75
2ej

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

Diseño e Implementación de un sistema generador de reportes para la elaboración de cuadros estadísticos y consultas de series de tiempo. (Sistema de finanzas públicas ejecutivo)

T E S I S

QUE PARA OBTENER EL TITULO DE:

INGENIERO EN COMPUTACION

P R E S E N T A :

MARIA DOLORES VARGAS VITE

DIRECTOR DE TESIS:
ING. RAUL MARTINEZ MERCADO

MEXICO, D. F. **SEPTIEMBRE 1993**

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

ÍNDICE

INTRODUCCIÓN..... I-1

1.TEORÍA DE REPORTES..... 1-1
1.1 Agrupación de datos en informes..... 1-2
1.2 Tipos de reporte..... 1-2
1.2.1 Reporte tabular..... 1-2
1.2.2 Reporte Matricial..... 1-3
1.2.3 Reporte de personalización..... 1-3
1.3 Estructura de un reporte..... 1-3
1.4 Formatos de impresión..... 1-5
1.4.1 Tipos de papel..... 1-5
1.4.2 Tintas..... 1-5
1.4.3 Fonts..... 1-5
1.4.4 Trazo..... 1-6
1.5 Tipos de impresoras..... 1-7
1.5.1 Impresoras de matriz de puntos..... 1-8
1.5.2 Impresoras de margarita..... 1-9
1.5.3 Impresoras de cinta..... 1-9
1.5.4 Impresoras de tambor..... 1-9
1.5.5 Impresora térmica..... 1-10
1.5.6 Impresora de chorro de tinta..... 1-10
1.5.7 Impresora láser..... 1-11

2.TEORÍA DE ARCHIVOS Y HOJAS ELECTRÓNICAS..... 2-1
2.1 Conceptos generales..... 2-2
2.1.1 Dato..... 2-2

2.1.2 Registro..... 2-2
2.1.3 Llave de registro..... 2-3
2.1.4 Archivo..... 2-3
2.2 Tipos de archivos..... 2-3
2.2.1 Archivo maestro..... 2-3
2.2.2 Archivo de transacciones..... 2-4
2.2.3 Archivo de tablas..... 2-4
2.2.4 Archivo de informes..... 2-5
2.2.5 Otros archivos..... 2-5
2.3 Métodos de organización de archivos..... 2-5
2.3.1 Organización secuencial..... 2-6
2.3.1.1 Lectura de archivos secuenciales..... 2-6
2.3.2 Organización de acceso directo..... 2-6
2.3.2.1 Direccionamiento directo..... 2-7
2.3.2.2 Requerimientos para el direccionamiento directo..... 2-7
2.3.3 Organización indexada..... 2-8
2.3.3.1 Características de un índice..... 2-8
2.4 Manipulación de archivos empleando lenguaje C.... 2-9
2.4.1 Abriendo un archivo..... 2-10
2.4.2 Cerrando un archivo..... 2-13
2.4.3 Leyendo y escribiendo datos..... 2-13
2.4.4 Detección de errores..... 2-18
2.4.5 Acceso secuencial y acceso aleatorio..... 2-19
2.5 Hojas electrónicas 2-22
2.5.1 Celdas..... 2-22
2.5.2 Valores y rótulos..... 2-23
2.5.3 Rangos y nombres de rango..... 2-23
2.5.4 Fórmulas y funciones..... 2-24
2.5.5 Precedencia en los cálculos..... 2-25
2.5.6 Funciones @..... 2-25
2.5.7 Gráficos y diagramas..... 2-26
2.5.8 Formato de una hoja electrónica..... 2-26

3.Análisis y diseño del Sistema..... 3-1
3.1 Determinación de requerimientos..... 3-2
3.2 Diseño del Sistema..... 3-3
3.2.1 Series de Tiempo..... 3-3
3.2.1.1 Clasificación de las Series de Tiempo.... 3-4
3.2.2 Cuadros Estadísticos..... 3-7
3.2.2.1 Clasificación de Cuadros Estadísticos.... 3-8
3.2.3 Diseño de Salidas..... 3-8
3.2.3.1 Diseño de Cuadros Estadísticos..... 3-8
3.2.3.2 Elaboración de un Cuadro Estadístico..... 3-12
3.2.4 Diseño y procesamiento de archivos..... 3-17
3.2.4.1 Diseño de la Base de Datos..... 3-17
3.3 Relación de archivos..... 3-20
3.3.1 Archivos de Tablas..... 3-20
3.3.2 Archivos de Valores..... 3-21
3.3.3 Formato de registros de archivos..... 3-21
3.4 Flujo de información..... 3-23
3.5 Módulos que componen el Sistema..... 3-25
3.6 Desarrollo de Software..... 3-35
3.6.1 Lenguaje de Programación..... 3-35
3.6.2 Implementación..... 3-36

Conclusiones..... C-1

Bibliografía..... B-1

Anexos..... A-1

INTRODUCCIÓN

INTRODUCCIÓN

La Dirección General de Planeación Hacendaria, de la Secretaría de Hacienda y Crédito Público (a la cual pertenezco), tenía la necesidad de contar con un sistema computacional, capaz de almacenar grandes volúmenes de información, permitir el acceso a dicha información de manera oportuna y confiable, elaborar complejos reportes y salvaguardar los datos confidenciales, de manera tal que fueran accesibles solamente a aquellas personas autorizadas.

Dicha necesidad, me dio la pauta para desarrollar el Sistema de Finanzas Públicas Ejecutivo, el cual, pone a disposición de los usuarios un volumen considerable de información, que integra a los diversos sectores de la administración pública federal y pretende hacer de este una herramienta valiosa que apoye a los usuarios en el desempeño de sus labores.

Fue creado para usuarios ejecutivos, con la finalidad de brindarles una interfase más rápida y accesible a la información.

Está desarrollado en lenguaje C con librerías de Code Base, para operar en una red Ethernet Novell, permitiendo la exportación de la información a herramientas útiles como Lotus 1-2-3, facilitando así, impresiones de mayor calidad.

Para el almacenamiento de la información se emplea una base de datos compuesta por archivos de tipo tabla (formato dBase) y archivos random con un tipo de organización de acceso directo, lo que permite optimizar el tiempo de respuesta en la consulta y recuperación de datos.

Para la exportación de la información a Lotus, fue necesario implementar rutinas que pudieran crear el formato de una hoja de cálculo, tal como lo hace Lotus 1-2-3.

Este sistema orientado a red, permite al usuario un más fácil y rápido acceso a la información, siendo de esta

forma una herramienta auxiliar para la consulta de variables económicas (series de tiempo) así como para la consulta y diseño de cuadros estadísticos (reportes matriciales).

Es importante mencionar que tan solo es un sistema de consulta y su fuente de alimentación es el Sistema de Finanzas Públicas que trabaja en una minicomputadora VAX3400 y es en ese sistema en donde se realizan las actualizaciones a la información.

El Sistema, integra en una Base de Datos común, las principales variables financieras del sector público.

Con el Sistema de Finanzas Públicas Ejecutivo los usuarios cuentan con una herramienta de trabajo que les permite:

- Clasificar las variables que integran el sistemas de acuerdo con su naturaleza.
- Generar cuadros estadísticos a partir de la información contenida en el Sistema de Finanzas Públicas.
- Consultar la información que registra el sistema ya sea en forma de Series de Tiempo ó de Cuadros Estadísticos.
- Exportar la información hacia herramientas como Lotus 1-2-3, con el fin de realizar procesos con el Software local.

Una vez que se determinan las necesidades del usuario, la información requerida se ubica en el sistema para su consulta y manejo.

Las funciones básicas del Sistema de Finanzas Públicas Ejecutivo son:

- ORGANIZAR
- MANIPULAR
- RECUPERAR

Para su organización, las cifras relativas a cada variable se almacenan en forma de Series de Tiempo, lo que permite contar con la evolución de las variables a través del tiempo; definir diversas periodicidades (Anual, Semestral, Trimestral, Mensual, Quincenal, Semanal ó Diaria) sin afectar la operación del sistema y clasificar la información como Definitiva ó de flujo

nominal, de Presupuesto Original y de Presupuesto Modificado, según corresponda.

En lo que respecta a la manipulación posterior de las cifras ya organizadas, se cuenta con dispositivos para la consulta de series, así como el diseño y generación de cuadros estadísticos.

Para recuperar la información, el sistema distingue entre las cifras almacenadas en forma de series de tiempo y los cuadros estadísticos elaborados a partir de ellas. En ambos casos se cuenta con un índice temático, que permite al usuario localizar fácilmente información en el rango, tipo y periodicidad deseados.

CAPÍTULO 1

TEORÍA DE REPORTES

1.1 AGRUPACIÓN DE DATOS EN INFORMES

Con bastante frecuencia se requiere agrupar los registros en base a los contenidos de los campos de la base de datos o el valor de una expresión. Cuando se imprime un registro con apartado de grupo, se imprime el apartado de introducción de grupo antes del primer registro de detalle del grupo y, después de imprimir cada registro, se comprueba si se ha impreso el número de registros requerido o si el valor del campo o la expresión que define el grupo ha cambiado. Si es así, se imprime el apartado de resumen de grupo, seguido por el apartado de introducción del grupo para el siguiente grupo y así sucesivamente.

El apartado de resumen de grupo normalmente incluye subtotales u otros datos estadísticos de totales basados en los registros de un grupo.

1.2 TIPOS DE REPORTE

Podemos clasificar los reportes en tres categorías estándar

1.2.1 REPORTE TABULAR

En un reporte tabular, los campos de cada registro se alinean en columnas con las cabeceras al principio, igual que en la visualización de hojear (browse) o de la salida de una orden LIST en DBase. Las aplicaciones típicas para los informes por columna incluyen el registro de factura, listado de ingresos en efectivo, informe de

atrasos, lista de precios, e informes de inventario.

1.2.2 REPORTE MATRICIAL

En un reporte matricial, que a veces es referido como informe orientado a páginas o informe de página completa (en contraste con el reporte tabular orientado a líneas), los campos de un registro se disponen verticalmente o a lo largo de la página impresa. Como su nombre sugiere, la disposición de formulario se puede utilizar para "rellenar los huecos en blanco" de formularios preimpresos tales como los formularios de primas de seguro, historiales clínicos, altas medicas o de personal, o bien si la impresora soporta los tamaños y tipos de letra apropiados, para reproducir un formulario en su totalidad, incluyendo el texto preimpreso y los datos. En estos informes, un registro ocupa normalmente una o más páginas completas. Otros formatos de formulario, tales como la lista de referencia de clientes, puede requerir solamente unas cuantas líneas para cada registro. La característica a distinguir para el reporte matricial es que en la salida impresa no aparecen columnas distintas; los campos y el texto se pueden colocar en forma independiente en cualquier parte de la página.

1.2.3 REPORTE DE PERSONALIZACIÓN

En un reporte de personalización, tal como una carta personalizada, el cuerpo del informe consta de tres textos de formato libre , junto con los campos seleccionados de la base de datos activa, impresos de forma separada (igual que el nombre y la dirección en una carta), o mezclados en el texto.

1.3 ESTRUCTURA DE UN REPORTE

Todos los informes comprenden una serie de apartados o regiones, que se extienden horizontalmente a través de la página y abarcan una o más líneas impresas. Existen siete tipos de apartados, cada uno de los cuales se utiliza para la información impresa en una secuencia determinada en el informe del siguiente modo:

INTRODUCCIÓN DEL INFORME

Información impresa una vez al principio del informe, tal como un título adicional, una portada de página, o varias páginas de un texto introductorio.

CABECERA DE PAGINA

Información impresa al principio de cada página (excepto en el apartado de introducción del informe, donde la cabecera de página es opcional), tal como la fecha, número de página, título de página y encabezamiento de las columnas.

INTRODUCCIÓN DEL GRUPO

Información impresa al principio de cada grupo, que normalmente incluye el campo o la expresión que define al grupo.

CUERPO

Información que se imprime una vez para cada registro, incluyendo los campos de la base de datos, campos calculados y texto fijo.

RESUMEN DE GRUPO

Información impresa al final de cada grupo, tal como cuentas del grupo, subtotales y otros datos estadísticos, quizá con algún texto explicativo.

PIE DE PAGINA

Información impresa al final de cada página, tal como la fecha o número de página.

RESUMEN DEL INFORME

Información impresa al final del informe, tal como una conclusión o totales principales y otros datos estadísticos de resumen.

1.4 FORMATOS DE IMPRESIÓN

1.4.1 TIPOS DE PAPEL

El papel puede ser normal o especial. En el primer caso, se deposita una tinta sobre este, empleándose para ello varios procedimientos, entre los que cabe destacar el de impacto, mediante cinta entintada, el de chorro de tinta, el de plumilla o rotulador y el de depósito de toner. En el segundo caso, el papel es sensible a alguna variable física (luz para el papel fotosensible o temperatura para el papel térmico), de forma que, cuando se le somete a un cierto nivel de esa variable física, cambia de color de forma permanente. Por ejemplo, en el caso del papel térmico un pequeño foco calorífico hace de plumilla de escritura. Evidentemente, el papel especial es más caro que el normal, por lo que no se usa demasiado. Además, no suele ser muy estable, por lo que se degrada con el tiempo.

1.4.2 TINTAS

El trazo se puede hacer con una sola tinta o con varias tintas de distintos colores. A su vez, la intensidad de la tinta puede ser fija o variable, existiendo dispositivos que solo tienen la posibilidad de seleccionar entre dos intensidades y otros, más complejos, que permiten seleccionar entre una amplia gama de ellas. Lo más común, por su menor costo, es usar una sola tinta con una o dos intensidades , aunque se va extendiendo cada vez más el uso de varias tintas, como está pasando, por ejemplo, con la difusión de las impresoras de color.

1.4.3 FONTS

La presentación puede ser alfanumérica o gráfica. En el primer caso, se suele disponer exclusivamente de un juego de tipos, que permiten producir solamente textos. Por el contrario, en el segundo caso , se pueden generar dibujos.

En algunos dispositivos el juego de tipos es

intercambiable, como sucede en las impresoras de margarita, que permiten cambiar manualmente la margarita que lleva grabados los tipos. Otros dispositivos disponen simultáneamente de varios juegos de tipos , lo que permite una mayor flexibilidad de impresión.

Por su parte, los periféricos gráficos suelen tener la posibilidad de generar tipos de menor o mayor calidad. Como ejemplo de éste último caso, puede citarse el de las impresoras láser, que además de tener varios juegos de tipos de alta calidad, permiten reproducciones gráficas.

1.4.4 TRAZO

El trazo puede ser continuo o por puntos. En el primer caso, los caracteres o los dibujos se realizan con un trazo continuo, mientras que en el segundo se realizan por medio de una serie de puntos, más o menos próximos, según sea la calidad del dispositivo.

En general, el trazo por puntos es más económico, pero tiene una menor calidad que el continuo, puesto que los puntos empleados no suelen tener una gran resolución.

La representación alfanumérica por trazo de puntos se realiza asignando un rectángulo a cada caracter. Este rectángulo se retícula mediante una serie de filas y columnas, pudiéndose escribir un punto en cada uno de los cuadritos de esta retícula. Los tipos alfanuméricos se representan en base a una serie de puntos.

Por su parte las representaciones graficas se hacen dividiendo toda la hoja en una gran reticula y aproximando los dibujos por puntos de esta reticula. Es claro que, cuanto más fina sea la retícula , mejor será la calidad del dibujo.

El trazo continuo se puede obtener por impacto, mediante un tipo que tenga el caracter que se desea imprimir, o por una punta de impresión (plumilla, rotulador, punzón, etc.). En este ultimo caso, el movimiento relativo de la punta de impresión sobre el soporte produce el trazo, que por ello es continuo.

Algunos dispositivos de trazo por puntos tienen dos formas de trabajo: normal y de alta calidad. Esta mayor calidad se obtiene haciendo , por ejemplo, dos pasadas de trazado desplazadas el espesor de medio punto y

avanzando el cabezal de escritura de medio en medio punto, de forma que la separación horizontal y vertical entre puntos se reduzca, en éste caso, a la mitad, superponiéndose éstos. Con ello, se incrementa la calidad, pero se reduce la velocidad de impresión a la cuarta parte.

1.5 TIPOS DE IMPRESORAS

Existe una gran variedad de impresoras, con velocidades que varían desde 10 c.p.s. hasta 18,000 líneas por minuto o cuatro páginas por segundo. El ancho de las líneas es ampliamente variable, aunque los tamaños típicos son los de 80 y 132 caracteres. Según sus características, las impresoras se pueden clasificar, como se indica a continuación:

A. Por la forma de imprimir:

Impresoras de caracteres.
Impresoras de línea.
Impresoras de página.

B. Por el mecanismo de impresión:

Impresoras de impacto.
- Matriz de puntos.
- Margarita, bola, cilindro, etc.
- Cinta.
- Tambor.

Impresora térmica.
Impresora laser.
Impresora de chorro de tinta.

Las impresoras de caracteres escriben caracter a caracter, por lo que constituyen su nivel más lento. Las impresoras de línea imprimen simultáneamente toda una línea, por lo que permiten alcanzar altas velocidades de impresión. Finalmente las impresoras de página imprimen toda una página de golpe, aunque construyen internamente esta línea a línea. El ejemplo más representativo de impresoras de página lo forman las impresoras de láser, que constituyen, con las de cinta, el espectro de impresoras más rápido.

En general, las impresoras de impacto son más ruidosas que los otros tipos y suelen ser mecánicamente más complicadas, por lo que son más delicadas desde el

punto de vista del mantenimiento y de las condiciones ambientales que exigen para su funcionamiento.

1.5.1 IMPRESORAS DE MATRIZ DE PUNTOS

Este tipo de impresoras de impacto es de trazo por puntos y se puede clasificar como impresora de caracteres, puesto que va formando éstos, uno a uno a base de puntos.

La figura 1.1 muestra el mecanismo de impresión. En esencia, consta de tantos electroimanes como filas de puntos tienen los caracteres (generalmente 8). Estos electroimanes activan unos punzones redondos que, mediante una cinta entintada, imprimen puntos.

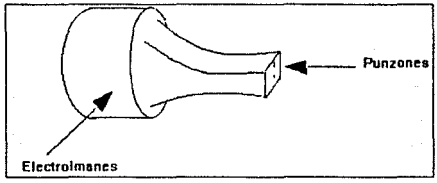


Figura 1.1. Mecanismo de Impresión por Matriz de Puntos.

