

LIBRERIAS

El **EDP** se apoyó en la librería de funciones **Vitamin C** de Creative Programming Consultants Inc, para realizar el manejo y control de la pantalla.

Con el uso de esta librería fué posible enfocar la atención sólo en el **EDP** ya que no hubo necesidad de desarrollar adicionalmente una librería de funciones que se encarga de efectuar ese trabajo.

A través de las funciones de la **Vitamin C**, el *editor* controla las operaciones relacionadas con: el cursor, la definición de ventanas y cajas, la captura de datos, el manejo de colores y el despliegue de mensajes en la pantalla.

Una de las características de la librería es que dispone de versiones para ser usadas en equipos con sistema operativo **UNIX.** Este hecho permite contemplar la posibilidad de transportar el programa a equipos con esas características.

UTILERIAS

Una de las tareas más pesadas que se deriva de la separación de un programa en varios archivos fuente, es la que se encarga de decidir que archivos se tienen que actualizar cada vez que se modifica una parte del programa.

Para realizar este trabajo, la versión 5.1 del compilador de **Microsoft C** contempla un programa de utilería que se llama **MAKE.**

Esta utilería se encarga del mantenimiento de programas a partir de un archivo de especificaciones. Este contiene una relación de los archivos de un sistema o programa que dependen entre sí y el procedimiento que se debe seguir para actualizarlos cada vez que se modifica alguno de ellos.

Para realizar su trabajo, el programa **MAKE** toma la fecha y la hora en que un archivo fué modificado por última vez y llama a los procesos que son necesarios para actualizar los archivos que resulten afectados.

El **EDP** cuenta con un archivo de especificaciones llamado "makefile" para llevar a cabo el mantenimiento del programa a través del uso de la utilería **MAKE.**

Todas las funciones que forman parte del programa, están declaradas en el archivo "edpfunc.h". La declaración de cada función contiene el tipo de valor que regresa la función y el tipo de cada uno de los parámetros que recibe. Estas declaraciones se incluyen como "header" en cada módulo que forma parte del programa.

Este procedimiento resulta útil durante el desarrollo de un programa, ya que es una manera de garantizar el uso adecuado de cada función, disminuyendo la probabilidad de que se presente alguna falla durante la ejecución del programa, debido a una causa de este tipo. Este archivo se genera con una opción de la versión 5.1 del compilador de **Microsoft C.**

Para usar esta utilería, es necesario proporcionar como parámetro todos los archivos que se deben analizar para obtener la declaración de las funciones.

Capítulo 3 Generador de Codigo

El *generador* de código es una de las partes principales del **EDP**. Su función consiste en generar un programa de salida que permita usar un formato diseñado en la pantalla.

Dentro de los propósitos de este trabajo solo se contemplaron dos módulos. Uno de ellos se encarga de generar un programa en lenguaje *C* para aplicarse en la librería de funciones **Vitamin C**. El otro genera un programa que se puede aplicar en el intérprete de comandos de **dBASE III**.

Para cada salida que se desea generar debe integrársele un módulo especial. El trabajo que debe realizar cada módulo consiste en generar un programa basándose en el descriptor de pantallas que construye el *editor*. En este programa se deben especificar las instrucciones necesarias para usar el formato en una aplicación en particular.

El *generador* de código puede ser llamado desde el **EDP** en cualquier momento durante la ejecución del programa. Cuando ocurre este hecho se presenta una lista de los módulos de salida que están integrados al *generador* para que se seleccione el que corresponda con el tipo de programa que se desea generar.

En este capítulo se describe el funcionamiento de cada uno de los módulos desarrollados para generar el programa de salida.

MODULO PARA VITAMIN C

Las opciones para generar el programa de salida que tiene definidas éste módulo se presentan en la pantalla a través de un formato de captura.

La primera opción permite especificar el orden en que se desean leer los campos definidos. Esto significa que los campos se pueden leer tomando como referencia el orden que guardan con respecto a las coor-

denadas de la pantalla o respetando la secuencia en que fueron definidos en la pantalla.

La segunda opción se usa para especificar si se desea crear un archivo que contenga la declaración de las variables usadas para leer cada campo definido.

El formato definido puede usarse de tres maneras diferentes que son: capturar, modificar o consultar.