Existe una gran variedad de impresoras de puntos, con velocidades que oscilan entre 30 y 300 c.p.s. Estas impresoras pueden contar con gran cantidad de características especiales , entre las que cabe destacar las siguientes:

Posibilidad de varios tipos de letra.
Impresión de alta calidad.
Impresión de alta densidad (132 c.p.l. en las de 80).
Impresión en avance y en retroceso. De esta forma se aprovecha el movimiento de retroceso de la cabeza, aumentando la velocidad efectiva de impresión.
Posibilidades gráficas, que permiten formar dibujos, hechos mediante puntos, con la resolución básica de la impresora o con la resolución de alta calidad.

1.5.2 IMPRESORAS DE MARGARITA

Existe un conjunto de impresoras cuyo mecanismo de impresión es similar al empleado por las maquinas de escribir, siendo su representante más significativo la impresora de margarita.

Estos periféricos disponen de un elemento en forma de margarita, de tambor o de bola, que lleva grabado en relieve los tipos. La impresión que es evidentemente de impacto, se produce posicionando el caracter deseado de la margarita o del tambor delante del martillo y activando un electroimán que le hace golpear. De esta forma, el caracter, al golpear el papel a través de una cinta entintada queda impreso.

Estas impresiones suelen llamarse de calidad de carta, puesto que imprimen como una máquina de escribir. Su velocidad de impresión suele ser del orden de 45 c.p.s. En la actualidad han sido desplazadas del mercado por las impresoras láser de bajo costo.

1.5.3 IMPRESORAS DE CINTA

Este tipo de impresora de impacto emplea una banda o cadena metálica circular, que tiene grabados en relieve los caracteres. Además, la impresora tiene tantos martillos como letras caben en una línea (generalmente 132).

La cinta o cadena gira, desplazándose dentro de los martillos, que, de forma perfectamente síncrona, golpean el caracter deseado, cuando pasa por delante suyo. El caracter es golpeado a través del papel y de una cinta entintada, por lo que queda impreso en el.

El tiempo máximo necesario para imprimir una línea es el tiempo que tarda la cinta o cadena en rotar delante de los martillos, por lo que se alcanzan velocidades que oscilan entre varios cientos de líneas por minuto a varios miles.

1.5.4 IMPRESORAS DE TAMBOR

El principio de estas impresoras es similar al de las de cinta, puesto que disponen también de tantos martillos como letras tengan las líneas. La diferencia estriba, en que, en este caso, el soporte de los caracteres no es una

cinta sino un tambor del tamaño de una línea (generalmente con 132 caracteres de longitud). El tambor tiene grabados en relieve en su perímetro todos los caracteres por cada una de las 132 posiciones de una línea.

La selección del caracter se hace eligiendo el caracter del tambor, puesto que éste está continuamente girando.

Dado que el tambor es más caro que la cinta o cadena, pues tiene grabados muchos más caracteres, éste tipo de impresora está en desuso.

1.5.5 IMPRESORA TÉRMICA

La impresora térmica se basa en el uso de papel termosensible, esto es, que cambia de color por efecto del calor. El cabezal de escritura está formado por un elemento con puntos caloríficos, por lo que la impresión tiene trazos de puntos de forma similar al de las impresoras de matriz de puntos.

Se emplean poco, puesto que, aunque son muy silenciosas y mecánicamente muy sencillas, el papel térmico es caro y poco estable (con el tiempo se deteriora).

En la actualidad se están popularizando las impresoras térmicas que emplean cinta térmica en vez de papel térmico. El cabezal calienta la cinta, que deposita su tinta en papel normal, por lo que se elimina uno de los mayores inconvenientes que tenían estos dispositivos.

1.5.6 IMPRESORA DE CHORRO DE TINTA

En éste tipo de impresora se genera un fino chorro de tinta pulverizada. Las gotitas de éste chorro quedan cargadas de electricidad estática, por lo que puede ser gobernado en dirección, mediante dos campos perpendiculares, de forma similar a como se gobierna el haz de electrones de un CRT, produciendo de ésta manera los caracteres deseados.

No ha tenido una gran difusión, puesto que otras tecnologías, como el láser le aventajan. Sin embargo, permite trabajar con varias tintas simultáneamente,

mezclándolas incluso entre sí, lo que hace que se aplique en la impresión en color.

1.5.7 IMPRESORA LÁSER

Las impresoras láser trabajan con un principio parecido al de las fotocopiadoras, por lo que imprimen páginas enteras de una vez, siendo impresoras de página. Cada día están tomando más auge, por su calidad, velocidad y bajo nivel de ruido, existiendo una amplísima gama de modelos, que cubren desde las necesidades de un pequeño computador personal, hasta las grandes instalaciones.

Como indica la figura 1.2, constan de un tambor cuya superficie admite ser cargada con electricidad estática, y de un láser, que puede barrer toda su superficie. Primeramente se carga toda la superficie del tambor y, seguidamente, con la luz del láser se elimina ésta carga estática en las zonas que no se desee entintar. A continuación, el tambor se expone a un polvo toner, que se pega en las zonas del tambor que están cargadas estáticamente. Este toner pasa al papel al precionarlo sobre el tambor y, finalmente, es fijado sobre el papel mediante calor.

En una fotocopiadora, la imagen se graba en el tambor ópticamente desde un original, mientras que en las impresoras se graba mediante un barrido del láser por su superficie siendo esta su mayor diferencia. Este barrido se realiza por microlíneas, mediante un espejo giratorio que refleja el rayo láser con un ángulo variable, que hace que éste recorra una generatriz. Además, girando adecuadamente el tambor se van produciendo las sucesivas generatrices que forman la página, de forma similar a como el haz de electrones del CRT barre la pantalla, creando su imagen. Para generar la señal que modula la intensidad del láser, hay que realizar previamente la operación de formato de imagen, que consiste en traducir el texto alfanumérico y las ordenes gráficas, enviadas por la computadora, en una matriz de bits, que determina la intensidad del láser en cada punto de la página.

Evidentemente, éste tipo de impresora es de trazo por puntos. Sin embargo, dado que los puntos se hacen de un tamaño muy reducido, la calidad alcanzada es muy buena. La calidad normal es de 300 puntos por pulgadas, pero hay impresoras de 1270 y 2540 puntos por pulgada.

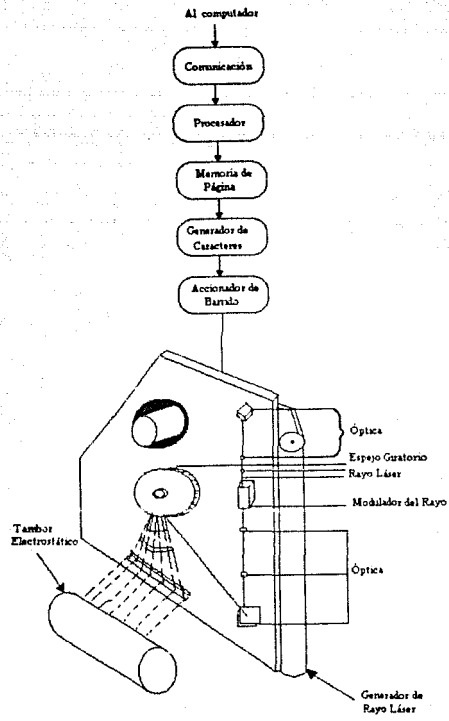


Fig. 1.2 Estructura de una impresora láser.

Otra gran ventaja de éste tipo de impresoras es que no están limitadas en cuanto a los tipos de letra, pues disponen de varios modelos, que se pueden mezclar en un mismo texto. Además, producen gráficos de muy buena calidad que se pueden mezclar con el texto.

Finalmente hay que destacar su alta velocidad, que alcanza hasta cuatro páginas por segundo, convirtiéndolas en las impresoras más rápidas.

CAPÍTULO 2

TEORÍA DE ARCHIVOS Y HOJAS ELECTRÓNICAS

2.1 CONCEPTOS GENERALES

2.1.1 DATO

Los elementos individuales de contenido se conocen como elementos dato. Por ejemplo, un cheque de banco tiene los siguientes elementos dato: quién origina el cheque, número de cheque, fecha, a quien se le va a pagar, cantidad numérica, cantidad con letra, notas, identificación del banco, número de cuenta y firma.

Cada elemento dato tiene un valor específico asociado con él. Por ejemplo, el número 1082 es el valor del dato asignado al elemento dato llamado número de cheque.

Cada elemento dato se identifica por un nombre y se le asigna un valor. La asociación de un valor con un campo crea una instancia del dato. Un nombre alterno para instancia es ocurrencia. Algunos sinónimos para "elemento dato" son "campo", o simplemente "elemento".

2.1.2 REGISTRO

El conjunto de datos que pertenecen a una entrada, como cheque de banco, es un registro. El cheque de banco tratado como unidad es, por lo tanto, un registro que consiste de siete campos por separado. Nótese que todos los datos están relacionados con la transacción de pago. Dado que ésta es una característica de los registros, una definición más completa del término es un conjunto de elementos datos relacionados entre sí y que pertenecen a una entidad de interés.

2.1.3 LLAVE DE REGISTRO

A menudo es necesario distinguir un registro específico de otro. Se selecciona un elemento dato en el registro, que es posible que sea único (que nunca se repite un valor) en todos los registros de un archivo y de esta forma se utiliza para propósitos de identificación. Este dato se conoce como llave del registro. Es importante notar que la llave del registro, o simplemente llave, es un dato que ya forma parte del registro, más que ser una pieza adicional de dato añadida al mismo solamente para propósitos de identificación.

2.1.4 ARCHIVO

Un archivo es una colección de registros que almacenamos en un medio magnético, generalmente un disco, para poder manipularlos en cualquier momento.

Cada registro se incluye dentro de un archivo porque pertenece a la misma entidad. Por ejemplo, un archivo de cheques consta solamente de cheques. Los registros de inventario y las facturas no pertenecen a un archivo de cheques, ya que pertenecen a entidades diferentes.

El número de registros en un archivo determina el tamaño del mismo.

2.2 TIPOS DE ARCHIVOS

Los registros se recaban y mantienen como archivos. Los cuatro tipos principales de archivos son el maestro, de transacciones, de tabla y de informes.

2.2.1 ARCHIVO MAESTRO

Los sistemas de información siempre están en movimiento, siempre existen y se utilizan mientras sean significativos para la empresa. Por lo tanto, los archivos en los cuales se guarda la información necesaria de las actividades del negocio continúan en existencia. Un archivo maestro es una colección de registros sobre un aspecto importante de las actividades de la compañía. Puede contener datos que describen el estado actual de acontecimientos específicos o de indicadores del negocio.

Por ejemplo, el archivo maestro de un sistema de cuentas por pagar muestra el saldo que se le debe a cada vendedor o proveedor a quienes la empresa compra suministros o servicios. El saldo debido a cada proveedor, refleja el estado actual de todas las cuentas, es decir, el resultado de todas las compras, pagos y créditos realizados entre la empresa y el proveedor.

Los archivos maestros son útiles solamente mientras sean exactos y estén actualizados; en otras palabras, se les debe dar mantenimiento para reflejar incluso los hechos más recientes que cambian el archivo antes de que éste pueda utilizarse. Los archivos maestros se mantienen al día a través del uso de archivos de transacciones.

2.2.2 ARCHIVO DE TRANSACCIONES

Un archivo de transacciones es un archivo temporal que tiene dos objetivos: acumular datos sobre los acontecimientos, conforme ocurren, y actualizar los archivos maestros para reflejar los resultados de las transacciones actuales. El término transacción se refiere a cualquier suceso del negocio que afecte a la empresa y sobre el cual se captan los datos. en las empresas los ejemplos de transacciones comunes son realizar compras, pagarlas, contratar personal, pagar a los empleados y registrar las ventas. Los datos importantes para la compañía se recaban de cada hecho y se guardan en un archivo: el archivo de transacciones.

Los archivos de transacciones se procesan contra los archivos maestros para actualizar éstos últimos.

2.2.3 ARCHIVO DE TABLAS

En muchos sistemas se incluye un tipo especial de archivo maestro para cubrir requerimientos especiales de procesamiento, donde los datos se deben consultar en forma repetitiva. Los archivos de tablas son archivos permanentes que contienen datos de referencia utilizados cuando las transacciones se procesan, se actualizan archivos maestros o se producen salidas. Como su nombre lo indica, estos archivos almacenan tablas de datos de referencia.

Con frecuencia se usan los archivos de tablas para almacenar los datos que de otra forma serían incluidos en archivos maestros o en programas de computadora. Los archivos de tablas conservan el espacio del almacenamiento y facilitan el mantenimiento del programa almacenando los

datos en un archivo que, de otra forma, se incluirían en programas o en registros del archivo maestro.

2.2.4 ARCHIVO DE INFORMES

Muy a menudo en los centros de sistemas de información, los operadores encuentran que la unidad central de proceso de la computadora produce mucho más datos de salida de lo que la impresora puede imprimir; por lo tanto, al seguir la secuencia normal de los hechos, la UCP se demoraría mientras los resultados se producen en la impresora.

Dado que la UCP es la parte más poderosa (y cara) de la mayor parte de los sistemas de cómputo, se desea obtener el máximo provecho de ella.

Los archivos de informes son el contenido recolectado de los informes individuales de salida o documentos producidos por el sistema, los crea el sistema en donde se producen muchos informes, pero no hay tiempo de impresión. El archivo de informe contiene los datos de salida no impresos.

2.2.5 OTROS ARCHIVOS

Otros archivos también participan en los sistemas de información. En la generalidad de los casos, son usos especiales de uno de los tipos de archivos analizados; por ejemplo, un archivo de respaldo es una copia de un archivo maestro, de uno de transacciones o de uno de tablas que se obtiene para asegurar que esté disponible una copia si algo le sucede al original.

2.3 MÉTODOS DE ORGANIZACIÓN DE ARCHIVOS

Los registros se almacenan en los archivos utilizando una organización, que determina cómo se emplea el almacenamiento y cómo se ubicarán y consultarán los registros. Existen tres formas comunes para almacenar y consultar los registros dentro de un archivo. Dos de estos métodos, el secuencial y el directo, están disponibles en todas las computadoras. El tercero, indexado, es posible solamente si se adquiere software especial.

2.3.1 ORGANIZACIÓN SECUENCIAL

Esta es la forma más sencilla para almacenar y consultar registros dentro de un archivo. En un archivo secuencial, los registros se almacenan uno después de otro, sin importar el valor real de los datos en los registros. El primer registro almacenado se coloca al principio del archivo; el segundo se almacena a continuación (no hay posición sin utilizarse), el tercero después del segundo y así sucesivamente. Este hecho nunca cambia en un archivo secuencial.

Es una característica de los archivos secuenciales que todos los registros se almacenan por su posición: un registro es el primero, el siguiente el segundo y así sucesivamente. No existen direcciones o asignaciones de ubicaciones en un archivo secuencial.

2.3.1.1 LECTURA DE ARCHIVOS SECUENCIALES

Para leer un archivo secuencial, el sistema comienza siempre al principio del archivo. Si el registro que se busca está en algún lugar dentro del archivo, el sistema lee todo el archivo hasta encontrarlo, de registro a registro. Por ejemplo, si sucede que un registro en particular está en décimo lugar dentro de un archivo, el sistema comienza en el primero y lee un registro después del otro hasta alcanzar el décimo. No puede ir directamente al décimo registro en un archivo secuencial sin empezar desde el principio. De hecho, no sabe que es el décimo. Dependiendo de la naturaleza del sistema que se ha diseñado, esta característica puede ser una ventaja o una desventaja.

2.3.2 ORGANIZACIÓN DE ACCESO DIRECTO

Cuando las características de los archivos secuenciales son una desventaja para un sistema propuesto, el diseño alternativo utiliza una organización de acceso directo. Este método requiere que el programa le diga al sistema dónde se almacena un registro, de manera que pueda buscarlo ahí. En contraste con la organización secuencial, el procesamiento de un archivo de acceso directo no requiere que el programa comience en el primer registro del archivo.

Los archivos de acceso directo son archivos con llave. Asocian un registro con un valor de llave específico y una ubicación de almacenamiento en particular. Todos los registros son almacenados por llave en direcciones más que por su posición. En otras palabras, si el programa conoce la

llave del registro, puede determinar la dirección de la ubicación del registro y consultarlo en forma independiente de cualquier otro registro dentro del archivo.

2.3.2.1 DIRECCIONAMIENTO DIRECTO

Considerando que los elementos a almacenar sean registros de cheques. El acceso directo de los registros se lleva a cabo utilizando un área de almacenamiento que tiene un espacio reservado para cada cheque. El sistema utiliza el número de cheque como llave física del registro, sabe que existe la llave y la utiliza (a diferencia de un archivo secuencial sin llave) para almacenar los datos. Por lo tanto, el número de cheque 1248 es almacenado en la dirección 1248, que es la ubicación reservada para el cheque de este número. Para consultar este cheque del almacenamiento en un sistema de computadora, se le indica al programa que utilice el 1248 como la llave de búsqueda. Sabe que la llave sirve como dirección y va directamente a la ubicación asignada para el registro con la llave 1248 y consulta este registro. Esta es una característica atractiva de la organización directa: los registros se consultan mucho más rápido que cuando el archivo debe ser rastreado desde el principio.

2.3.2.2 REQUERIMIENTOS PARA EL DIRECCIONAMIENTO DIRECTO

El tipo de acceso directo que utiliza la llave de registro como la dirección de almacenamiento se llama direccionamiento directo. Cuando se puede utilizar es simple y rápido, sin embargo, los requerimientos de este método limitan a menudo su uso. El direccionamiento directo debe tener un conjunto de datos en donde:

1. La llave del conjunto de datos esté en un orden cerrado y ascendente con pocos valores que no se utilicen y que signifiican espacios de almacenamiento desperdiciado. Por lo tanto, se requieren pocos intervalos abiertos de los valores de llave.

2. Las llaves del registro conforman a los números de las direcciones del almacenamiento; en otras palabras, existe una dirección del almacenamiento para cada valor de la llave actual o posible del archivo y no hay valores de llave duplicados.

En una lista de llaves cerrada, para cada valor de la

llave existe un espacio de almacenamiento con una direccion que es equivalente a esta llave. Los valores de las llaves están también en una secuencia cercana; existen pocas omisiones en la secuencia. Esto es importante en el diseño del archivo, dado que, cuando el almacenamiento es asignado para este archivo, comienza en el valor menor de la llave y se extiende hasta el valor mayor de ésta. Si el intervalo de las llaves incluye omisiones en el rango, esto crea espacio de almacenamiento desperdiciado. El almacenamiento se debe ubicar incluso cuando no se va a utilizar.