El procedimiento de captura por medio de un formato, en algunas ocasiones es similar al procedimiento de recuperación o actualización de datos. La diferencia consiste en que en el primero, los datos se obtienen por primera vez y en el otro se toman de un registro en la memoria y se presentan en la pantalla para permitir que sean actualizados o consultados.

Por esta razón, en el *Generador de Código* se contempla la posibilidad de que se desee ocupar el mismo formato para los dos procedimientos mencionados.

De esta manera, las siguientes tres opciones permiten seleccionar el uso que se le desea dar al formato definido. Estas opciones son: formato para capturar, formato para modificar, formato para consultar.

Es posible seleccionar más de uno para ser incluido en el mismo programa de salida. Si no se elige ninguno se selecciona el formato de captura por omisión.

El nombre del archivo de salida y las funciones o subprogramas que contiene éste, se forman tomando como base el nombre del formato. Las características del archivo de salida pueden ser consultadas en la Tabla 8. Si el formato no tiene asignado un archivo, se despliega un mensaje en la pantalla solicitando el nombre del archivo.

Las opciones seleccionadas se indican en el registro de banderas del **EDP** en cada uno de los bits reservados para este propósito.

El procedimiento que sigue este módulo para generar el programa de salida consiste en escribir en un archivo las instrucciones en lenguaje *C* para obtener el formato definido de acuerdo a las opciones seleccionadas por el usuario.

Si se selecciona la opción para declarar los campos definidos, se toma de la tabla de campos cada uno de ellos agrupándolos con respecto

a su tipo para formar las declaraciones a partir del nombre del campo, tipo y longitud; posteriormente se crea el archivo correspondiente y se escriben las declaraciones en éste.

Para saber el orden en que deben ser tomados los campos definidos en la tabla, se forma un vector que contiene los índices de los campos ordenados.

Este vector se necesita debido a que si se ordenara la tabla de campos, sería necesario actualizar los índices que guardan la referencia de los campos tanto en la pantalla como en su IMAGEN. Esto no sería conveniente porque el proceso emplearía mucho tiempo y el cambio en el orden de los campos sería permanente. Si no se elige la opción para ordenar los campos, el vector se forma con la secuencia indicada por la tabla.

La librería de funciones **Vitamin C** sólo utiliza variables de tipo *string* o cuerdas de caracteres para leer un campo de la pantalla. Debido a esto, se forma una lista con las variables que se deben usar para leer los campos de la pantalla. Para los campos que nos son de tipo *string* se declaran variables temporales de éste tipo y se incluyen en la lista junto con las variables de los campos tipo *string*. Paralelamente se crea otro vector que contiene los índices de la lista que corresponden a las variables temporales.

Para cada tipo de formato, se usa una función diferente. El procedimiento que llevan a cabo es muy similar, sólo difieren en que escriben las instrucciones que van de acuerdo con el tipo de formato en cuestión.

El cuerpo de cada función contiene un bloque de declaraciones. En éste se incluye lo siguiente:

- Si el tipo de formato es para capturar o modificar se declaran las variables temporales y una variable que se utiliza para que la función regrese un valor
- Se declara una variable para manejar la ventana que contiene el formato

Se toman los atributos del registro de ventanas y se escribe la instrucción que se usa para presentarla en la pantalla. Si no se definió la ventana, se define una por omisión con los siguientes atributos: del tamaño de la pantalla, sin marco, sin título, activa y con cursor.

Si hay cajas definidas, se toman de la tabla y se escriben las instrucciones que permiten dibujar cada caja de acuerdo a los atributos seleccionados.

Se hace un barrido completo de la matriz para obtener, en forma de cadenas, los caracteres de texto que contiene. Después se escriben las instrucciones para desplegar los caracteres de texto en la pantalla. desplegar mensajes de texto en la pantalla.

Para leer campos de un formato en la pantalla, la librería **Vitamin C** se apoya en dos funciones: una para controlar el proceso de captura que toma de una tabla los campos que se tienen que leer y sus atributos, y otra para llenar esa tabla.

De esta manera, si hay campos definidos se sigue un proceso diferente para cada tipo de formato.

Para capturar se escriben las instrucciones para hacer lo siguiente:

- llenar la tabla de lectura de campos a partir de la lista de cadenas y sus atributos
- dejar el control de la pantalla a la función de lectura y tomar el valor que ésta regresa
- cerrar la ventana
- verificar el valor regresado por la función de lectura
- terminar la ejecución de la función y regresar el valor adecuado
- asignar el contenido de cada una de las variables temporales a sus campos correspondientes

Para modificar se sigue un proceso similar, la diferencia se encuentra en que para este caso, los campos ya contienen datos, y antes de llenar la tabla de lectura se deben incluir las instrucciones para asignar el contenido de cada uno de los campos que no son de tipo *string*, a las variables temporales.

Para consultar, se usan dos funciones diferentes para desplegar los datos que contienen cada uno de los campos definidos. Una se usa para desplegar los campos de tipo carácter o numérico ya sea entero o decimal, y la otra se usa para desplegar los campos que contienen cadenas de caracteres.

Para este tipo de formato se escriben las instrucciones para:

- desplegar cada uno de los campos de la tabla de acuerdo a su tipo
- cerrar la ventana
- hacer una pausa para visualizar la información en la pantalla

MODULO PARA DBASE III

Las opciones para generar el programa de salida que tiene definidas éste módulo se presentan en la pantalla a través de un formato de captura. Estas opciones son las mismas que contempla el módulo para **Vitamin C.**

El procedimiento para generar el programa de salida en lenguaje de comandos de **dBASE III**, es muy similar al que se sigue en el módulo para **Vitamin C.** A continuación se señalan las diferencias con respecto a ese módulo.

Los nombres de los archivos de salida tanto para el programa como para las declaraciones se forman con las extensiones que identifica **dBASE III** para archivos de ese tipo. Debido a que **dBASE III** no contempla la definición de cajas y ventanas, éstas se sustituyen por una serie de comandos que permiten dibujar el tipo de marco seleccionado para cada una de ellas.

En éste módulo no es necesario utilizar variables temporales para leer los campos de la pantalla ya que la función que controla el proceso de captura, puede leer cualquier tipo de dato permitido en **dBASE III.**

En el formato de captura, se escribe un comando para indicar que los campos que se van a leer a continuación pasaran a formar parte de un registro nuevo en la Base de Datos.

Hasta aquí queda descrita la forma en que se genera el código de salida del programa.

Una característica importante que posee el programa generado es que se encuentra indentado de tal manera que resulte fácil comprender la función que desempeña. El programa generado se encuentra libre de errores y queda listo para ser procesado por algún compilador de **lenguaje C** o el intérprete de comandos **dBASE III**, según el caso, y así obtener el código que permite usar el formato definido durante una aplicación real.

Para integrar otros módulos al *Generador de Código* es necesario desarrollar cada uno de ellos en forma similar a los que ya se integraron.

Una vez desarrollados, deben ser dados de alta en el *generador* de código y contemplarlos dentro del formato de captura que contiene las opciones del código. Este programa se encuentra en el archivo *edpcode.c*.

Los archivos que contengan los módulos que se quieran integrar deben ser incluidos en el archivo de especificaciones que usa la utilería **MAKE** y en el archivo de especificaciones que se proporciona al ligador. Posteriormente se debe ejecutar el programa **MAKE** para crear una nueva versión del programa.

Capítulo 4. Biblioteca de Funciones

El *Generador de Código* considera la posibilidad de generar programas en lenguaje C que tienen como función leer datos de la pantalla a través de un formato definido con el *editor*.

Para realizar sus objetivo, el programa se apoya en algunas funciones que contiene la librería **Vitamin C.**

Sin embargo, cabe la posibilidad de que no todos los usuarios del programa dispongan de esta librería; debido a esto, el propósito de este capítulo consiste en presentar un pequeño *Manual de Referencia* con las funciones que son llamadas dentro del programa que se encuentran en esa librería.

Las funciones que se incluyen en dicho manual, pueden servir como base para crear una **Biblioteca de Funciones** que permita sustituir la necesidad de disponer de la librería **Vitamin C.**

De esta manera se pretende abrir la posibilidad de que el EDP sea usado para definir formatos de captura o recuperación de datos enfocados hacia aplicaciones en *lenguaje C.*

Manual de Referencia de la Biblioteca de Funciones

NOMBRE

atget

SINOPSIS

```
int     atget (ren,col,campo,mascara)
int     ren;            renglón de la coordenada de lectura
```

int	**col;**	**columna de la coordenada de lectura**
char	*** campo;**	**apuntador al campo de lectura**
char	*** mascara;**	**formato de cada carácter del campo**

FUNCION

Prepara un campo para lectura en la posición especificada. La posición del campo se especifica por *ren* y *col*. Esta posición es referida con respecto a la ventana actual o, si no hay una ventana abierta, respecto a la pantalla física. Cualquier valor por omisión asignado al *campo* se despliega de acuerdo al formato especificado en *mascara*.