2.3.3 ORGANIZACIÓN INDEXADA

Una tercer manera de tener acceso a los registros almacenados en el sistema es a través de un índice. La forma básica de un índice incluye una llave de registro y la dirección de almacenamiento para éste. Para encontrar un registro cuando no se conoce la llave de almacenamiento (al igual que con las direcciones), es necesario rastrear los registros; sin embargo, si se utiliza un índice el rastreo será más rápido, dado que lleva menos tiempo buscar en un índice que en un archivo de datos en su totalidad.

2.3.3.1 CARACTERÍSTICAS DE UN ÍNDICE

Un índice es un archivo separado del archivo maestro al cual pertenece. En el índice, cada registro contiene dos datos: la llave del registro y una dirección del almacenamiento (no duplica cada dato en el registro del archivo maestro).

Para encontrar un registro específico cuando el archivo se almacena bajo una organización indizada, el índice primero se rastrea para encontrar la llave del registro que se desea. Cuando se encuentra, la dirección correspondiente del almacenamiento se anota y entonces el programa da acceso al registro directamente. Este método utiliza un rastreo secuencial de este índice, seguido de un acceso directo para el registro apropiado. El índice ayuda a dar más velocidad a la búsqueda, comparada con la de un archivo secuencial, pero es mucho más lento que el direccionamiento directo.

2.4 MANIPULACIÓN DE ARCHIVOS EMPLEANDO LENGUAJE C

Las funciones de entrada y salida (E/S) de las librerías estándar del lenguaje C, permiten leer y escribir datos a, y desde, archivos y dispositivos. C tiene disponibles los tres tipos siguientes de funciones de E/S:

1. Funciones estándar de E/S (Stream I/O).
2. Funciones de E/S de bajo nivel (Low-level I/O).
3. Funciones para la consola y puertos de E/S

La característica fundamental de las funciones estándar de E/S, es que la E/S en el procesamiento de archivos, se realiza a través de un buffer o memoria intermedia.

También permiten la E/S con formato.

La utilización de un buffer o memoria intermedia para realizar las operaciones de E/S es una técnica, implementada en software, diseñada para hacer las operaciones de E/S más eficientes. Un buffer es un área de datos en la memoria (RAM), asignada por el programa que abre el archivo. La utilización de buffers en operaciones de E/S, reduce el número de accesos al dispositivo físico (disco por ejemplo) asociado con el archivo, necesarios para la transferencia de información entre el programa y el archivo; un acceso a un dispositivo físico consume mucho más tiempo que un acceso a la memoria (RAM). Cuando un archivo no tiene asociado un buffer, cada byte escrito a, o leído desde, el archivo es físicamente transferido en el momento de la operación. En cambio, cuando un archivo tiene asociado un buffer, todas las operaciones de E/S requeridas son servidas desde ese buffer; la transferencia física de datos se hace en múltiplos del tamaño del buffer.

Las funciones estándar de E/S, como su nombre indica, proporcionan la forma más normal de E/S en un programa C. Permiten escribir y leer datos de un archivo, de las siguientes formas:

Primera, los datos pueden ser escritos o leídos caracter a caracter con las funciones fputc( ) y fgetc( ).

Segunda, los datos pueden ser escritos y leídos palabra a palabra con las funciones putw( ) y getw( ). Se entiende por palabra, palabra maquina o valor de tipo int.

Tercera, los datos pueden ser escritos y leídos como cadenas de caracteres con las funciones fputs( ) y fgets( ).

Cuarta, los datos pueden ser escritos y leidos con formato, con las funciones fprintf( ) fscanf( ).

Quinta, los datos pueden ser escritos y leídos como registros o bloques, (ésto es, como un conjunto de datos de longitud fija, tales como estructuras o elementos de un array), con las funciones fwrite( ) y fread( ).

2.4.1 ABRIENDO UN ARCHIVO

Para poder escribir o leer sobre un archivo, primeramente hay que abrirlo con las funciones fopen(), fdopen(), o freopen(). El archivo puede ser abierto para leer, para escribir o para leer y escribir; y puede ser abierto en modo texto o en modo binario.

La necesidad de dos modos diferentes, es por las incompatibilidades existentes entre C y MS-DOS ya que C fue diseñado originalmente para el sistema operativo UNIX. El modo texto es para ver los archivos como si estuvieran bajo UNIX; y el modo binario, para verlos como si estuvieran bajo MS-DOS.

En modo texto, un final de línea es representado en C por un único caracter ('\n'), pero en un archivo de MS-DOS es representado por dos caracteres (CR+LF).

Esto significa que, bajo MS-DOS, cuando C escribe en un archivo convierte el caracter '\n', en dos caracteres CR + LF; y cuando C lee de un archivo y encuentra los caracteres CR + LF, los convierte a '\n'; y cuando encuentra un Ctrl+Z lo interpreta como un EOF.

En modo binario estas conversiones no tienen lugar.

Cuando un programa comienza su ejecución, son abiertos automáticamente cinco archivos, que se corresponden con dispositivos. Estos archivos y los dispositivos asociados a ellos son:

Nombre	Dispositivo
stdin	Dispositivo de entrada estándar
stdout	Dispositivo de salida estándar
stderr	Dispositivo de error estándar
stdaux	Dispositivo auxiliar estándar
stdprn	Dispositivo de impresión estándar

De estos cinco, dos de ellos, el dispositivo de serie y el dispositivo de impresión, dependen de la configuración de la maquina, por lo tanto pueden no estar presentes.

FUNCIONES PARA ABRIR UN ARCHIVO

fopen(path, acceso)

Esta función abre el archivo especificado por path. El argumento acceso especifica como es abierto el archivo.

A las formas de acceso mencionadas, se les puede añadir un caracter t o b (rb, ab+, etc.), para indicar si el archivo se abre en modo texto o en modo binario. Si t o b no se especifican, el modo es definido por la variable global _fmode de C (texto por defecto).

La función fopen( ) devuelve un apuntador a una estructura de tipo FILE, la cual se corresponde con el buffer asociado con el archivo abierto. Un apuntador nulo indica un error. Este apuntador es utilizado por las funciones de C, para leer y escribir datos en un archivo. Por eso, antes de utilizar la función fopen( ), debemos definir un apuntador de tipo FILE, tipo que está declarado en el archivo stdio.h.

Ejemplo:

```
#include <stdio.h>
FILE *fp;

fp = fopen("datos","w");
```

Este ejemplo indica que se abre el archivo datos para escribir, y que será referenciado por el apuntador fp.

Debe especificarse el archivo de cabecera stdio.h, porque contiene la declaración de FILE.

fdopen(num, acceso)

Asocia un stream con un número de archivo, num, resultante de haber abierto el archivo con la función a nivel de sistema open( ). Esto nos permite procesar el archivo como si hubiera sido abierto por la función fopen( ). La descripción para el argumento acceso, es la misma que la dad en la función fopen( ).

```
#include <stdio.h>

FILE *fdopen(int num, char *acceso);
```

La función, devuelve un apuntador al archivo abierto por ella. Un apuntador nulo indica un error.

Ejemplo:

```
#include <stdio.h>
#include <fcntl.h>
#include <io.h>

FILE *fp;
int nf;

nf = open("datos",O_RDONLY);
fp = fdopen(nf,"r");
```

freopen(path, acceso, fp)

Esta función cierra el archivo actualmente asociado con el apuntador fp; y reasigna fp, al archivo indicado por path. Es utilizada para redireccionar stdin, stdout, stderr, stdaux y stdprn, a archivos especificados por el usuario. La descripción para el argumento acceso, es la misma que la dada en la función fopen( ).

```
#include <stdio.h>

FILE *freopen(const char *path, const char *acceso,
FILE *fp)
```

La función freopen( ) devuelve un apuntador al archivo abierto nuevamente. Si ocurre un error, el archivo original es cerrado y se devuelve un apuntador nulo.

Ejemplo:

```
#include <stdio.h>
FILE *fp;
fp = freopen("datos","w",stdout);
```

Este ejemplo, reasigna stdout al archivo llamado datos. Ahora lo que se escriba en stdout, será escrito en datos.

2.4.2 CERRANDO UN ARCHIVO

Después de haber finalizado el trabajo con un archivo, éste debe cerrarse con la función fclose( ). Si un archivo no se cierra explícitamente, es cerrado automáticamente cuando finaliza el programa. Sin embargo, es aconsejable cerrar un archivo cuando se ha finalizado con él, ya que el número de archivos abiertos al mismo tiempo está limitado.

FUNCIONES PARA CERRAR UN ARCHIVO

fclose(fp)

Esta función cierra el archivo apuntado por fp. Cualquier dato en el buffer asociado, se escribe en el archivo antes de cerrarlo.

```
#include <stdio.h>

int fclose(FILE *fp);
```

Si el archivo es cerrado, la función fclose( ) devuelve un cero. Si ocurre un error entonces devuelve un EOF.

fcloseall( ) (Solo para MS-DOS)

Esta función cierra todos los archivos abiertos, excepto los archivos estándar.

```
#include <stdio.h>

int fcloseall(void);
```

La función fcloseall( ) devuelve un entero igual al número de archivos cerrados. Si ocurre un error, entonces devuelve un EOF.

2.4.3 LEYENDO Y ESCRIBIENDO DATOS

Las operaciones de lectura y escritura siempre empiezan en una posición perfectamente definida en todo momento. A esta posición le denominamos apuntador de lectura escritura (L/E). Cada vez que se efectúa una operación de lectura o escritura, el apuntador de L/E avanza a la siguiente posición. Cuando un archivo se abre, el apuntador de L/E es posicionado automáticamente al principio del archivo, excepto cuando se abre para añadir información, en tal caso,

es posicionado al final del archivo.

Para detectar el final de un archivo, disponemos de la función feof( ).

El apuntador de L/E puede ser situado en cualquier parte del archivo, utilizando la función fseek( ). Para situarse al principio de un archivo se dispone de la función rewind( ); para determinar en qué posición nos encontramos, tenemos la función ftell( ).

ENTRADA/SALIDA CARACTER A CARACTER

fputc(car, fp)

Esta función escribe un caracter car en la posición indicada por el apuntador de L/E del archivo apuntado por fp.

```
#include <stdio.h>

int fputc(int car, FILE *fp);
```

La función fputc( ), devuelve el caracter escrito o un EOF si ocurre un error. Sin embargo, dado que EOF es un valor aceptado por car, se debe utilizar la función ferror( ) para verificar si ha ocurrido un error.

La macro putc( ) desarrolla la misma función y de la misma forma que la función fputc( ).

fgetc(fp)

Esta función lee un caracter, del archivo apuntado por fp, de la posición indicada por el apuntador de L/E.

```
#include <stdio.h>

int fgetc(FILE *path);
```

La función fgetc( ) devuelve el caracter leído o EOF si ocurre un error o si se detecta el final del archivo. No obstante, dado que EOF es un valor aceptado, se debe utilizar la función ferror( ) o feof( ) para distinguir si se ha detectado el final del archivo o si ha ocurrido un error.

La macro getc desarrolla la misma función y de la misma forma que la función fgetc( ).

ENTRADA/SALIDA PALABRA A PALABRA

putw(entb, fp)

Esta función escribe un valor binario entb de tipo int, en el archivo apuntado por fp.

```
#include <stdio.h>

int putw(int entb, FILE *fp);
```

La función putw( ) devuelve el valor escrito o un EOF si ocurre un error. No obstante, dado que EOF es un valor válido, se debe utilizar la función ferror( ) para verificar si ha ocurrido un error.

getw(fp)

Esta función lee el siguiente valor binario de tipo int, del archivo apuntado por fp y avanza el apuntador de L/E al siguiente valor no leído.

```
#include <stdio.h>

int getw(FILE *fp)
```

La función getw( ) devuelve el valor leído o un EOF si ocurre un error o se detecta el final del archivo. No obstante, dado que EOF es un valor válido, se debe utilizar la función ferror( ) o feof( ) para distinguir si se ha detectado el final del archivo o si ha ocurrido un error.

ENTRADA/SALIDA DE CADENA DE CARACTERES

fputs(cadena, fp)

Esta función copia la cadena de caracteres, cadena, en el archivo apuntado por fp. La terminación '\0' no se copia.

```
#include <stdio.h>

int fputs(const char *cadena, FILE *fp);
```

La función fputs( ), si no hay error, devuelve un cero. En caso contrario, devuelve un valor distinto de cero.

Para recuperar de una forma sencilla la información escrita en el archivo, es aconsejable copiar el caracter '\n' después de cada cadena escrita sobre el archivo.

fgets(cadena, n, fp)

Esta función lee una cadena de caracteres, cadena, del archivo apuntado por fp. La terminación '\0' es añadida automáticamente a la cadena leída. Entendiendo como cadena desde la posición actual dentro del archivo, hasta el primer caracter nueva línea ('\n') incluido éste, hasta el final del archivo, o hasta que el número de caracteres sea igual a n-1.

```
#include <stdio.h>

char *fgets(char *cadena, int n, FILE *fp);
```

La función fgets( ) devuelve la cadena leída. Si el valor devuelto es NULL, quiere decir que ha ocurrido un error o que se ha detectado un EOF. Utilizar feof( ) o ferror( ) para determinar lo que ha ocurrido.

ENTRADA/SALIDA CON FORMATO

fprintf(fp, formato, arg...)

Esta función escribe sus argumentos (arg) en el archivo apuntado por fp, con el formato especificado.

```
#include <stdio.h>

int fprintf(FILE *fp, const char *formato, arg...);
```

La función fprintf( ) devuelve el número de caracteres escritos o un valor negativo si ocurre un error.

fscanf(fp, formato, arg...)

Esta función lee sus argumentos (arg) del archivo apuntado por fp, con el formato especificado. Cada argumento arg, debe ser un apuntador a una variable en la que queremos almacenar el valor leído. El tipo de cada una de éstas variables debe corresponderse con la especificación de

formato indicada para cada variable.

```
#include <stdio.h>

int fscanf(FILE *fp, const char *formato, arg...);
```

La función fscanf( ) devuelve el número de argumentos que han sido leídos y asignados. Si el valor devuelto es un cero, significa que no se han asignado valores; y si es un EOF; significa que se ha detectado el final del archivo.

ENTRADA/SALIDA UTILIZANDO REGISTROS O BLOQUES

fwrite(buffer, n, c, fp)

Esta función escribe hasta c elementos de longitud n bytes, almacenados en el buffer, en el archivo apuntado por fp.

```
#include <stdio.h>

size_t fwrite(const void *buffer, size_t n, size_t c,
FILE *fp);
```

La función fwrite( ) devuelve el número de elementos escritos, que puede ser menor que c si ocurre un error.

fread(buffer, n, c, fp)

Esta función lee hasta c elementos de longitud n bytes, del archivo apuntado por fp, y los almacena en el buffer.

```
#include <stdio.h>

size_t fread(void *buffer,size_t n,size_t c,FILE *fp);
```

La función fread( ) devuelve el número de elementos leídos, que puede ser menor que c si ocurre un error. Utilizar las funciones feof( ) o ferror( ), para distinguir si se ha detectado el final del archivo o si ha ocurrido un error.

Si n o c son cero, fread( ) devuelve un cero y el contenido del buffer permanece igual.

2.4.4 DETECCIÓN DE ERRORES

Cuando en una operación sobre un archivo ocurre un error, éste puede ser detectado por la función ferror( ). Cuando ocurre un error, el indicador de error permanece activado hasta que el archivo se cierra, a no ser que utilicemos la función clearerr( ) o rewind( ) para desactivarlo explícitamente.

FUNCIONES PARA DETECCIÓN DE ERRORES

ferror(fp)

Esta función verifica si ha ocurrido un error en una operación con archivos. Cuando ocurre un error, el indicador de error para ese archivo se pone activo y permanece en este estado, hace que sea ejecutada la función de clearerr( ).

```
#include <stdio.h>

int ferror(FILE *fp);
```

La función ferror( ), devuelve un cero si no a ocurrido un error y un valor distinto de cero en caso contrario.

clearerr(fp)

Esta función desactiva el indicador de error y el indicador de fin de archivo (EOF) para un determinado archivo, poniendolos a valor cero.

```
#include <stdio.h>

void clearerr(FILE *fp);
```

feof(fp)

Esta función indica si se ha alcanzado el fin del archivo apuntado por fp.

```
#include <stdio.h>
int feof(FILE *fp);
```

La función feof( ) devuelve un valor distinto de cero, cuando se intenta leer un elemento del archivo y nos encontramos con un eof (end of file del archivo). En caso contrario devuelve un cero.

2.4.5 ACCESO SECUENCIAL Y ACCESO ALEATORIO

Exister tres organizaciones de archivos básicas, de cuya combinación se derivan multitud de organizaciones posibles. Estas son:

- Secuencial
- Aleatoria
- Secuencial indexada

En cada caso, se elegirá una u otra en función de las características de los soportes y del modo de acceso requerido.

En cuanto a los tipos de acceso, distinguimos dos:

- Acceso secuencial
- Acceso aleatorio o directo

Se habla de acceso secuencial cuando se van accesando posiciones sucesivas sucesivas, esto es tras acceder a la posición N, se accede a la posición N + 1; y se habla de acceso aleatorio o directo, cuando se accede directamente a la posición deseada, sin necesidad de acceder a las posiciones que le preceden.

Un archivo en C está organizado secuencialmente y el acceso puede ser secuencial o aleatorio si utilizamos la función fseek( ).

ACCESO ALEATORIO A UN ARCHIVO

fseek(fp, desp, pos)

Esta función mueve el apuntador de L/E asociado con el archivo apuntado por fp, a una nueva localización desplazada desp bytes de la posición dada por el argumento pos.

```
#include <stdio.h>
int fseek(FILE *fp, long desp, int pos);
```

pos	Definición
SEEK_SET	Principio del archivo
SEEK_CUR	Posición actual del apuntador de L/E
SEEK_END	Final del archivo.

La función fseek( ) devuelve un cero si no se ha producido un error y un valor distinto de cero en caso contrario.

ftell(fp)

Esta función da como resultado la posición actual del apuntador de L/E, dentro del archivo apuntado por fp. Esta posición es relativa al principio del archivo.

```
#include <stdio.h>

long ftell(FILE *fp);
```

La función ftell( ) devuelve la posición actual del apuntador de L/E, o el valor de -1L si ocurre un error.

rewind(fp)

Esta función pone el apuntador de L/E del archivo apuntado por fp, al comienzo del mismo.

```
#include <stdio.h>

void rewind(FILE *fp);
```

Una llamada a esta función equivale a:

```
(void) fseek(fp, 0L, SEEK_SET);
```

Con la excepción de que rewind( ) desactiva los indicadores de error y de fin de archivo, y fseek( ) no.