La lectura no toma lugar al momento de llamar a la función. La información realacionada con el campo se deposita en una tabla de *gets*. Las llamadas subsecuentes a esta función, agregan más entradas a esta tabla de tal manera que varios campos puedan ser procesados en grupo cuando se hace la llamada a la función *readgets ()*.

VERIFICACION

Regresa 0, si la operación fue correcta

Regresa -1, si ocurre algún error o la tabla se encuentra llena

VEASE TAMBIEN

redagets ()

NOMBRE

atsay

SINOPSIS

void	**atsay (ren,col,cuerda)**	
int	**ren;**	**renglón de la coordenada**
int	**col;**	**columna de la coordenada**
char	*** cuerda;**	**cuerda de caracteres terminada con NULL**

FUNCION

Escribe una cuerda de caracteres en la posición especificada. Las coordenadas son relativas a la ventana actual o de lo contrario se refieren a la pantalla física.

NOMBRE

atsaynum

SINOPSIS

void	atsaynum (ren,col,numero,mascara,tipo)	
int	ren;	renglón de la coordenada
int	col;	columna de la coordenada
char	* numero;	apuntador al número a desplegar
char	* mascara;	formato para desplegar el número
int	tipo;	tipo de número

FUNCION

Los valores numéricos pueden ser formateados, y desplegados en una posición específica, por medio de esta función. Para permitir a la función manejar números enteros y números reales se debe pasar el tipo del valor numérico que se va a desplegar. Los tipos válidos se definen como constantes **INTEGER** y **REAL**.

VEASE TAMBIEN

atget ()

NOMBRE

empty

SINOPSIS

void	empty (cuerda, tamano)	
char	* cuerda;	cuerda que se va inicializar
int	tamano;	número de caracteres (bytes) de la cuerda incluido el NULL

FUNCION

Esta función forma una cadena en blanco del tamaño especificado -1 seguida por el carácter **NULL**. Este tipo de cuerdas es usado durante la lectura de campos.

NOMBRE

getone

SINOPSIS

```
int     getone ()
```

FUNCION

Esta función se encarga de leer un carácter del teclado. Espera la entrada de algún carácter y regresa el valor de la tecla presionada.

NOMBRE

readgets

SINOPSIS

```
int     readgets ()
```

FUNCION

Cada llamada a la función *atget ()* guarda la información de un campo de lectura en una tabla de *gets*. La función *readgets ()* usa esta tabla para permitir la entrada de datos de a acuerdo a esa información.

La entrada toma lugar en el orden en que se depositaron los campos en la tabla, aunque el usuario tiene la libertad de moverse a través de los campos según su deseo. Cuando termina la entrada de datos, la tabla se limpia para ser usada posteriormente. Las teclas disponibles para capturar los datos pueden ser consultadas en la Tabla 2.

NOMBRE

wxopen

SINOPSIS

```
int     wxopen(sup,izq,inf,der,tit,ctr)
int     sup;        renglón de la coordenda superior
int     izq;        columna de la coordenda superior
int     inf;        renglón de la coordenda inferior
int     der;        columna de la coordenda inferior
char    * tit;      título de la ventana
int     ctrl;       byte de control para la ventana
```

FUNCION

Las coordenadas definen la posición y el tamaño físico de la ventana. El tamaño físico es definido como la parte de la ventana que es visible para el usuario. El byte de control

es un valor entero que controla diferentes opciones para la operación de la ventana. Cada opción tiene un valor constante asignado. Las opciones disponibles son las siguientes:

-BORDER

Dibuja un marco alrededor de la ventana (debe ser usada con BD1,BD2,BD3 o BD4)

-ACTIVE

Despliega la ventana en la pantalla al momento de abrirla

-CURSOR

Despliega el cursor en la ventana

-BD1

Tipo de marco DOBLE

-BD2

Tipo de marco SENCILLO

-BD3

Tipo de marco COMBINADO

-BD4

Sin marco

-NOADJ

No permite ajustar el tamaño físico de la ventana

-STANDARD

Abre la ventana de arriba hacia abajo

-CENTER

Abre la ventana del centro hacia afuera

- TLDN

Abre la ventana de la esquina superior hacia la esquina inferior

- CUSTOM

Abre la ventana con scroll de arriba hacia abajo

VERIFICACION

Regresa -1, si no fue posible abrir la ventana o, si todo estuvo correcto, regresa el número asignado a la ventana. Este número debe ser conservado para que otras funciones puedan tener acceso a la ventana