_fsopen(path, modo, sh) (Sólo DOS y OS/2)

Esta función abre el archivo especificado por path y permite compartirlo por otros procesos. El argumento acceso especifica cómo es abierto el archivo y el argumento sh es una constante que indica los atributos para compartir el el archivo.

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

FILE *_fsopen(char *path, char *acceso, int sh);
```

acceso: r, w, a, r+, w+, a+
Para indicar el tipo de conversión, pueden añadirse los caracteres t o b.

sh: SH_COMPAT, SH_DENYNO, SH_DENYRD, SH_DENYRW, SH_DENYWR

sh	Significado
SH_COMPAT	Modo compatible.
SH_DENYRW	Acceso denegado al archivo, para leer y escribir.
SH_DENYRD	Acceso denegado al archivo, para leer.
SH_DENYWR	Acceso denegado al archivo, para escribir.
SH_DENYNO	Permite el acceso al archivo, para leer y escribir.
O_NOINHERIT	El archivo no es heredado por un proceso hijo.

La función _fsopen( ) devuelve un apuntador a una estructura de tipo FILE, la cual se corresponde con el archivo abierto. Un apuntador nulo indica un error.

Si SHARE.COM (o SHARE.EXE) no está instalado, MS-DOS ignora el modo compartido.

2.5 HOJAS ELECTRÓNICAS

La forma más facil de imaginar lo que es una hoja electrónica u hoja de cálculo es examinando una carpeta de varias columnas de un contador. Esta, contiene hojas de papel que están divididas en filas y columnas. Usualmente, hay una columna ancha a la izquierda para los rótulos del texto que describen lo que se encuentra en cada fila. A la derecha de la columna de rótulos hay varias columnas para introducir números. Estas carpetas son utilizadas manualmente para listar ventas o gastos, para preparar presupuestos o planes financieros, o para cualquier otra tarea similar.

Una hoja de cálculo, es una carpeta electrónica de varias columnas, sólo que ésta es mucho más flexible. Su estructura de filas y columnas proporciona un medio adecuado para analizar datos organizados.

Considerando un presupuesto o plan financiero de una empresa, cada fila puede ser una cuenta, un elemento de ingreso o de gasto. Cada columna puede ser un período de tiempo: meses trimestres o años. Cada hoja puede ser un departamento o almacén. Sumando las columnas se obtiene el total de una cuenta, sumando las filas, se obtendrá el total para un período de tiempo y si se suman las hojas, se obtendrá el total de la compañía.

Las filas, columnas y hojas proporcionan una red de tres dimensiones con filas horizontales, columnas verticales y láminas de hojas.

La versión 3 de Lotus 1-2-3 permite disponer de 256 hojas, conteniendo cada una de ellas 8192 filas y 256 columnas. Cada columna puede tener hasta 240 caracteres de ancho, y las palabras y números pueden entremezclarse como sea necesario.

2.5.1 CELDAS

Cuando se introduce la información en una hoja, ésta es almacenada en una posición específica. Se sabe que esa posición es la fila, columna y hoja en que se encuentra esa información.

Una sola dirección, la intersección de una fila, columna y una hoja se le
denomina una celda. Se puede pensar en una hoja como una gran colección de celdas.

2.5.2 VALORES Y RÓTULOS

Una celda puede contener sólo dos tipos de información: valores y rótulos. Los valores son números o fórmulas; pueden ser formateados de muchas formas, incluyendo formatos monetarios, porcentajes, fechas u horas. Los rótulos son texto: una cadena de letras y números. Pueden ser utilizados como títulos, como los rótulos de fila y columna, y para almacenar una serie de ordenes (denominada una macro) de la hoja.

Basándose en el primer caracter que se teclea 1-2-3 determina si la entrada que se está realizando es un rótulo o un valor. Si se teclea un número (del 0 al 9) o uno de los símbolos numéricos + - @ (. $ # como primer caracter 1-2-3 considera que se está introduciendo un valor. Si se teclea una letra (de la A a la Z), un espacio, o cualquier símbolo especial no considerado numérico, entonces 1-2-3 considera le entrada de un rótulo.

Todos los rótulos tienen un prefijo de rótulo delante de ellos, colocado automáticamente por 1-2-3 o por el usuario. Los símbolos de prefijo de rótulo son ' " ^ \ |. Si 1-2-3 detecta un rótulo, éste coloca automáticamente el prefijo de rótulo delante de sus entrada. Se puede realizar una entrada numérica como un rótulo poniendo un prefijo de rótulo delante de ésta. Los valores pueden ser introducidos como un número entre 10 $^{-99}$ y 10 ^{99} y ser formateados de diversas formas.

2.5.3 RANGOS Y NOMBRES DE RANGO

Denominamos rango a un grupo rectangular de celdas adyacentes. Un rango no puede ser un grupo con forma de L, sino que debe ser un rectángulo completo o un bloque rectangular.

Los rangos son utilizados en muchas ordenes de 1-2-3 para especificar un grupo de celdas sobre las que se desea que trabaje. Para especificar un rango en una orden, se puede utilizar un par de direcciones para las celdas primera y última del rango (las esquinas superior izquierda y la inferior derecha) o utilizar un nombre de rango definido anteriormente. Un nombre de rango es una palabra o frase que tiene una longitud de hasta quince caracteres e identifica a dicho rango.

2.5.4 FÓRMULAS Y *FUNCIONES*

La posibilidad de introducir rótulos y valores no resulta muy útil a menos que se pueda hacer algo con ellos. Se necesita poder sumar columnas de números, calcular porcentajes y trabajar con otras relaciones matemáticas. Las fórmulas representan el medio que permite hacer esto. Las fórmulas operan sobre números, otras fórmulas o texto. Cuando una fórmula utiliza texto, se le denomina fórmula de cadena y puede contener el operador de cadena (para la concatenación). Las fórmulas que contiene los operadores aritméticos estándar + - * / ^ se denominan fórmulas numéricas. Las fórmulas que contienen los operadores lógicos = < > <= >= <> #AND# #OR# #NOT# se denominan fórmulas lógicas.

Las fórmulas utilizan la notación algebraica estándar, con paréntesis anidados si fuera necesario. Siempre comienzan con un número o uno de estos símbolos numéricos: + - @ . ($ #.

Las fórmulas pueden tener una longitud de hasta 512 caracteres, pero no puede contener espacios, excepto dentro de un literal. Un literal es cualquier conjunto de letras, números o símbolos que van entre comillas. Las fórmulas utilizan normalmente datos de otras posiciones de la hoja. esto se realiza por medio de una dirección de celda, un rango de celdas o un nombre de rango.

Ejemplo de fórmulas:

Fórmula	Contenido de la celda que alberga la fórmula
+B5	El contenido de B5
+C6-C7El	Resultado de restar el contenido de C7 al contenido de C6
0.45*1590	El producto de 0.45 multiplicado por 1590
+subtotal*impuesto	El producto de los rangos denominados "subtotal" e "impuesto"
+"Querido">>&D5	La combinación de la cadena literal "Querido" con el contenido de D5, (las comillas definen la cadena literal, un caracter tras otro).
+fecha <= hoy	El valor es 1 (verdadero) si un rango denominado "fecha" es igual o menor que un rango denominado "hoy", si no es asi contendra el valor 0 (falso)

2.5.5 PRECEDENCIA EN LOS CÁLCULOS

Cuando 1-2-3 calcula o evalúa una fórmula, lo hace en un orden particular. Este orden es determinado por el número de prioridad de los operadores que se están utilizando y los paréntesis colocados en la fórmula. Los operadores con un número de prioridad inferior se realizan antes en el cálculo.

Operador	Descripción	Prioridad
^	Exponenciación	1
- +	Negativo o positivo	2
*	Multiplicación	3
/	División	3
+	Suma	4
-	Resta	4
=	Igual que	5
<	Menor que	5
>	Mayor que	5
<=	Menor o igual que	5
>=	Mayor que o igual a	5
<>	Distinto a	5
#NOT#	No lógico	6
#AND#	Y lógico	7
#OR#	O lógico	7
&	Concatenación	7

2.5.6 FUNCIONES @

Lotus 1-2-3 posee algunas fórmulas implementadas denominadas funciones @ que pueden ser utilizadas dentro de otras fórmulas o pueden utilizarse solas. Las funciones @ siempre comienzan con el signo @ e incluyen cálculos numéricos, lógico, de cadenas, financieros y estadísticos. Una función @ siempre es considerada como un valor, tanto si está operando sobre cadenas como sobre números.

Ejemplos de funciones @:

@SUM(A4..A7)	Suma A4, A5, A6 y A7
@TODAY	Visualiza la fecha activa
@PI*G3^2	Calcula el área de un círculo cuyo radio está en G3
@PV(C4,F6,B1)	Calcula el valor actual de una serie de inversiones iguales cuyo valor está contenido en C4, con un interés contenido en F6, para el número de periodos indicado en B1
@RIGHT(T22,5)	Visualiza los cinco caracteres del extremo derecho de una cadena contenida en T22

2.5.7 GRÁFICOS Y DIAGRAMAS

Un gráfico le permite dar un significado visual a un conjunto de números para mostrar la diferencia o igualdad entre ellos y mostrar las tendencias que están produciendo. Mucha gente observa que puede conseguir una información más rapida y fácil de los gráficos, o incluso que puede ver cosas en los gráficos que no vería en los números que los producen.

Con 1-2-3, un gráfico es una representación gráfica de uno o más rangos de una o más hojas. Si se tiene un rango en una hoja que contenga ventas de la compañía por mes, se puede producir un gráfico de líneas que hace que sea más fácil mostrar las fluctuaciones en las ventas mensuales. Proporcionando los rangos para los gastos y ganancias por mes, se pueden producir líneas adicionales en el mismo gráfico y no sólo decir si está subiendo o bajando, sino como se está moviendo en relación de uno con el otro.

La Versión 3 de 1-2-3 proporciona siete tipos principales de gráficos:

- Líneas
- Barras
- Barras apiladas
- Sectores
- XY
- Alto-bajo-cerrado-abierto
- Mixta

2.5.8 FORMATO DE UNA HOJA ELECTRÓNICA

Lotus 1-2-3 utiliza uno de los más complejos formatos de archivos que existen en el mercado. La información almacenada en un archivo de 1-2-3 es como la de un programa de computadora. Esta se encuentra organizada como un código de operación, seguido por un registro delimitador de longitud, el cual, es a su vez seguido por los datos.

Cada código de operación es un número hexadecimal de dos bytes. Con el byte menos significativo (LSB) en la primera posición. La longitud de éste, está dada por un número hexadecimal, también con el byte menos significativo en la primera posición. Los datos son una serie de cero o más bytes hexadecimales, dados para cada registro como el byte 0, byte 1, etc.

Todo archivo de 1-2-3, cuenta con una serie de códigos de operación que por sí mismos le dan el formato a una hoja de cálculo. Dichos códigos se muestran a continuación.

00h (00d) BOF

longitud	2 bytes	
Significado	Begining of file (inicio de archivo)	
# Byte	0-1 Número de versión del formato de archivo 1028(0404h) para un archivo de 1-2-3 1030(0406h) para un archivo de 1-2-3/2	

06h (06d) RANGE

longitud	8 bytes	
Significado	El rango de celdas a ser utilizadas en el archivo	
# Byte	0-1	columna inicial
	2-3	renglón inicial
	4-5	columna final
	6-7	renglón final

El rango describe el área activa, si un archivo fue creado con el comando FILE SAVE y el rango seleccionado si hubiese sido creado con FILE XTRACT. En ambos casos, los renglones y columnas hallados en blanco (vacías) son removidos. Si no existen datos en el rango, el valor de la columna de inicio es ajustado a -1.

Cuando se crea un archivo de hoja de cálculo de manera externa, es necesario grabar este registro tan cerca como sea posible del registro BOF (código de operación 00).

96h (150d) CPI

longitud	variable	
Significado	Cell Pointer Index (índice de apuntadores a celdas) Este registro es una lista de columnas que contienen una o más celdas activas	
# Byte	0-1	Número de columna (como entero)
	2-3	El número de renglón más pequeño de una celda activa.
	4-5	El número de renglón más grande de una celda activa.

2Fh (47d) CALCCOUNT

longitud	1 byte	
Significado	Contiene un contador de iteraciones que establece el número de veces que 1-2-3 recalculará las fórmulas El valor de default es uno, pero puede ser inicializado de 1 a 50.	
# Byte	0	número de iteraciones

02h (02d) CALCMODE

longitud	1 byte	
Significado	Modo de recálculo	
# Byte	0	00h para recálculo manual
		FFh para recálculo automático

Recálculo manual: Las fórmulas sólo son recalculadas cuando se pulsa CALC (F9). Si se realizan cambios en las celdas y no se recalcula la hoja de trabajo activa , 1-2-3 visualiza el indicador CALC en la línea de estado, en la parte inferior de la pantalla.

Recálculo automático: 1-2-3 recalcula todas las celdas afectadas por cualquier cambio realizado en una celda

03h (03d) CALCORDER

longitud	1 byte	
Significado	Método de recálculo	
# Byte	0	00h orden natural de recálculo
		01h recálculo por columnas
		FFh recálculo por renglones

Natural: Cuando se encuentra preparado para recalcular una fórmula 1-2-3 recalcula cualquier fórmula que haga referencia a la celda. AA1 contiene una fórmula que hace referencia a la celda a AD1, que también contiene una fórmula, entonces AD1 se calculará antes que AA1.

Por columnas: Recalcula fórmulas empezando en la celda AA1, descendiendo por la hoja de trabajo columna a columna a través de los archivos activos. En el ejemplo dado anteriormente, la celda AA1 se recalculará antes que la fórmula de la celda AD1.

Por renglones: Recalcula fórmulas empezando en la celda AA1, yendo por la hoja de trabajo renglón a renglón a través de los archivos activos. En el ejemplo anterior, la celda AA1 sería calculada antes que la fórmula de la celda AD1.

04h (04d) SPLIT

longitud	1 byte	
Significado	Muestra el tipo de split en la ventana	
# Byte	0	00h sin split
		01h split vertical
		FFh split horizontal

05h (05d) SYNC

longitud	1 byte	
Significado	Sincroniza la ventana con los movimientos del cursor	
# Byte	0	00h ventana no sincronizada FFh sincronizada

07h (07d) WINDOW1

longitud	31 bytes	
Significado	Describe la ventana numero 1	
# Byte	0-1	columna actual de la posición del cursor
	2-3	renglón actual de la posición del cursor
	4	byte de formato de la celda (Ver la tabla 1.1)
	5	no usado (puede ser 00h)
	6-7	ancho de la columna
	8-9	número de columnas en la ventana
	10-11	número de renglones en la ventana
	12-13	columna más a la izquierda
	14-15	renglón superior
	16-17	numero de columnas para titulos
	18-19	número de renglones para titulos
	20-21	columna izquierda de titulos
	22-23	renglón superior de titulos
	24-25	ancho de las columnas de margen
	26-27	ancho de los renglones de margen
	28-29	ancho de la ventana
	30	no usado (puede ser 00h)

08h (08d) COLW1

longitud	3 bytes	
Significado	Define el ancho de una columna	
# Byte	0-1	Número de la columna
	2	ancho

64h (100d) HIDVEC1

longitud	32 bytes
Significado	Registro de columnas ocultas

Estos 32 bytes corresponden a un arreglo de 256 bits. Cada bit representa una columna de la hoja de trabajo Un bit con valor de 1 representa una columna oculta. El arreglo esta dispuesto del byte menos significativo al byte más significativo, byte 0 al byte 31. El bit 0 del byte 0 representa la columna A. Estos registros normalmente representan la hoja completa, pero cuando la ventana está rebanada, representan las columnas de la ventana # 1.

18h (24d) TABLE

longitud	25 bytes	
Significado	Describe el rango de una tabla	
# Byte	0	0 no hay tabla 1 existe una tabla 1 2 existe una tabla 2
	1-2	Número de columna inicial del rango de la tabla
	3-4	Número de renglón inicial del rango de la tabla
	5-6	Número de columna final del rango de la tabla
	7-8	Número de renglón final del rango de la tabla
	9-10	Celda de entrada 1 columna inicial
	11-12	Celda de entrada 1 renglón inicial
	13-14	Celda de entrada 1 columna final
	15-16	Celda de entrada 1 renglón final
	17-18	Celda de entrada 2 columna inicial
	19-20	Celda de entrada 2 renglón inicial
	21-22	Celda de entrada 2 columna final
	23-24	Celda de entrada 2 renglón final

19h (25d) QRANGE

longitud	25 bytes	
Significado	Describe el rango para una búsqueda	
# Byte	0-1	rango de entrada columna inicial
	2-3	rango de entrada renglon inicial
	4-5	rango de entrada columna final
	6-7	rango de entrada renglon final
	8-9	rango de salida columna inicial
	10-11	rango de salida renglón inicial
	12-13	rango de salida columna final
	14-15	rango de salida renglón final
	16-17	criterio columna inicial
	18-19	criterio renglón inicial
	20-21	criterio columna final
	22-23	criterio renglón final
	24	Comandos 0 no hay comandos 1 ha sido encontrado 2 ha sido seleccionado 3 ha sido borrado 4 es único.

1Ah (26d) PRANGE

longitud	8 bytes	
Significado	Define el rango de impresión	
# Byte	0-1	Columna inicial
	2-3	Renglón inicial
	4-5	Columna final
	6-7	Renglón final.

30h (48d) UNFORMATTED

longitud	1 byte	
Significado	Formato de impresión	
# Byte	0	00h formateada
		01h no formateada

1Ch (28d) FRANGE

longitud	8 bytes	
Significado	Define el rango de llenado	
# Byte	0-1	Columna inicial
	2-3	Renglón inicial
	4-5	Columna final
	6-7	Renglón final

1Bh (27d) SRANGE

longitud	8 bytes	
Significado	Define el rango de ordenamiento	
# Byte	0-1	Columna inicial
	2-3	Renglón inicial
	4-5	Columna final
	6-7	Renglón final

1Dh (29d) KRANGE

longitud	9 bytes	
Significado	Define el rango primario de la tecla de clasificacion	
# Byte	0-1	Columna inicial
	2-3	Renglón inicial
	4-5	Columna final
	6-7	Renglón final
	8	orden
		00h orden descendente
		FFh orden ascendente

23h (35d) KRANGE2

longitud	9 bytes	
Significado	Define el rango secundario de la tecla de clasificación	
# Byte	0-1	Columna inicial
	2-3	Renglón inicial
	4-5	Columna final
	6-7	Renglón final
	8	orden
		00h orden descendente
		FFh orden ascendente

67h (103d) RRANGES

longitud	25 bytes	
Significado	Rangos de regresión lineal	
# Byte	Rango de las variables dependientes	
	0-1	Columna inicial
	2-3	Renglón inicial
	4-5	Columna final
	6-7	Renglón final
	Rango de las variables independientes	
	8-9	Columna inicial
	10-11	Renglón inicial
	12-13	Columna final
	14-15	Renglón final
	Rango de salida	
	16-17	Columna inicial
	18-19	Renglón inicial
	20-21	Columna final
	22-23	Renglón final
	24	Cero banderas interceptadas
		0 no forzadas; cero interceptadas
		-1 forzadas a ser interceptadas en el origen

69h (105d) MATRIXRANGES

longitud	40 bytes	
Significado	Rango de una matriz matemática	
# Byte	0-1	Matriz fuente inversa columna inicial
	2-3	Renglón inicial
	4-5	Columna final
	6-7	Renglón final
	8-9	Matriz destino inversa columna inicial
	10-11	Renglón inicial
	12-13	Columna final
	14-15	Renglón final
	16-17	Rango de la matriz multiplicando columna inicial
	18-19	Renglón inicial
	20-21	Columna final

22-23	Renglón final	
24-25	Rango de la matriz multiplicador columna inicial	
26-27	Renglón inicial	
28-29	Columna final	
30-31	Renglón final	
32-33	Rango de la matriz producto columna inicial	
34-35	Renglón inicial	
36-37	Columna final	
38-39	Renglón final	

20h (32d) HRANGE

longitud	16 bytes	
Significado	Define el rango de distribución	
# Byte	0-1	rango de valores columna inicial
	2-3	rango de valores renglón inicial
	4-5	rango de valores columna final
	6-7	rango de valores renglón final
	8-9	rango de depósito columna inicial
	10-11	rango de depósito renglón inicial
	12-13	rango de depósito columna final
	14-15	rango de depósito renglón final

66h (102d) PARSERANGES

longitud	16 bytes	
Significado	Parse ranges	
# Byte	0-1	Parse input range columna inicial
	2-3	renglon inicial
	4-5	columna final
	6-7	renglón final
	8-9	Parse output range columna inicial
	10-11	renglón inicial
	12-13	columna final
	14-15	renglón final

24h (36d) PROTEC

longitud	1 byte	
Significado	Protección global Proporciona los medios para proteger la hoja de trabajo contra escritura.	
# Byte	0	00h significa proteccion global en off (des habilitada) 01h significa protección global en on (habilitada)

25h (37d) FOOTER

longitud	242 bytes	
Significado	Pie de impresión Permite introducir una línea de texto en la parte inferior de la página (un pie de página).	
# Byte	0-241	Cadena de caracteres ASCII, terminada con un caracter nulo

26h (38d) HEADER

longitud	242 bytes	
Significado	Encabezado de impresión Permite introducir una línea de texto en la parte superior de la página (un encabezado).	
# Byte	0-241	Cadena de caracteres ASCII, terminada con un caracter nulo.

27h (39d) SETUP

longitud	40 bytes	
Significado	Describe el ajuste de impresión. Permite la definición de códigos para controlar la impresora	
# Byte	0-39	Cadena de caracteres ASCII, terminada con un caracter nulo.

28h (40d) MARGINS

longitud	10 bytes	
Significado	Describe el modo de los márgenes de impresión	
# Byte	0-1	margen izquierdo
	2-3	margen derecho
	4-5	longitud de página
	6-7	margen superior
	8-9	margen inferior

29h (41d) LABELFMT

longitud	1 byte	
Significado	Define la alineación de las etiquetas	
# Byte	0	27h izquierda 22h derecha 5Eh centrada

2Ah (42d) TITLES

longitud	16 bytes	
Significado	bordes de impresión	
# Byte		
	0-1	borde de renglones columna inicial
	2-3	borde de renglones renglón inicial
	4-5	borde de renglones columna final
	6-7	borde de renglones renglón final

	8-9	borde de columnas columna inicial
	10-11	borde de columnas renglón inicial
	12-13	borde de columnas columna final
	14-15	borde de columnas renglón final

2Dh (45d) GRAPH

longitud	437 bytes
Significado	Describe la definición actual de gráficas

01h (01d) EOF

longitud	0 bytes
Significado	End of file(Fin de archivo)
# Byte	No existen datos.

0Bh (11d) NAME

longitud	24 bytes	
Significado	Describe el nombre de un rango de celdas	
# Byte	0-15	Cadena de caracteres terminada con un caracter nulo
	16-17	Columna inicial
	18-19	Renglón inicial
	20-21	Columna final
	22-23	Renglón final

Debe existir un registro por cada nombre de rango.

0Dh (13d) INTEGER

longitud	7 bytes	
Significado	Define la celda de un valor numérico entero	
# Byte	0	byte de formato (Ver tabla 1.1)
	1-2	número de columna
	3-4	número de renglón
	5-6	valor entero

Una celda puede contener un valor entero en el rango de -32767 a +32767 (decimal).

0Eh (14d) NUMBER

longitud	13 bytes	
Significado	Define la celda de un valor de punto flotante	
# Byte	0	byte de formato
	1-2	número de columna
	3-4	número de renglón
	5-12	valor reál de 64 bits
	(Ver la tabla 1.2)	

0Fh (15d) LABEL

longitud	variable a 245 bytes	
Significado	Define la celda de una etiqueta	
# Byte	0	byte de formato
	1-2	número de columna
	3-4	número de renglon
	5-245(max)	Cadena de caracteres, terminada con un caracter ASCII nulo (00h), de una longitud de 240 caracteres como máximo

El byte 5 siempre es uno de los siguientes caracteres:

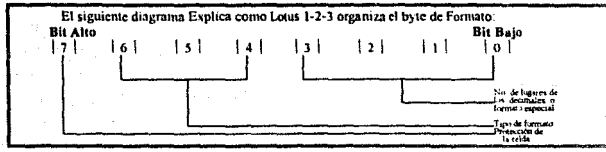
|	Analizador de formato de linea, para un comando de impresión
\	Caracter de repetición
'	Alineación a la izquierda
"	Alineación a la derecha
^	Centrado

10h (16d) FORMULA

longitud	variable a 2064 bytes	
Significado	Define la celda de una fórmula	
# Byte	0	byte de formato
	1-2	número de columna
	3-4	número de renglón
	5-12	valor numérico de fórmula (expresado como un valor long real de 64 bits IEEE)
	13-14	tamaño de la fórmula en bytes
	15-2063	codigo de fórmula; 2048 bytes como máximo

Lotus 1-2-3 analiza las fórmulas en notación polaca inversa (Ver tabla 1.3).

Tabla 1.1. Byte de Formato



Bit	Descripción	Código	Binario	Significado
7	Protección	1	1	Celda Protegida
		0	0	No Protegida
6,5,4	Tipo de Formato	0	000	Fijo
		1	001	Notación Científica
		2	010	Corriente
		3	011	Porciento
		4	100	coma
		5	101	No usada
		6	110	No usada
		7	111	Especial
3,2,1,0	Si el tipo de formato es de 0 a 6, estos bits representan el número de lugares de los decimales	0-15	0000 a 1111	
	Si el tipo de formato es 7	0	0000	+ / -
		1	0001	Formato General
		2	0010	dia-mes-año
		3	0011	dia-mes
		4	0100	mes-año
		5	0101	texto

Tabla 1.2. Formato Interno de números de Punto Flotante para Lotus

Byte	Valor
0	0 es un número positivo -1 es un número negativo 2 es un rango 3 es una cadena
1-2	exponente, como entero signado
3-10	fracción sin signo de 64 bits

Tabla 1.3. Códigos de Operación de Fórmulas y Formatos

Decimal	Hexadecimal	Operación	Notas
0	00	Constante	Seguido por un número real de 64 bits
1	01	Variable	Seguido por 4 bytes de coordenadas, renglón / columna; byte 0-1 = columna, byte 2-3 renglón. El byte menos significativo primero
2	02	Rango	Seguido por un rango de 8 bytes byte 0-1 número de columna inicial byte 2-3 número de renglón inicial byte 4-5 número de columna final byte 6-7 número de renglón final
3	03	Return	Significa fin de la fórmula
4	04	Paréntesis	Indica paréntesis en la fórmula original; Ignorados durante el recálculo
5	05	Constante entera de dos bytes	Seguida por un entero signado de dos bytes
6	06	Constante de caracteres	Longitud variable, terminada con un nulo
8	08	Negacion	Negación
9	09	+	Adición
10	0A	-	Substracción
11	0B	*	Multiplicación
12	0C	/	División
13	0D	^	Exponenciación
14	0E	=	Igual que
15	0F	< >	Diferente de
16	10	< =	Menor o igual que
17	11	> =	Mayor o igual que
18	12	<	Menor que
19	13	>	Mayor que
20	14	# AND #	Y lógico
21	15	# OR #	O lógico
22	16	# NOT #	NO logico
23	17	- unitario	(ignorada durante el recálculo)
31	1F	na	@ na - no aplicable
32	20	err	@ err - error
33	21	abs	@ abs -valor absoluto
34	22	int	@ int - valor entero
35	23	sqrt	@ sqrt - raiz cuadrada
36	24	log	@ log - logaritmo base 10
37	25	ln	@ ln - logaritmo base e (log natural)
38	26	pi	@ pi
39	27	sin	@ sin - seno
40	28	cos	@ cos - coseno
41	29	tan	@ tan - tangente
42	2A	atan2	@ atan2 - arco tangente 4 cuadrantes
43	2B	atan	@ atan - arco tangente 2 cuadrantes
44	2C	asin	@ asin - arco seno
45	2D	acos	@ acos - arco coseno

46	2E	exp	@exp - exponencial anti logaritmo
47	2F	mod	@mod(x,y) - X mod Y
48	30	sel	@choose
49	31	isna	@isna(x) - x = NA then 1
50	32	iserr	@iserr(x) - x = ERR then 1
51	33	false	@false - return 0
52	34	true	@true - return 1
53	35	rand	@rand - genera números random entre 0 y 1
54	36	date	@date - genara el número de días desde 1/1/1990
55	37	today	@today_número de día
56	38	pmt	@pmt - formato de pago
57	39	pv	@pv - valor presente
58	3A	fv	@fv - valor futuro
59	3B	if	@if - booleano
60	3C	day	@day - día del mes
61	3D	month	@month - mes del año
62	3E	year	@year - año
63	3F	round	@round - redondea un número x a d lugares decimales
64	40	time	@time
65	41	hour	@hour
66	42	minute	@minute
67	43	second	@second
68	44		@ISNumber
69	45		@ISString
70	46	length	@length
71	47	value	@value
72	48		@fixed
73	49		@mid
74	4A		@chr
75	4B		@ASCII
76	4C		@find
77	4D		@Datevalue
78	4E		@Timevalue
79	4F		@Cellpointer
80	50	sum	@sum(rango | celdas | constante)
81	51	avg	@avg(rango | celdas | constante)
82	52	cnt	@cnt(rango | celdas | constante)
83	53	min	@min(rango | celdas | constante)
84	54	max	@max(rango | celdas | constante)
85	55	vlookup	@vlookup(x,rango,offset) x=dirección de una celda o constante rango=tabla offset=renglón en una tabla
86	56	npv	@npv(int ,rango) presente neto no valor int=interés rango=flujos de efectivo

87	57	var	@var(rango) - varianza
88	58	std	@std(rango) - desviación estándar
89	59	irr	@irr(suposición,rango) proporción interna de retorno suposición=% estimado rango=flujos de efectivo
90	5A	hlookup	@hlookup(x,rango,offset) x=dirección de una celda o constante rango=tabla offset=renglón en una tabla
91	5B	dsum	función estática de la base de datos
92	5C	avg	función estática de la base de datos
93	5D	dcnt	función estática de la base de datos
94	5E	dmin	función estática de la base de datos
95	5F	dmax	función estática de la base de datos
96	60	dvar	función estática de la base de datos
97	61	dstd	función estática de la base de datos
98	62		@index
99	63		@Cols
100	64		@Rows
101	65		@Repeat
102	66		@Upper
103	67		@Lower
104	68		@Left
105	69		@Right
106	6A		@Replace
107	6B		@Proper
108	6C		@Cell
109	6D		@Trim
110	6E		@Clean
111	6F		@S
112	70		@V
113	71		@Stteq
114	72		@Call
116	74		@Rate
117	75		@Term
118	76		@CTerm
119	77		@SLN
120	78		@SOY
121	79		@DDB
156	9C		@ @ @aafstart
206	CE		@aafunknown
255	FF		@aafend

CAPÍTULO 3

ANÁLISIS Y DISEÑO DEL SISTEMA

3.1 DETERMINACIÓN DE REQUERIMIENTOS

Periódicamente, las diversas empresas del Sector Público rinden un informe financiero a la Dirección General de Planeación Hacendaria, de la Secretaría de Hacienda y Crédito Público. Ésta se encarga de recopilar, almacenar, analizar y distribuir dicha información.

De ahí la necesidad de contar con una herramienta que apoyara a los empleados en el desempeño de sus labores de manera eficiente.

La Dirección contaba ya con un sistema que permitía almacenar la información, consultarla y elaborar reportes a partir de ella. Dicho sistema, conocido como el Sistema de Información Hacendaria, tenía la desventaja de ser excesivamente lento en cuanto a la emisión de reportes de información, ya que éstos eran muy complejos.

Debido a su naturaleza, los reportes requerían el desplegado de los totales de grupo al inicio de dicho grupo y no al final, como lo hacen los reporteadores convencionales.

Para poder elaborar estos reportes, el sistema contaba con un reporteador, que haciendo uso de listas ligadas, creaba la estructura del reporte en la memoria de la computadora, llevaba a cabo las operaciones para el cálculo de los totales a partir de un evaluador de expresiones y posteriormente hacía un vaciado de la información en un archivo ASCII para poder enviarlo a la impresora.

Dado que esta información es consultada por ejecutivos de alto nivel, ya que les es útil en la toma de decisiones, resultaba indispensable solventar los problemas provocados

por el sistema y permitir el acceso a la información oportunamente y de manera sencilla.

Así mismo, dada la importancia de la información, se requería limitar el acceso a ésta a aquellas personas autorizadas.

3.2 DISEÑO DEL SISTEMA

Primeramente se requería organizar y almacenar la información recopilada, de forma tal que su consulta y manejo posterior se pudiera llevar a cabo con facilidad.

Se pensó en organizar la información en SERIES DE TIEMPO.

3.2.1 SERIES DE TIEMPO

Para éste sistema, una serie de tiempo es una variable económica o financiera, compuesta por un conjunto de observaciones que contienen valores para diferentes periodos de tiempo (mensuales, anuales, trimestrales etc.), la cual se identifica por una clave especifica que se encuentra clasificada por cinco grandes rubros pertenecientes a las finanzas públicas.

Estas series se clasifican de acuerdo a su naturaleza en tres tipos:

- SERIES CONCEPTUALES
- SERIES BÁSICAS
- SERIES GENERADAS

SERIES CONCEPTUALES

Las series conceptuales se utilizan para organizar la información en la base de datos. La clave se forma de un máximo de cinco caracteres alfanuméricos mediante los cuales se puede identificar una serie dependiendo de su nivel de desagregación. Para una mejor comprensión de como se utilizan estas series se presenta el siguiente ejemplo:

A.	Todas las series que principian con A corresponden al sector paraestatal.
AA.	Agrupa a todas las series de las entidades controladas por el gobierno.
AAA.	Esta serie engloba a todos los flujos de efectivo.
AAA01	Identifica a la entidad Petróleos Mexicanos.

SERIES BÁSICAS

Las series básicas reciben este nombre porque son aquellas que alimentan los usuarios con valores de acuerdo a su periodicidad.

La longitud de la clave es de siete caracteres alfanuméricos, los cuales identifican a la serie de acuerdo a la organización que se tenga en el área a la que corresponda.

SERIES GENERADAS

Estas series tienen las mismas características que las básicas con la diferencia de que su valor se calcula a partir de las operaciones efectuadas entre dos o más series básicas o la combinación de básicas y generadas, los operadores que se pueden utilizar en una fórmula son:

+ suma
- resta
* multiplicación
/ división

Además en una fórmula se pueden incluir valores constantes.

3.2.1.1 CLASIFICACIÓN DE LAS SERIES DE TIEMPO

Las Series de Tiempo que registra el sistema se clasifican bajo cinco grandes rubros:

A SECTOR PARAESTATAL
B GOBIERNO FEDERAL
D FUENTES DE FINANCIAMIENTO
E BANCA DE DESARROLLO
F SECTOR PUBLICO CONSOLIDADO

Cada uno de estos rubros presenta varias subclasificaciones, cuyo numero depende del nivel de desagregación de las estadísticas que agrupan.

A continuación se muestra hasta el segundo nivel de desagregación de las subclasificaciones de información.

Clasificación del Sector Paraestatal:

AA	ENTIDADES DENTRO DEL CONTROL PRESUPUESTAL
AB	ENTIDADES FUERA DEL CONTROL PRESUPUESTAL
AAA	FLUJO DE EFECTIVO
AAB	ESTADOS FINANCIEROS

Clasificacion del Sector Gobierno federal:

BA	INGRESOS PRESUPUESTALES DEL GOBIERNO FEDERAL
BB	INGRESO DEL GOBIERNO FEDERAL
BC	DEFICIT PRESUPUESTAL
BD	VARIACIONES DE CUENTAS AJENAS
BE	DEFICIT MONETARIO TOTAL
BF	FINANCIAMIENTO NETO
BG	INGRESOS ESTATALES
BH	COMERCIO EXTERIOR
BI	GASTO TOTAL POR CONCEPTO DE ADEFAS
BAA	SECTOR PETROLERO
BAB	SECTOR NO PETROLERO TRIBUTARIO
BAC	SECTOR NO PETROLERO NO TRIBUTARIO
BAD	INGRESOS COMPENSADOS RESTO SEC. PUBLICO

Casificación del Sector Fuentes de Financiamiento:

DA	FUENTES
DB	USOS
DC	FUENTES
DD	BASE MONETARIA POR USOS
DE	DEFICIT FINANCIERO
DF	SISTEMA BANCARIO
DG	CIRCULACION TOTAL
DH	SALDOS M1
DAA	RECURSOS INTERNOS
DAB	PESOS
DAC	EXT. NETO DOLARES

Clasificación del Sector Banca de Desarrollo:

EA	BANCOS NACIONALES DE DESARROLLO Y FONDOS Y FIDEICOMISOS.
EAA	PROGRAMA FINANCIERO ORIGEN Y APLICACION DE RECURSOS

Clasificación del Sector Público Consolidado:

FA	DEFICIT ECONOMICO
FB	DEFICIT PRESUPUESTAL
FC	DEFICIT EXTRAPRESUPUESTAL
FAA	GASTO PUBLICO TOTAL
FAB	INGRESOS PUBLICOS TOTALES
FAC	AHORRO CORRIENTE DEL SECTOR PUBLICO
FAD	SUPERAVIT ECONOMICO PRIMARIO
FAE	DIFERENCIA CON FUENTES DEL FINANCIAMIENTO
FAF	INTERMEDIACION FINANCIERA

La clave de cada una de las series está formada por letras y números, mediante los cuales pueden ser identificadas, e incluye cinco niveles de desagregación.

El primer nivel comprende sólo una letra e indica a que sector pertenence la serie, el segundo y tercero, también compuestos por letras, nos indican una subclasificación que tiene el sector, el cuarto, define la entidad a la que pertenece esa serie y el quinto, indica una subclasificación de la entidad.

A	A	A	01	01
1	2	3	4	5

1 La serie pertenece al Sector paraestatal
2 La serie corresponde a una de las Entidades dentro del control presupuestal
3 La serie correspone a los Flujos de Efectivo
4 La serie corresponde a la entidad Petroleos Mexicanos
5 La serie presenta los datos correspondientes a ingresos.

CONSIDERACIONES PARA EL DISEÑO DEL SISTEMA

Para poder crear un ambiente amigable se penso en la utilización de ventanas que permitieran la consulta de series en una forma organizada y sencilla, desglosando la clasificación elaborada a medida que se seleccionara una opción de la ventana mostrada.

Considerando las facilidades que nos ofrecen las herramientas desarrolladas para computadoras personales, como Lotus 1-2-3 y ALLWAYS y el auge de las redes de computadoras, se decidió desarrollar el sistema en un ambiente multiusuario para trabajar en una red de computadoras.

Tomando en cuenta las ventajas que proporciona una hoja cálculo de Lotus 1-2-3 para el análisis de datos organizados, se pensó en solventar el problema de los reportes emitidos por el Sistema de Información Hacendaria, haciendo uso de hojas electrónicas a las que se les dió el nombre de CUADROS ESTADÍSTICOS.

3.2.2 CUADROS ESTADÍSTICOS