VEASE TAMBIEN

wclose ()

NOMBRE

wclose

SINOPSIS

```
int     wclose (num),
int     num;        número de la ventana a cerrar
```

FUNCION

Esta función cierra la ventana señalada por el valor de *num* y regresa toda la memoria ocupada en el sistema por ésta. Una vez que la ventana se cierra, toda la información que contiene se pierde totalmente.

La ventana física se borra de la pantalla y se restaura automáticamente la región que ocupaba.

VERIFICACION

Regresa -1, si la ventana no se encuentra abierta o se especificó un número inválido

VEASE TAMBIEN

wxopen ()

Capítulo 5. Aplicación

En este capítulo se tratará de ilustrar como se puede aplicar el **EDP** para generar un programa que permita capturar y recuperar registros para un catálogo de clientes.

Los campos que contiene el catálogo se muestran en la tabla 11:

cuerda	cuerda	cuerda	cuerda	cuerda	cuerda	real	real	real	real
clave (10)	nombre (30)	atencion (20)	direccion (30)	tel1 (9)	tel2 (9)	credito (14)	descto (5)	saldo (14)	ventas (14)

Tabla 11. Catálogo de Clientes.

el formato que se va a definir se presenta en la siguiente figura.

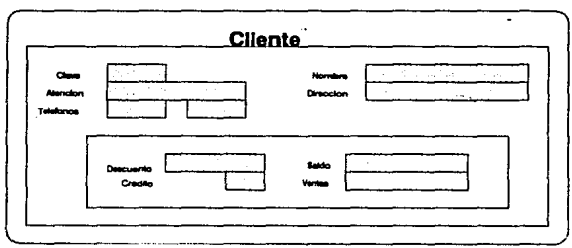


Figura 16. Formato de captura para el Catálogo de Clientes

Para mostrar la capacidad que tiene el **EDP** para generar diferentes tipos de código, se usará el mismo formato como entrada para cada módulo que se encuentra integrado al *Generador de Código*. Para generar el programa, se seleccionaron del menú de la figura 15 las opciones siguientes:

CAMPOS

- Generar archivo con la *declaracion*.

FORMATOS

- captura
- modificación
- consulta

CODIGO

- comandos para **dBASE III**
- *lenguaje C* para **Vitamin C**

En el *Apéndice A* se incluyen los archivos del código generado por el módulo para obtener programas en *lenguaje C* para ser aplicados en la librería de funciones **Vitamin C.**

En el *Apéndice B* se incluyen los archivos del código generado por el módulo para obtener programas en lenguaje de comandos **dBASE III.**

Conclusiones

El número de programadores que vuelven su vista hacia los manejadores de Base de Datos para desarrollar sistemas, se incrementa constantemente. Por esta razón, es importante crear herramientas de programación que ayuden a reducir el tiempo y las líneas de código necesarias para implantar aplicaciones relacionadas con el desarrollo de sistemas.

La herramienta que se proporciona como la parte materializada de este trabajo de tesis, puede contribuir en parte a conseguir ese objetivo.

Para apoyar lo anterior, se puede mencionar que al momento de concluir la redacción de este documento, el programa ya se esta usando en el desarrollo de aplicaciones reales obteniendo satisfactoriamente los resultados que se esperaban.

En este trabajo quedan plasmados una buena parte de los conocimientos adquiridos durante el desarrollo e implantación del programa.

En lo que a mi respecta, puedo mostrarme satisfecho de que el la elaboración de esta tesis haya enriquecido en gran medida mi experiencia en el campo profesional.

Mantengo la esperanza de que el contenido de este trabajo sirva como una propuesta para motivar el desarrollo de programas que estén enfocados hacia las necesidades propias de nuestro país.

BIBLIOGRAFIA

Ashton-Tate. *dBASE III Reference Manual*, 1985

Creative Programming Consultants, Inc. *Vitamin C User Guide*, 1987.

Kernighan B., Ritchie D. *The C Programming Language*, Prentice-Hall, 1978.