Los cuadros estadísticos son reportes matriciales, elaborados a partir de las series de tiempo en arreglos de renglones y columnas definidos por los usuarios en hojas de cálculo, en donde se muestran Estados Financieros a partir de los valores de las series al ser generados por el sistema para un año y periodo determinados.

En ellos se presenta, en forma más clara, el uso de las series.

En el diseño de cuadros estadísticos se agrega un caracter más a la clave de las series, para indicar el tipo de información que se requiere, asignándose un '0' (cero) si los valores de la serie corresponden a la información definitiva o de flujo nominal, 'A' si la información es de presupuesto original y una 'M' si los valores son de presupuesto modificado.

El reporteador en tal caso debería ser capaz de leer el cuadro patrón, considerando el formato de una hoja de cálculo tal como lo utiliza Lotus 1-2-3, identificar aquellas celdas en las que existiera la clave de una serie y sustituirla por el valor correspondiente a ella para el año y periodo indicados.

Dado que las series serían almacenadas en una celda de tipo texto y como el cálculo de totales se dejaría al evaluador de expresiones del mismo Lotus, al sustituir la serie por su valor el tipo de celda también debería modificarse a numérico sin alterar la estructura del archivo, para que pudiera ser leída por Lotus sin ningún problema.

3.2.2.1 CLASIFICACIÓN DE CUADROS ESTADÍSTICOS

Los cuadros están clasificados también por rubros:

SP	SECTOR PARAESTATAL
GF	GOBIERNO FEDERAL
FF	FUENTES DE FINANCIAMIENTO
EA	BANCA DE DESARROLLO
SF	SECTOR PÚBLICO CONSOLIDADO

Por cada rubro existe un conjunto de cuadros, que reportan la información necesaria para cada uno de ellos.

3.2.3 DISEÑO DE SALIDAS

3.2.3.1 DISEÑO DE CUADROS ESTADÍSTICOS

SINTAXIS PARA LA ELABORACIÓN DE CUADROS ESTADÍSTICOS

Los cuadros estadísticos son una de las partes modulares que componen al Sistema. Para su diseño, se creó una sintaxis que permite elaborar cuadros, compuestos en su mayoría de valores y datos generales de series.

Debido a que muchos de éstos cuadros presentan formatos similares entre si, la sintaxis permite elaborar cuadros base o patrón, constituidos por variables de sustitución que son en si las que forman la sintaxis.

Esta sintaxis es muy sencilla y fácil de comprender.

El cuadro base se elabora en una hoja de cálculo y está formado por fórmulas expresadas de acuerdo a las variables (todos los cuadros son hojas de cálculo), posteriormente la hoja es procesada por el sistema, obteniéndose un cuadro estadístico con valores.

Haciendo un uso adecuado de la sintaxis, se podrán obtener cuadros base, que se utilicen para un mayor número de cuadros estadísticos.

VARIABLES DE SUSTITUCIÓN

Las variables de sustitución son parámetros que se sustituyen en la hoja de cálculo (cuadro estadístico), por un valor real para el cuadro generado, estas variables nos sirven para emitir diferentes cuadros con un sólo patrón de cuadro.

Las variables de sustitución siempre deberan ir encerradas entre corchetes, ya que de esto depende que sean consideradas como variables y no como texto. Estas se encuentran clasificadas en cinco tipos que son:

1) Variables Predefinidas
2) Prompts
3) Datos de Series
4) Valores de Series
5) Comodines

1) VARIABLES PREDEFINIDAS

Las variables predefinidas son aquellas que inician con un ampersand (&), precedido por una expresión que nos indica un tipo de fecha, un año de emisión, etc.

Existen diferentes tipos de expresiones (ver el anexo A).

La sintaxis para expresar una variable predefinida es la siguiente:

[&expresión]

Ejemplo:

[&FECHA1] [&emisión1] [&HORA]

Una de las características que presentan estas variables, es que dependiendo del formato en el que sean escritas serán sustituidas; es decir, si se escribe la variable [&DIA1] ésta sera sustituida por JUEVES y si se escribe [&Dia1] sera sustituida por Jueves; como se puede observar se toman en cuenta las mayúsculas y minúsculas.

2) PROMPTS

El prompts es una variable que inicia con un subguión (_), precedido por un enunciado, el cual, en el momento en que el cuadro es procesado, se nos presenta en pantalla y detiene el proceso hasta que se proporciona otro enunciado para sustituir al primero.

La sintaxis para expresar un prompt es la siguiente:

[_Enunciado]

Ejemplo:

[_Cual es el nombre de la persona que captura la información?]

El mensaje que se observaría es :

Cual es el nombre de la persona que captura la información?_

3) DATO DE SERIE

Esta variable inicia con una admiración cerrada (!) y nos indica que se va a realizar la sustitución, por alguno de los datos generales que tiene la serie.

La sintaxis para expresar esta variable es:

[!clave de la serie.campo]

Ejemplo:

[!AAA0101.tipo]

se sustituye por: GENERADA

Lo cual indica que la serie AAA0101 es del tipo Generada.

Nota: Si no se especifica el dato que se desea y solo se pone la clave de la serie, ésta será sustituida por su descripción.

[!clave de la serie]

4) VALOR DE LA SERIE

Esta variable se sustituye por el valor que tiene la serie, de acuerdo a los parámetros que se hayan indicado desde el sistema o en la sintaxis de la expresión.

La sintaxis para expresar esta variable es:

[clave de la serie.modo de información.año;periodo]

Clave de Serie:
Es un formato predefinido con el cual se identifica una serie de tiempo.

Modo de información:
Es una letra A ,D o @. La A indica que el valor que se desea de la serie es Acumulado, la D que el valor es desacumulado y el @ indica que se va ha tomar el Modo de Información que se dio como parámetro de entrada al sistema.

Año:
Es un número que nos indica el año para el cual se desea el valor de la serie, este dato puede ser relativo al proporcionado como parámetro de entrada al sistema (con aumentos o decrementos al mismo).

Periodo:

El periodo es un número que nos indica el mes, semana, trimestre, etc. (periodo), para el cual se desea el valor de la serie. Este dato puede ser también relativo al proporcionado como parámetro de entrada al sistema (con aumentos o decrementos al mismo).

Ejemplos:

[BBE05Q40.A.90:05]

Se requiere el valor acumulado de la serie BBE05Q40 al periodo 5 de 1990.

[BBE05Q40.@.-1:+5]

Se requiere el valor acumulado o desacumulado, de acuerdo a lo que se de como parámetro de entrada en la ejecución del sistema, de la serie BBE05Q40 para el año indicado en la lista de parámetros menos uno, periodo de entrada más cinco.

[BBE05Q40]

Se requiere el valor de la serie BBE05Q40 de acuerdo a los datos que se den en la lista de parámetros de entrada en la ejecución del Sistema.

5) COMODÍN

El comodín es la única variable de sustitución que no se escribe encerrada entre corchetes. Su existencia en el cuadro implica la sustitución de estos por los parámetros de entrada proporcionados por el usuario cuando el sistema se los solicita para tal efecto.

Dicha sustitución es la primera que se lleva a cabo al procesar el cuadro, ya que puede existir un comodín dentro de otra variable de sustitución.

Se puede emplear un máximo de 10 comodines (&0 a &9).

La sintaxis de esta variable es:

&0, &1, &2, &3, &4, &5, &6, &7, &8, &9

Ejemplo:

Podemos tener en el cuadro patrón [&0AA&1&20.A.90:12]. Si en los parámetros de entrada se indica sustituir &0 por A, &1 por 01 y &2 por 02.

Después de sustituir los comodines tendríamos [AAA01020.A.90:12]

FÓRMULAS

Las fórmulas son expresiones algebráicas que se componen de valores de series, celdas y constantes como operandos y como operadores suma(+), resta(-), multiplicación(*), división(/), exponencial(^) y paréntesis (). Fuera de estos operadores no se puede usar ninguna otra función que venga integrada en Lotus 1-2-3.

La sintaxis para la expresión de fórmulas es:

[valor de la serie] operador [valor de la serie]
[valor de la serie] * cte.+ celda

Las celdas se expresan de las misma forma que en Lotus 1-2-3.

Ejemplo:
([AAA01010.A.90:12]+[AAA01020.@])*35/D25

3.2.3.2 ELABORACIÓN DE UN CUADRO ESTADÍSTICO

Cada vez que se vaya a diseñar un nuevo cuadro deben ser considerados los siguientes puntos:

1) Se deben clasificar los cuadros de acuerdo al formato que tengan para poder obtener uno que sea base o patrón para los demás.

2) No solo se debe tomar en cuenta el formato de texto del cuadro para elaborar un cuadro base, sino también las claves de serie que lo conforman, así como la clasificación por rubros que tienen los cuadros.

3) Ocupar todas las variables de sustitución que sea posible, con el fin de obtener un solo cuadro base. Esto en el caso de no haberse podido realizar la primera consideración debido a las características del cuadro.

Para poder elaborar un cuadro, es esencial conocer las instrucciones básicas para operar Lotus 1-2-3, ya que los cuadros estadísticos son hojas de cálculo con un formato

específico, el cual es interpretado y procesado por el Sistema.

El ejemplo que a continuación se presenta es solo una guía para poder elaborar cuadros. Con éste no se pretende enseñar al lector a trabajar con Lotus 1-2-3 y se considera que las instrucciones ya se conocen.

DISEÑO

Como ya se a mencionado anteriormente, la herramienta en donde se diseña el cuadro es Lotus 1-2-3, por tanto, el primer paso es accesar el paquete.

Se explica como elaborar un cuadro, como el mostrado en la fig. 3.1

Para obtener éste cuadro, se realizó un análisis que permitió identificar las partes que de él eran variables y aquellas que permanecían constantes al considerar diferentes entidades o empresas.

Las partes variables se encuentran encerradas dentro de rectángulos y se explican a continuación:

1. Es muy común, colocar en la esquina superior izquierda del cuadro, la fecha y hora en que éste se ha emitido (generado). Para expresarlo existen diversos formatos (Consultar Anexo A), de los cuales, para el ejemplo se eligen los más usuales.

2. Al centro y debajo del primer título, generalmente se incluye la dependencia o entidad a la que pertenece la información de ese cuadro. Esta se expresa de acuerdo a la sintáxis de la variable de sustitución Dato de Serie (consultar eÍ anexo A).

3. En este punto se expresa una variable predefinida, que indica el modo en que se ha de presentar la información (acumulada o desacumulada). Para el ejemplo se elige el plural femenino, sin embargo, existen otras opciones (consultar el anexo A).

4. Al centro y en la parte final del primer título, en algunos cuadros como en este caso, es necesario poner el año para el cual se va a emitir, pues no necesariamente tiene que ser el año lectivo, para ello existen también algunas variables predefinidas. Para conocer los diferentes tipos consultar el anexo A

5. Es costumbre, expresar en la esquina superior derecha la clave del cuadro que se está presentando,

[&FECHA1]
[AHORA1]

FLUJO DE EFECTIVO DE LOS ORGANISMOS Y EMPRESAS DURANTE [&SEMI1] 4

[&PAG#1]

1 [&AÑO1] 2

3 CIFRAS AMORDEFIEN MILLONES DE PESOS

CONCEPTOS	ENERO	FEBRERO	MARZO	ABRIL	MAYO	JUNIO	JULIO	AGOSTO	SEPTIEMBRE	OCTUBRE	NOVIEMBRE	DICIEMBRE
INGRESOS CORRIENTES	[A&AA101	[A&AA101	[A&AA101	[A&AA101	[A&AA101	[A&AA101	[A&AA101	[A&AA101	[A&AA101	[A&AA101	[A&AA101	[A&AA101
VTA. BNES. Y SERVS	[A&AA102	[A&AA102	[A&AA102	[A&AA102	[A&AA102	[A&AA102	[A&AA102	[A&AA102	[A&AA102	[A&AA102	[A&AA102	[A&AA102
INTERNAS	[A&AA107	[A&AA107	[A&AA107	[A&AA107	[A&AA107	[A&AA107	[A&AA107	[A&AA107	[A&AA107	[A&AA107	[A&AA107	[A&AA107
EXTERNAS	[A&AA105	[A&AA105	[A&AA105	[A&AA105	[A&AA105	[A&AA105	[A&AA105	[A&AA105	[A&AA105	[A&AA105	[A&AA105	[A&AA105
CUOTAS SEG. SOC.	[A&AA109	[A&AA109	[A&AA109	[A&AA109	[A&AA109	[A&AA109	[A&AA109	[A&AA109	[A&AA109	[A&AA109	[A&AA109	[A&AA109
GASTOS DE OPER	[A&AA111	[A&AA111	[A&AA111	[A&AA111	[A&AA111	[A&AA111	[A&AA111	[A&AA111	[A&AA111	[A&AA111	[A&AA111	[A&AA111
SERVS PERSONA	[A&AA112	[A&AA112	[A&AA112	[A&AA112	[A&AA112	[A&AA112	[A&AA112	[A&AA112	[A&AA112	[A&AA112	[A&AA112	[A&AA112
ADQ. MAT. Y SUMS	[A&AA113	[A&AA113	[A&AA113	[A&AA113	[A&AA113	[A&AA113	[A&AA113	[A&AA113	[A&AA113	[A&AA113	[A&AA113	[A&AA113
INTERESES	[A&AA114	[A&AA114	[A&AA114	[A&AA114	[A&AA114	[A&AA114	[A&AA114	[A&AA114	[A&AA114	[A&AA114	[A&AA114	[A&AA114
SERVS GRALES	[A&AA117	[A&AA117	[A&AA117	[A&AA117	[A&AA117	[A&AA117	[A&AA117	[A&AA117	[A&AA117	[A&AA117	[A&AA117	[A&AA117
DEFICIT/SUP. DE OP.	[A&AA118	[A&AA118	[A&AA118	[A&AA118	[A&AA118	[A&AA118	[A&AA118	[A&AA118	[A&AA118	[A&AA118	[A&AA118	[A&AA118
TRANS CORRIENTE	[A&AA119	[A&AA119	[A&AA119	[A&AA119	[A&AA119	[A&AA119	[A&AA119	[A&AA119	[A&AA119	[A&AA119	[A&AA119	[A&AA119
IMPUESTOS PAGADOS	[A&AA122	[A&AA122	[A&AA122	[A&AA122	[A&AA122	[A&AA122	[A&AA122	[A&AA122	[A&AA122	[A&AA122	[A&AA122	[A&AA122
DEFICIT/SUP CTA CORR.	[A&AA123	[A&AA123	[A&AA123	[A&AA123	[A&AA123	[A&AA123	[A&AA123	[A&AA123	[A&AA123	[A&AA123	[A&AA123	[A&AA123
INGRESOS DE CAPITAL	[A&AA124	[A&AA124	[A&AA124	[A&AA124	[A&AA124	[A&AA124	[A&AA124	[A&AA124	[A&AA124	[A&AA124	[A&AA124	[A&AA124
TRANS. DE CAPITAL	[A&AA125	[A&AA125	[A&AA125	[A&AA125	[A&AA125	[A&AA125	[A&AA125	[A&AA125	[A&AA125	[A&AA125	[A&AA125	[A&AA125
GASTOS DE CAPITAL	[A&AA129	[A&AA129	[A&AA129	[A&AA129	[A&AA129	[A&AA129	[A&AA129	[A&AA129	[A&AA129	[A&AA129	[A&AA129	[A&AA129
INVERSION FISICA	[A&AA130	[A&AA130	[A&AA130	[A&AA130	[A&AA130	[A&AA130	[A&AA130	[A&AA130	[A&AA130	[A&AA130	[A&AA130	[A&AA130
INVERSION FINAN.	[A&AA131	[A&AA131	[A&AA131	[A&AA131	[A&AA131	[A&AA131	[A&AA131	[A&AA131	[A&AA131	[A&AA131	[A&AA131	[A&AA131
VAR CUENTAS AJENAS	[A&AA132	[A&AA132	[A&AA132	[A&AA132	[A&AA132	[A&AA132	[A&AA132	[A&AA132	[A&AA132	[A&AA132	[A&AA132	[A&AA132
DEF/SUP TOTAL	[A&AA139	[A&AA139	[A&AA139	[A&AA139	[A&AA139	[A&AA139	[A&AA139	[A&AA139	[A&AA139	[A&AA139	[A&AA139	[A&AA139
FINANCIAMIENTO NETO	[A&AA140	[A&AA140	[A&AA140	[A&AA140	[A&AA140	[A&AA140	[A&AA140	[A&AA140	[A&AA140	[A&AA140	[A&AA140	[A&AA140
ENDEUDAMIENTO NETO	[A&AA141	[A&AA141	[A&AA141	[A&AA141	[A&AA141	[A&AA141	[A&AA141	[A&AA141	[A&AA141	[A&AA141	[A&AA141	[A&AA141
INTERNO	[A&AA142	[A&AA142	[A&AA142	[A&AA142	[A&AA142	[A&AA142	[A&AA142	[A&AA142	[A&AA142	[A&AA142	[A&AA142	[A&AA142
COLOCACION	[A&AA143	[A&AA143	[A&AA143	[A&AA143	[A&AA143	[A&AA143	[A&AA143	[A&AA143	[A&AA143	[A&AA143	[A&AA143	[A&AA143
AMORTIZACION	[A&AA144	[A&AA144	[A&AA144	[A&AA144	[A&AA144	[A&AA144	[A&AA144	[A&AA144	[A&AA144	[A&AA144	[A&AA144	[A&AA144
EXTERNO	[A&AA145	[A&AA145	[A&AA145	[A&AA145	[A&AA145	[A&AA145	[A&AA145	[A&AA145	[A&AA145	[A&AA145	[A&AA145	[A&AA145
COLOCACION	[A&AA146	[A&AA146	[A&AA146	[A&AA146	[A&AA146	[A&AA146	[A&AA146	[A&AA146	[A&AA146	[A&AA146	[A&AA146	[A&AA146
AMORTIZACION	[A&AA147	[A&AA147	[A&AA147	[A&AA147	[A&AA147	[A&AA147	[A&AA147	[A&AA147	[A&AA147	[A&AA147	[A&AA147	[A&AA147
VAR. DE DISPONIB.	[A&AA148	[A&AA148	[A&AA148	[A&AA148	[A&AA148	[A&AA148	[A&AA148	[A&AA148	[A&AA148	[A&AA148	[A&AA148	[A&AA148
PRESUPUESTO												
INGRESO PROPIO	[A&AA101	[A&AA101	[A&AA101	[A&AA101	[A&AA101	[A&AA101	[A&AA101	[A&AA101	[A&AA101	[A&AA101	[A&AA101	[A&AA101
GASTO TOTAL	[A&AA111	[A&AA111	[A&AA111	[A&AA111	[A&AA111	[A&AA111	[A&AA111	[A&AA111	[A&AA111	[A&AA111	[A&AA111	[A&AA111
DEF/SUP PRESUP	[A&AA139	[A&AA139	[A&AA139	[A&AA139	[A&AA139	[A&AA139	[A&AA139	[A&AA139	[A&AA139	[A&AA139	[A&AA139	[A&AA139
FINANCIAMIENTO	[A&AA123	[A&AA123	[A&AA123	[A&AA123	[A&AA123	[A&AA123	[A&AA123	[A&AA123	[A&AA123	[A&AA123	[A&AA123	[A&AA123
AHORRO/DESAHORRO	[A&AA127	[A&AA127	[A&AA127	[A&AA127	[A&AA127	[A&AA127	[A&AA127	[A&AA127	[A&AA127	[A&AA127	[A&AA127	[A&AA127
INGRESO DE CAP.	[A&AA124	[A&AA124	[A&AA124	[A&AA124	[A&AA124	[A&AA124	[A&AA124	[A&AA124	[A&AA124	[A&AA124	[A&AA124	[A&AA124
TRANS. DE CAPITAL	[A&AA125	[A&AA125	[A&AA125	[A&AA125	[A&AA125	[A&AA125	[A&AA125	[A&AA125	[A&AA125	[A&AA125	[A&AA125	[A&AA125
FINANCIAMIENTO NETO	[A&AA140	[A&AA140	[A&AA140	[A&AA140	[A&AA140	[A&AA140	[A&AA140	[A&AA140	[A&AA140	[A&AA140	[A&AA140	[A&AA140
TOTAL DE TRANS.	[A&AA154	[A&AA154	[A&AA154	[A&AA154	[A&AA154	[A&AA154	[A&AA154	[A&AA154	[A&AA154	[A&AA154	[A&AA154	[A&AA154
AJUSTES DE EJERCICIO	[A&AA133	[A&AA133	[A&AA133	[A&AA133	[A&AA133	[A&AA133	[A&AA133	[A&AA133	[A&AA133	[A&AA133	[A&AA133	[A&AA133
AJUSTES DE GASTO	[A&AA136	[A&AA136	[A&AA136	[A&AA136	[A&AA136	[A&AA136	[A&AA136	[A&AA136	[A&AA136	[A&AA136	[A&AA136	[A&AA136

* Cifras preliminares

la cual, es elaborada por medio de comodines o caracteres de sustitución, que permiten indicarlo así para diferentes emisiones.

6. La parte sombreada que se muestra en este punto es la más importante en el cuadro, ya que se trata de los valores de las series.

Como se vió en puntos anteriores, para sustituir el valor de una serie, es necesario expresarlo de acuerdo a una sintaxis formada de variables de Valor de Serie y Fórmulas; éstas últimas compuestas de las primeras.

Esta área presenta en la columna del mes de enero un formato para valor de serie diseñado con comodines. Los formatos de valor de serie que aquí se muestran son los siguientes:

Renglón	Formato
1	[A&0A&101&2 @ 0]
2	[A&0A&102&2 @ 0]
3	[A&0A&107&2 @ 0]
4	[A&0A&105&2 @ 0] + [A&0A&108&2 @ 0]
5	[A&0A&109&2 @ 0]
6	[A&0A&111&2 @ 0]
7	[A&0A&112&2 @ 0]
8	[A&0A&113&2 @ 0]
9	[A&0A&114&2 @ 0]
10	[A&0A&117&2 @ 0]
11	[A&0A&118&2 @ 0]
12	[A&0A&119&2 @ 0]
13	[A&0A&122&2 @ 0]
14	[A&0A&123&2 @ 0]
15	[A&0A&124&2 @ 0]
16	[A&0A&125&2 @ 0]
17	[A&0A&129&2 @ 0]
18	[A&0A&130&2 @ 0]
19	[A&0A&131&2 @ 0]
20	[A&0A&132&2 @ 0]
21	[A&0A&139&2 @ 0]
22	[A&0A&140&2 @ 0]
23	[A&0A&141&2 @ 0]
24	[A&0A&142&2 @ 0]
25	[A&0A&143&2 @ 0]
26	[A&0A&144&2 @ 0]
27	[A&0A&145&2 @ 0]
28	[A&0A&146&2 @ 0]
29	[A&0A&147&2 @ 0]
30	[A&0A&148&2 @ 0]
31	[A&0A&101&2 @ 0]+[A&0A&124&2 @ 0]+[A&0A&133&2 @ 0]
32	[A&0A&111&2 @ 0]+[A&0A&122&2 @ 0]+[A&0A&129&2 @ 0]+ [A&0A&136&2 @ 0]