Rice E.D., *An Introduction to Information Structures and Paging Considerations for On-Line Text Editing Systems*, Advance in Informations Systems Science, Vol 4, Plenum Press, 1972

The Software Bottling Company. *Flash Code Manual*, Agosto 1985.

Tubilla Alberto, Sigüenza Rodrigo. *Diseño e Implantación de un Editor de Pantalla Tipo EMACS*, Comunicaciones Tēcnicas IIMAS-UNAM, febrero 1986.

Unify Corporation. *Developer's Reference Manual*, 1987.

Unisoft Corporation. *UNIX the Users's Manual*, 1982.

Apéndice A

Listado del programa en lenguaje C para la librería Vitamin C

```
/* catalogo.c */

#include <stdio.h>
#include <vcstdio.h>
#include "catalogo.h"

/* formato para capturar datos */

Acatalogo ()
{

        char    tmp66[16], tmp67[7], tmp68[16], tmp69[16];
        int     key;
        int     edpadre;

        edpadre= wxopen (5,1, 14,79, " Cliente ", CURSOR+ACTIVE+BORDER+802+STANDARD, 0,0);

        if (edpadre < 0) return edpadre;

        box (4,7, 7,67, COMBO, vc.dflt);

        atsay (1,6, "Clave: ");
        atsay (1,36, "Nombre: ");
        atsay (2,3, "Atencion: ");
        atsay (2,33, "Direccion: ");
        atsay (3,2, "Telefonos: ");
        atsay (5,14, "Credito: ");
        atsay (5,42, "Saldo: ");
        atsay (6,12, "Descuento: ");
        atsay (6,41, "Ventas: ");

        empty (clave, 10);
        empty (nombre, 30);
        empty (atencion, 20);
        empty (direccion, 30);
        empty (tel1, 9);
        empty (tel2, 9);
        empty (tmp66, 15);
        empty (tmp67, 6);
        empty (tmp68, 15);
        empty (tmp69, 15);

        atget (1,13, clave, "XXXXXXXXXX");
        atget (1,44, nombre, "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX");
        atget (2,13, atencion, "XXXXXXXXXXXXXXXXXX");
```

```
        atget (2,44, direccion, "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx");
        atget (3,13, tel1, "999-9999");
        atget (3,24, tel2, "999-9999");
        atget (5,23, tmp66, "999,999,999.99");
        atget (6,32, tmp67, "99.99");
        atget (5,49, tmp68, "999,999,999.99");
        atget (6,49, tmp69, "999,999,999.99");

        key= readgets ();
        wclose (edpwdo);

        sscanf (tmp66, "%lf", &credito);
        sscanf (tmp67, "%lf", &descto);
        sscanf (tmp68, "%lf", &saldo);
        sscanf (tmp69, "%lf", &ventas);

        return key;

}

/* formato para capturar datos */

Mcatalogo ()
{
        char    tmp64[16], tmp67[7], tmp68[16], tmp69[16];
        int     key;
        int     edpwdo;

        edpwdo= wopen (5,1, 14,79, " Cliente ", CURSOR+ACTIVE+BORDER+BO2+STANDARD, 0,0);

        if (edpwdo < 0) return edpwdo;

        box (4,7, 7,67, COMBO, vc.dflt);

        atsay (1,6, "Clave: ");
        atsay (1,36, "Nombre: ");
        atsay (2,3, "Atencion: ");
        atsay (2,33, "Direccion: ");
        atsay (3,2, "Telefonos: ");
        atsay (5,14, "Credito: ");
        atsay (5,42, "Saldo: ");
        atsay (6,12, "Descuento: ");
        atsay (6,41, "Ventas: ");

        sprintf (tmp66, "%lf", credito);
        sprintf (tmp67, "%lf", descto);
        sprintf (tmp68, "%lf", saldo);
        sprintf (tmp69, "%lf", ventas);
```

```
        atget (1,13, clave, "XXXXXXXXX");
        atget (1,44, nombre, "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX");
        atget (2,13, atencion, "XXXXXXXXXXXXXXXXXX");
        atget (2,44, direccion, "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX");
        atget (3,13, tel1, "999-9999");
        atget (3,24, tel2, "999-9999");
        atget (5,23, tmp86, "999,999,999.99");
        atget (6,32, tmp87, "99.99");
        atget (5,49, tmp88, "999,999,999.99");
        atget (6,49, tmp89, "999,999,999.99");

        key= readgets ();
        uclose (edpaubo);

        sscanf (tmp86, "%lf", &credito);
        sscanf (tmp87, "%lf", &descto);
        sscanf (tmp88, "%lf", &saldo);
        sscanf (tmp89, "%lf", &ventas);

        return key;
}

/* Formato para consultar datos */

Ccatalogo ()
{
        int     edpaubo;

        edpaubo= wopen (5,1, 14,79, " Cliente ", CURSOR+ACTIVE+BORDER+BD2+STANDARD, 0,0);

        if (edpaubo < 0) return edpaubo;

        box (4,7, 7,47, COMBO, vc.dflt);

        atsay (1,6, "Clave: ");
        atsay (1,36, "Nombre: ");
        atsay (2,3, "Atencion: ");
        atsay (2,33, "Direccion: ");
        atsay (3,2, "Telefonos: ");
        atsay (5,14, "Credito: ");
        atsay (5,42, "Saldo: ");
        atsay (6,12, "Descuento: ");
        atsay (6,41, "Ventas: ");
        atsay (1,13, clave);
        atsay (1,44, nombre);
        atsay (2,13, atencion);
        atsay (2,44, direccion);
        atsay (3,13, tel1);
        atsay (3,24, tel2);
```

```
        atssynum (5,23, &credito,"999,999,999.99",REAL);
        atssynum (6,32, &descto,"99.99",REAL);
        atssynum (5,49, &saldo,"999,999,999.99",REAL);
        atssynum (6,49, &ventas,"999,999,999.99",REAL);

        getone ();      /* pause */
        wclose (edpudo);

        return 0;
}

/* fin de programa */
```

Listado del archivo que contiene la declaración de campos en lenguaje C

```
/* catalogo.h */

extern double   credito, descto, saldo, ventas;
extern char     clave[10], nombre[30], atencion[20], direccion[30], tel1[9], tel2[9];
```

Apéndice B

Listado del programa en lenguaje de comandos dBASE III

```
* catalogo.prg

* Formato para capturar datos

PROCEDURE Acatalogo

CLEAR

        @ 5,1 SAY '┌──────────────────────────────────────────────────────────────────────┐'
        @ 6,1 SAY '│                                                                      │'
        @ 7,1 SAY '│                                                                      │'
        @ 8,1 SAY '│                                                                      │'
        @ 9,1 SAY '│                                                                      │'
        @ 10,1 SAY '│                                                                     │'
        @ 11,1 SAY '│                                                                     │'
        @ 12,1 SAY '│                                                                     │'
        @ 13,1 SAY '│                                                                     │'
        @ 14,1 SAY '└─────────────────────────────────────────────────────────────────────┘'

        @ 5,35 SAY ' Cliente '


        @ 10,9 SAY '┌──────────────────────────────────────────┐'
        @ 11,9 SAY '│                                          │'
        @ 12,9 SAY '│                                          │'
        @ 13,9 SAY '└──────────────────────────────────────────┘'

        @ 7,8 SAY 'Clave: '
        @ 7,38 SAY 'Nombre: '
        @ 8,5 SAY 'Atencion: '
        @ 8,35 SAY 'Direccion: '
        @ 9,4 SAY 'Telefonos: '
        @ 11,16 SAY 'Credito: '
        @ 11,44 SAY 'Saldo: '
        @ 12,14 SAY 'Descuento: '
        @ 12,43 SAY 'Ventas: '

        APPEND BLANK

        @ 7,15 GET clave PICTURE '!!!!!!!!!!'
        @ 7,46 GET nombre PICTURE '!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!'
        @ 8,15 GET atencion PICTURE '!!!!!!!!!!!!!!!!!!!!'
        @ 8,46 GET direccion PICTURE '!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!'
        @ 9,15 GET tel1 PICTURE '999-9999'
        @ 9,26 GET tel2 PICTURE '999-9999'
```

```
        @ 11,25 GET credito PICTURE '999,999,999.99'
        @ 12,34 GET descto PICTURE '99.99'
        @ 11,51 GET saldo PICTURE '999,999,999.99'
        @ 12,51 GET ventas PICTURE '999,999,999.99'

        READ

RETURN

* Formato para modificar datos

PROCEDURE Mclie

CLEAR

        @ 5,1 SAY '┌──────────────────────────────────────────────────────────────────────┐'
        @ 6,1 SAY '│                                                                      │'
        @ 7,1 SAY '│                                                                      │'
        @ 8,1 SAY '│                                                                      │'
        @ 9,1 SAY '│                                                                      │'
        @ 10,1 SAY '│                                                                      │'
        @ 11,1 SAY '│                                                                      │'
        @ 12,1 SAY '│                                                                      │'
        @ 13,1 SAY '│                                                                      │'
        @ 14,1 SAY '└──────────────────────────────────────────────────────────────────────┘'

        @ 5,35 SAY ' Cliente '