```
33      [A&0A&139&2.@.0]+[A&0A&154&2.@.0]
34      [A&0A&123&2.@.0]+[A&0A&132&2.@.0]+[A&0A&124&2.@.0]+
        [A&0A&125&2.@.0]+[A&0A&140&2.@.0]
35      [A&0A&123&2.@.0]+[A&0A&132&2.@.0]
36      [A&0A&124&2.@.0]
37      [A&0A&125&2.@.0]
38      [A&0A&140&2.@.0]
39      [A&0A&154&2.@.0]
40      [A&0A&133&2.@.0]
41      [A&0A&136&2.@.0]
```

Como se puede observar, todas estas series se piden con un incremento de cero, relativo al periodo base; esto quiere decir, que se sustituiran por el valor de la serie en el período para el cual sea emitido el cuadro.

Las columnas posteriores tienen la misma sintaxis pero un incremento en el período base diferente, ya que por ser este un cuadro anual, el valor en cada una de las columnas indica el valor de la serie para cada uno de los períodos en el año.

Estos formatos de Valor de serie están formados por comodines; el &0 será sustituido por una A, el &1 por 01 y el &2 por un 0.

De este modo se le da formato al cuadro y una vez que se tienen todas las partes capturadas en la hoja de Lotus 1-2-3, el cuadro está listo para ser procesado por el sistema.

3.2.4 DISEÑO Y PROCESAMIENTO DE ARCHIVOS

3.2.4.1 DISEÑO DE LA BASE DE DATOS

Para almacenar las claves de las series de tiempo y sus atributos, se eligió un archivo de tipo tabla, con la estructura que se muestra a continuación:

TABLA FPDGRALS

CAMPO	DESCRIPCION
DGCLV	Clave de la serie
DGDES	Descripción de la serie
DGUNI	Unidad monetaria en la que se expresan los valores de la serie
DGPER	Periodicidad de la serie
DGTIP	Tipo de Serie (Básica o Generada)
DGANI	Año a partir del cual se reportó información para la serie
DGPEI	Periodo a partir del cual se reportó información para la serie
DGANF	Ultimo año para el que existen valores de la serie
DGPEF	Ultimo periodo para el cual existen valores de la serie
DGANS	Año hasta el cual es permitida la consulta de valores de una serie a usuarios no propietarios
DGPS	Periodo limite de consulta de valores de una serie a usuarios no propietarios
DGFUE	Fuente de información de la serie
DGFUA	Fecha de la última actualización a la serie
DGAREA	Clave del área a la cual pertenece la serie
REGREINI	Apuntador al primer valor de la serie en el archivo de flujo nominal
REGORINI	Apuntador al primer valor de la serie en el archivo de presupuesto original
REGMODINI	Apuntador al primer valor de la serie en el archivo de presupuesto modificado

Las series conceptuales se utilizan para organizar la información en la base de datos y permitir el acceso a las series básicas correspondientes a las primeras. Para almacenar dichas series y sus atributos se eligió de igual forma un archivo de tipo tabla, con la estructura mostrada a continuación:

TABLA FPDGRALS

CAMPO	DESCRIPCION
DGCLV	Clave de la serie conceptual
DGDES	Descripción de la serie conceptual

El catálogo de usuarios nos sirve para clasificar a los usuarios que tienen acceso al sistema y la forma en que pueden hacerlo.

Existen los usuarios propietario a los que se les da acceso a toda la información de las series correspondiente al area a la que pertenecen y los usuarios de consulta, a quienes sólo se les permite consultar la información de la serie hasta el año y período de seguridad asignados por los propietarios.

Para almacenar ésta información se eligió un archivo de tipo tabla.

TABLA FPUSER

CAMPO	DESCRIPCION
USCLV	Clave del usuario
USNOM	Nombre del usuario
USTEL1	Numero telefónico del usuario. Telefono 1
USEXT1	Extensión telefónica 1
USTEL2	Número telefónico del usuario. Teléfono 2
USEXT2	Extensión telefónica 2
USUBIC	Domicilio del usuario
USAREA	Area a la que pertenece el usuario

Para poder presentar a los usuarios del sistema un menu de cuadros disponibles, se requería de un catálogo a partir del cual el Sistema pudiera leerlos.

Para almacenar esta información se eligió también un archivo de tipo tabla.

TABLA DGCUADRO

CAMPO	DESCRIPCION
DGCLV	Clave asignada al cuadro
DGCDES	Descripción del cuadro
DGCPER	Periodicidad del cuadro
DGCRES	Clave de la persona responsable del cuadro
DGCAREA	Area a la que pertenece el cuadro

Un aspecto muy importante que debía ser considerado, era el volumen de información, ya que el sistema desarrollado para el equipo VAX, a partir del cual sería alimentado éste Sistema, contenía información histórica de todas las variables económicas, desde 1975 a la fecha.

Inicialmente se penso en almacenar la información en un archivo de tipo tabla con la siguiente estructura:

CAMPO	DESCRIPCION
CLAVE	Clave de la serie a la cual corresponde el valor
AÑO	Año en el que fue reportado el valor
PERIODO	Periodo en el que fué reportado el valor
VALOR	Valor reportado para la serie

A 19 años de información de series mensuales (en su mayoría las series tienen esta periodicidad), corresponden 228 registros, porque tenemos 12 periodos al año.

Utilizando ésta estructura, tendríamos 228 registros con la misma clave de serie por cada una de las series almacenadas en el Sistema, aunque la llave de busqueda sería una clave compuesta, se tendria una gran cantidad de información repetida.

Considerando la longicud de la clave de 7 caracteres, el año de 2, el periodo de 3 y el valor de 16, para un archivo de dBase, tendríamos un registro de 29 bytes de longitud. Lo que significa 29 * 288 = 8352 bytes de espacio para cada una de las series.

Tomando en cuenta que el Sistema se compone por aproximadamente 35105 series, la cantidad de espacio en disco necesaria para almacenar la información correspondiente a un tipo de información seria de 293,196,960 bytes, y como el sistema almacena tres tipos de información para cada serie (de Flujo nominal, de presupuesto original y de presupuesto modificado), esta cantidad se incrementaría.

Había que elegir otra forma de almacenamiento.

Después de un concienzudo análisis, se decidió almacenar la información en archivos random, compuestos por elementos de tipo double de 8 bytes de longitud.

El primer elemento de cada archivo sería un número compuesto que indicaría el rango de años de la información almacenada (19751993).

A cada serie se le reservaría el espacio correspondiente al rango de años indicados en el primer elemento y a la periodicidad de la serie; de forma tal que todas las series tendrian un espacio asignado para almacenar sus valores.

Para establecer la liga entre la serie y sus datos con el conjunto de valores correspondiente a ella se incluyó en el registro de los datos de la serie un apuntador al primer elemento del conjunto de valores.

El análisis realizado, revelo que para 19 años de información, considerando 35105 series mensuales, se necesitaban:

$$19 * 12 * 35105 * 8 = 64,031,520 + 8 = 64,031,528 \text{ bytes}$$

8 bytes por el primer elemento
8 bytes por cada elemento (numero de tipo doble)
35105 series
12 periodos al año para las series mensuales
19 años

64,031,528 bytes de espacio en disco para almacenar la información correspondiente a un tipo de información de la serie.

Tomando en cuenta los tres tipos de información que presentan las series, necesitaríamos:

64,031,528	bytes para el archivo de valores de flujo nominal
64,031,528	bytes para el archivo de valores de presupuesto original
30,330,720	bytes para el archivo de valores de presupuesto modificado, ya que de este tipo de información se tiene a partir de 1985.
158,393,776	bytes para almacenar todos los valores reportados para las series.

Mucho menos de lo que se requería en el diseño original.

3.3 RELACION DE ARCHIVOS

3.3.1. ARCHIVOS DE TABLAS

FPDGRALS	Tabla de los Datos Generales de las Series de Tiempo.
FPCVDES	Tabla que contiene clave y descripción de las Series Conceptuales
FPUSER	Catálogo de Usuarios del Sistema.
DGCUADRO	Catálogo de Cuadros estadísticos

3.3.2 ARCHIVOS DE VALORES

FPVALR	Archivo de valores de flujo nominal.
FPVALO	Archivo de valores de presupuesto original.
FPVALM	Archivo de valores de presupuesto modificado.

3.3.3 FORMATO DE REGISTROS DE ARCHIVOS

TABLAS

Tabla : FPDGRALS.
Organización : Indexada (DGCLV).

Campo	Nombre de Campo	Tipo	Longitud
1	DGCLV	Caracter	7
2	DGDES	Caracter	60
3	DGUNI	Caracter	25
4	DGPER	Caracter	1
5	DGTIP	Caracter	1
6	DGANI	Caracter	2
7	DGPEI	Caracter	3
8	DGANF	Caracter	2
9	DGPEF	Caracter	3
10	DGANS	Caracter	2
11	DGPS	Caracter	3
12	DGFUE	Caracter	20
13	DGFUA	Caracter	9
14	DGAREA	Caracter	2
15	REGREINI	Caracter	8
16	REGORINI	Caracter	8
17	REGMODINI	Caracter	7

Tabla : FPCVDES.
Organización : Indexada (DGCLV).

Campo	Nombre de Campo	Tipo	Longitud
1	DGCLV	Caracter	7
2	DGDES	Caracter	60

Tabla : FPUSER.
Organización : Indexada (USCLV).

Campo	Nombre de Campo	Tipo	Longitud
1	USCLV	Caracter	4
2	USNOM	Caracter	50
3	USTEL1	Caracter	7
4	USEXT1	Caracter	4
5	USTEL2	Caracter	7
6	USEXT2	Caracter	4
7	USUBIC	Caracter	60
8	USAREA	Caracter	3

Tabla : DGCUADRO.
Organización : Indexada (DGCLV).

Campo	Nombre de Campo	Tipo	Longitud
1	DGCLV	Caracter	8
2	DGCDES	Caracter	60
3	DGCPER	Caracter	1
4	DGCRES	Caracter	4
5	DGCAREA	Caracter	3

ARCHIVOS DE VALORES

Archivos : FPVALR, FPVALO, FPVALM.
Modo : Random.
Organización: Acceso directo.
Descripción :

Los elementos que forman los archivo, son de tipo doble (8 bytes). El primero de ellos, en cada archivo, indica el rango de años para los cuales se puede almacenar información de cada una de las series.

3.4 FLUJO DE INFORMACIÓN

1) La información existente en la base de datos del Sistema de Finanzas Públicas de la MicroVax 3400, es transportada en modo ASCII hacia archivos secuenciales: uno de datos generales y tres de valores (flujo nominal, presupuesto original y presupuesto modificado), todos con extensión TXT.

   La información generada para cada uno de estos archivos corresponde a un año de emisión.

2) Los archivos generados con información del Sistema, son transportados a la Red Pública de la Dirección General de Planeación Hacendaria de la SHCP.
   Este proceso consiste en la extracción de los archivos de la Vax3400 al disco duro de alguna máquina PC compatible, por medio del paquete PCSA. Posteriormente, conectando la PC como nodo de la Red, se transladan directamente del disco duro de la máquina al directorio correspondiente del Sistema de Finanzas Públicas Ejecutivo.

3) Dentro del directorio del Sistema (en la Red), se prosigue a la carga de información hacia la base de datos (.DBF) y archivos de valores (.DAT); dejando de esta forma, la información actualizada y lista para ser consultada desde el Sistema de Finanzas Públicas Ejecutivo.

La generación de los archivos de información en la Vax3400 corresponde a solo un año de emisión, por lo que la carga de éstos al Sistema de la Red será para el mismo periodo.

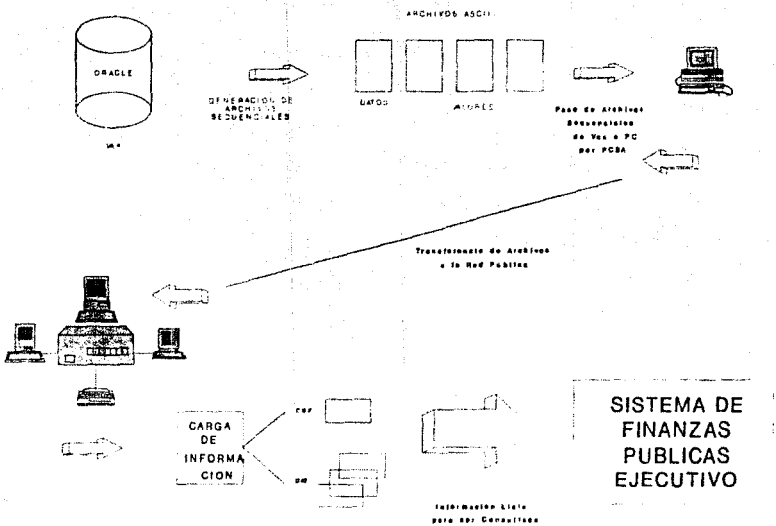


DIAGRAMA DE FLUJO DE INFORMACION

3.5 MÓDULOS QUE COMPONEN EL SISTEMA

El sistema se compone básicamente de dos módulos:

- Series de Tiempo.
- Cuadros Estadísticos.

El módulo de Series de Tiempo, muestra la información que se tiene disponible en el banco de datos y permite manipularla de diversas formas. Este módulo se compone de cinco submódulos que en conjunto llevan a cabo la tarea.

El módulo de Cuadros Estadísticos, presenta un menú de cuadros disponibles para consulta de información en forma de reportes y permite la generación de los mismos.

Las figuras mostrada a continuación pretenden dar una explicación más objetiva sobre el diseño del Sistema.