        @ 10,9 SAY '┌──────────────────────────────────────────────┐'
        @ 11,9 SAY '│                                              │'
        @ 12,9 SAY '│                                              │'
        @ 13,9 SAY '└──────────────────────────────────────────────┘'

        @ 7,8 SAY 'Clave: '
        @ 7,38 SAY 'Nombre: '
        @ 8,5 SAY 'Atencion: '
        @ 8,35 SAY 'Direccion: '
        @ 9,4 SAY 'Telefonos: '
        @ 11,16 SAY 'Credito: '
        @ 11,44 SAY 'Saldo: '
        @ 12,14 SAY 'Descuento: '
        @ 12,43 SAY 'Ventas: '

        @ 7,15 GET clave PICTURE '!!!!!!!!!'
        @ 7,46 GET nombre PICTURE '!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!'
        @ 8,15 GET atencion PICTURE '!!!!!!!!!!!!!!!!!!'
        @ 8,46 GET direccion PICTURE '!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!'
        @ 9,15 GET tel1 PICTURE '999-9999'
```

```
        @ 9,26 GET tel2 PICTURE '999-9999'
        @ 11,25 GET credito PICTURE '999,999,999.99'
        @ 12,34 GET descto PICTURE '99.99'
        @ 11,51 GET saldo PICTURE '999,999,999.99'
        @ 12,51 GET ventas PICTURE '999,999,999.99'

        READ

RETURN

* formato para consultar datos

PROCEDURE Cclie

CLEAR

        @ 5,1 SAY '┌──────────────────────────────────────────────────────────────┐'
        @ 6,1 SAY '│                                                              │'
        @ 7,1 SAY '│                                                              │'
        @ 8,1 SAY '│                                                              │'
        @ 9,1 SAY '│                                                              │'
        @ 10,1 SAY '│                                                              │'
        @ 11,1 SAY '│                                                              │'
        @ 12,1 SAY '│                                                              │'
        @ 13,1 SAY '│                                                              │'
        @ 14,1 SAY '└──────────────────────────────────────────────────────────────┘'

        @ 5,35 SAY ' Cliente '


        @ 10,9 SAY '┌──────────────────────────────────────────┐'
        @ 11,9 SAY '│                                          │'
        @ 12,9 SAY '│                                          │'
        @ 13,9 SAY '└──────────────────────────────────────────┘'

        @ 7,8 SAY 'Clave: '
        @ 7,38 SAY 'Nombre: '
        @ 8,5 SAY 'Atencion: '
        @ 8,35 SAY 'Direccion: '
        @ 9,4 SAY 'Telefonos: '
        @ 11,16 SAY 'Credito: '
        @ 11,44 SAY 'Saldo: '
        @ 12,14 SAY 'Descuento: '
        @ 12,43 SAY 'Ventas: '

        @ 7,15 SAY clave PICTURE '!!!!!!!!!'
        @ 7,46 SAY nombre PICTURE '!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!'
        @ 8,15 SAY atencion PICTURE '!!!!!!!!!!!!!!!!!!!'
        @ 8,46 SAY direccion PICTURE '!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!'
```

```
        @ 9,15 SAY tel1 PICTURE '999-9999'
        @ 9,26 SAY tel2 PICTURE '999-9999'
        @ 11,25 SAY credito PICTURE '999,999,999.99'
        @ 12,34 SAY descto PICTURE '99.99'
        @ 11,51 SAY saldo PICTURE '999,999,999.99'
        @ 12,51 SAY ventas PICTURE '999,999,999.99'

        WAIT

RETURN

* fin de programa
```

Listado del archivo que contiene la declaración de campos en dBASE III

```
* catalogo.mem

clave      pub C **
nombre     pub C **
atencion   pub C **
direccion  pub C **
tel1       pub C **
tel2       pub C  **
credito    pub N  0.0
descto     pub N  0.0
saldo      pub N  0.0
ventas     pub N  0.0
```