En la figura 1, podemos observar el diagrama general del Sistema, en el que se muestra el funcionamiento del mismo y la forma en que se presenta la información para su consulta.

Tanto en la Consulta de Series como en la Consulta y emisión de Cuadros, el primer paso que se lleva a cabo es la clasificación de estos por Sector.

Así, al elegir cualquiera de las dos opciones y seleccionar un sector, son presentadas las series o cuadros disponibles para dicho sector, según corresponda y posteriormente se muestra la información para su manipulación.

La fig. 2 muestra el diagrama de bloques del Sistema, en el se presentan como entradas, parámetros que son las opciones elegidas por el usuario y archivos que en el módulo de Cuadros Estadísticos representan los cuadros patrón, a partir de los cuales se genera la información solicitada.

En la fig. 3 se muestran cada uno de los módulos que componen al Sistema e indican los requerimientos de entrada y lo que entregan a la salida. El módulo de Series además de presentar la información correspondiente a las series seleccionadas, permite la exportación de dicha información a archivos de Lotus 1-2-3 y a archivos ASCII en forma de

reportes tabulares.

Las fig. 4 muestra el primer submódulo, del módulo de Series, el cual se encarga de organizar la información y presentar los menus para consulta de información.

En la fig. 5 observamos un módulo de búsqueda de palabra clave, la cual es proporcionada como parámetro de entrada y como salida se obtiene una clasificación de las Series que contiene dicha palabra dentro de su descripción.

También se muestra el módulo de desplegado de información, el cual se encarga de presentar en pantalla la correspondiente a la selección realizada en el módulo de menus.

La fig. 6 muestra los módulos de generación de reportes tabulares y de exportación de información a Lotus 1-2-3; ambos requieren como datos de entrada el conjunto de series seleccionadas en el módulo de menus, de las cuales entregarán información en archivos; el primero de tipo ASCII y el segundo en una hoja de cálculo.

La fig. 7 presenta el módulo de menus para consulta y generación de Cuadros Estadísticos, disponibles para cada uno de los Sectores que componen al Sistema. Este módulo pasa como parámetros de entrada al módulo de emisión de Cuadros la selección hecha por el usuario en esta parte y las indicaciones que para su emisión se hicieron.

En la fig. 8 observamos el módulo de Emisión de Cuadros Estadísticos. Este módulo recibe como entradas: el cuadro patrón con claves de series y variables de sustitución e indicaciones de como llevar a cabo dicha sustitución y entrega como resultado un reporte matricial en una hoja de cálculo con valores de las series incluidas en dicho reporte.

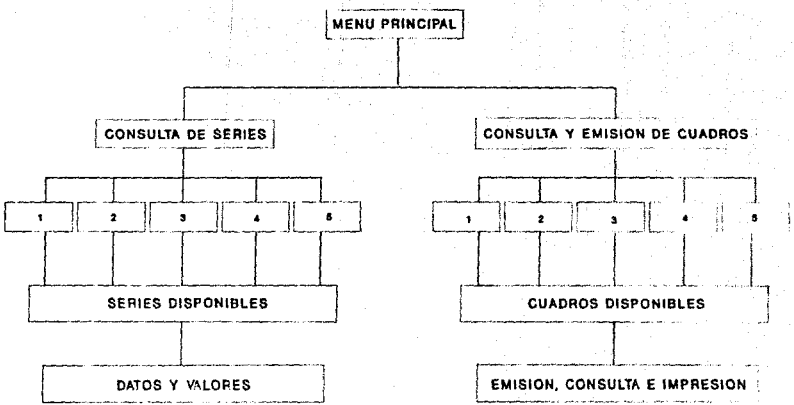
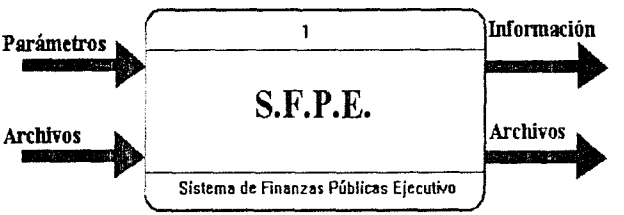
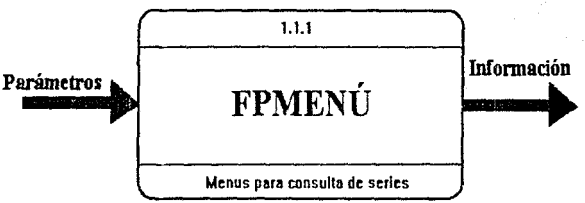


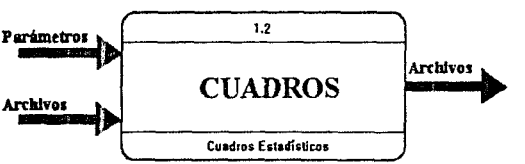
DIAGRAMA GENERAL

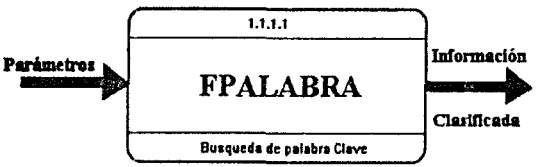
1 SECTOR PUBLICO CONSOLIDADO 2 GOBIERNO FEDERAL 3 BANCA DE DESARROLLO 4 SECTOR PARAESTATAL 5 FUENTES DE FINANCIAMIENTO

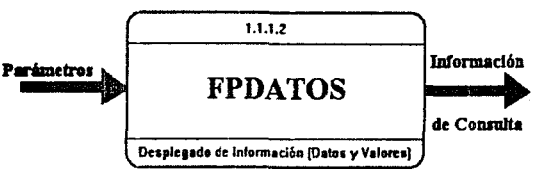

Parámetros
Archivos
1
S.F.P.E.
Sistema de Finanzas Públicas Ejecutivo
Información
Archivos

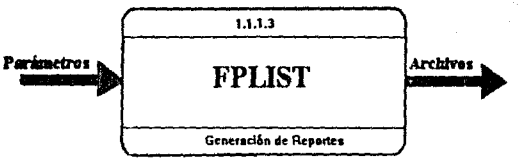

1.1.1
Parámetros
FPMENÚ
Información
Menus para consulta de series

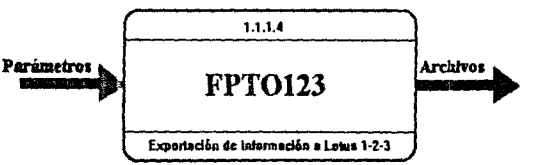

1.1
Parámetros
SERIES
Información
Archivos
Series de Tiempo

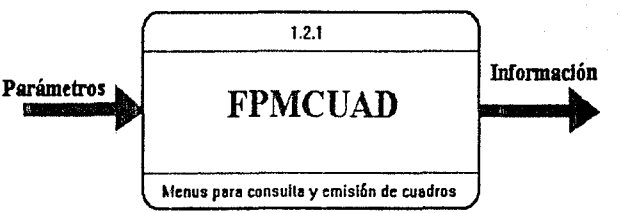

1.2
Parámetros
Archivos
CUADROS
Archivos
Cuadros Estadísticos

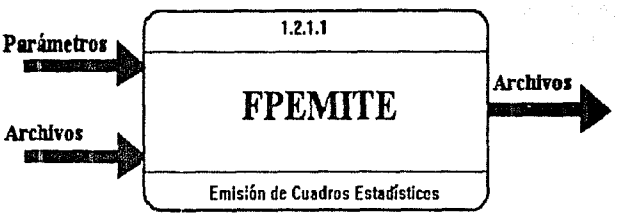

1.1.1.1
Parámetros
FPALABRA
Información
Clasificada
Busqueda de palabra Clave


1.1.1.2
Parámetros
FPDATOS
Información
de Consulta
Despliegado de Información (Datos y Valores)


1.1.1.3
Parámetros
FPLIST
Archivos
Generación de Reportes


1.1.1.4
Parámetros
FPTO123
Archivos
Exportación de información a Lotus 1-2-3


1.2.1
Parámetros
FPMCUAD
Información
Menus para consulta y emisión de cuadros


1.2.1.1
Parámetros
Archivos
FPEMITE
Archivos
Emisión de Cuadros Estadísticos

3.6 DESARROLLO DEL SOFTWARE

3.6.1 LENGUAJE DE PROGRAMACIÓN

En todo desarrollo de un sistema, la elección de un lenguaje de programación es un factor importante; ya que de esto depende la explotación adecuada del hardware elegido, así como la rapidez con que se pueda desarrollar el mismo.

Para el desarrollo del presente Sistema, el lenguaje C fue elegido por las características que a continuación se describen.

El lenguaje C fue desarrollado para optimizar la comunicación entre un sistema operativo de computadora y su lenguaje de máquina interna, facilitando la programación de tareas y haciéndolas más directas. C es un lenguaje de programación de proposito general, relativamente estructurado, de bajo nivel (es decir, los objetos que maneja son los mismos que usa el procesador); se caracteriza por su condición de poseer una amplia gama de operadores. La generalidad del lenguaje, así como su carencia de restricciones, lo hace más conveniente que otros lenguajes de alto nivel para ciertas tareas. Esto ha hecho que sea ampliamente usado para el desarrollo de sistemas operativos (Unix y sus derivados), compiladores y para utilerías de muy diversas características.

A pesar de la tipificación de C como "Lenguaje de Sistemas", sus usos se han extendido mucho, gracias al soporte de muchas empresas que han desarrollado un gran número de librerías de funciones para su aplicación a diferentes áreas. También su popularidad ha ido creciendo en respuesta a las facilidades ofrecidas por los compiladores actuales.

Estos dos hechos han propiciado la utilización de C no sólo en el desarrollo de sistemas operativos, sino también en aplicaciones administrativas, juegos y en general para toda clase de programas tales como dBase y Norton Utilities, entre otros que fueron desarrollados en C.

Una característica muy importante de C es la utilización de direcciones a través de apuntadores. C no posee procedimientos tipo Pascal, sino que todo es desarrollado por medio de funciones. Sin embargo, existen funciones que no regresan ningún valor, resultando parecidas a los procedimientos en Pascal, la diferencia radica en que no es posible pasar argumentos por referencia, todo argumento es pasado por valor, excepto cuando se trata de una estructura, en cuyo caso se pasa un apuntador a esa

estructura.

Esto significa que cuando se requiera que sea modificado un valor pasado a una función es necesario pasarle su dirección, simulando un proceso de referencia, de manera que la función pueda actuar sobre el objeto original.

Por otra parte el nuevo estándar ANSI C aumentará la capacidad de portabilidad de los programas escritos en C.

Los escritores de compiladores prometen con esto, que si el programa de aplicación desarrollado, esta dentro de las fronteras especificadas por el estándar, el programa compilará y correrá con éxito en cualquier computadora que soporte el compilador ANSI. Desde luego si se supone que la aplicación correrá en DOS la portabilidad no será un requisito indispensable, sin embargo si se decide llevar el software a un ambiente UNIX o a una minicomputadora DEC o IBM el ANSI C deberá ser utilizado.

3.6.2 IMPLEMENTACIÓN

La implementación de los módulos que componen el Sistema se previó de manera flexible, para lo cual se pensó en adquirir utilerías comerciales para la interface con el usuario (menus y ventanas) y para la manipulación de la Base de datos.

Code Base es un paquete de librerías, desarrolladas en lenguaje C para la manipulación de archivos de dBase y generación de menus y ventanas; por lo que al cumplir con los requisitos, fue el elegido.

PRESENTACIÓN DE MENUS PARA SELECCIÓN DE SERIES

Para la presentación de menus se desarrollo el programa fpmenu.c, el cual lee un header menus.h y extrae información para generar estos menus en pantalla. En el cuarto nivel extrae los datos del menú de un archivo .dbf formato dBase, usando rutinas de Code Base V 4.2.

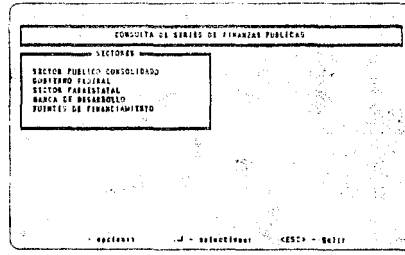
A continuación se muestra una porción del programa para la presentación de menus.

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <d4base.h>
```

```
#include <w4.h>
#include "menus.h"

void main()
{
        int  main_menu,w_tit,w_pie,i,num;

        w4clear( -1 ),          /* limpia la pantalla */
        w4cursor( -1,-1 ),      /* desaparece el cursor */

        /* Abre los archivos FPCVDES, FPDGRALS y sus respectivos
             archivos de indice para extraer de ellos i nformación
             para el cuarto y último nivel de menus */
        d4lock_code(2);
        b_ref = d4use("F:\\EJE\\FPUBLI\\DATOS\\FPCVDES");
        i_ref = i4open("F:\\EJE\\FPUBLI\\DATOS\\FPCVDES");
        b_ref2 = d4use("F:\\EJE\\FPUBLI\\DATOS\\FPDGRALS");
        i_ref2 = i4open("F:\\EJE\\FPUBLI\\DATOS\\FPDGRALS");
        if ( b_ref < 0 || i_ref < 0 || b_ref2 < 0 || i_ref2 < 0 ) {
             aviso(ERR_DAT);
             w4exit(1);
        };

        /* Extrae la posición de inicio de cada uno de los sectores para la busqueda
             de una palabra clave dentro del sector */
        d4select(b_ref2);  i4select(i_ref2);
        d4seek_str("B");  pos_b = d4recno()-1L;
        d4seek_str("D");  pos_d = d4recno()-1L;
        d4seek_str("E");  pos_e = d4recno()-1L;
        d4seek_str("F");  pos_f  = d4recno()-1L;
        numrecs = d4reccount();

        /* Define las ventanas y menus */
        w_tit =  w4define( 1,1, 3,79 ),          /* Define la ventana de titulo */
        w4border( DOUBLE_TOP, F_WHITE ),         /* Especifica un tipo de borde */
        w4activate( w_tit );                     /* Activa la ventana */
        /* Dibuja el texto centrado, en la ventana definida */
        w4centre( 1, "CONSULTA DE SERIES DE FINANZAS PUBLICAS" );

        w_pie=w4define(24,0,24,79);              /* Define la ventana de indicaciones */
        w4attribute(B_WHITE);                    /* Especifica un atributo de fondo */
        w4activate( w_pie );                     /* Activa la ventana */

        /* Formatea en cad las indicaciones que se desplagarán
             en la ventana y las dibuja centradas */
        sprintf(cad,"   %c %c - opciones      %c-+ - seleccionar     <ESC> - Salir   ",24,25,27);
        w4centre(0,cad);

        /* Define la ventana para el menu principal */
        main_menu   = w4define( -1,-1,-1,-1 ),   /* Define la ventana */
        w4border( DOUBLE_TOP, F_WHITE ),  /* Especifica un tipo de borde*/
        w4 title("SECTORES ", F_WHITE | F_INTENSE ); /* Especifica un titulo */
```

```
/* Define las opciones que contendrá el menu */
n4skip_over( n4( "" ), 1 );
for (i = 0; i < MAXSECTOR; i++) {
      n4(sectores[i]);
      switch(i) {
            case 0 : n4action( func2BDF);  n4parm(i); break;
            case 1 : n4action( func2BDF);  n4parm(i); break;
            case 2 : n4action( func2A);    break;
            case 3 : n4action( func2E);    break;
            case 4 : n4action( func2BDF);  n4parm(i);
      }.
}
n4skip_over( n4( "" ), 1 ).
/* Elige la posición inicial en donde se dibujara el menu */
n4calc( main_menu, 4,1 ).
w4select( main_menu ).                    /* Selecciona la ventana */
/* Activa el menu para que se lleven a cabo las funciones indicadas
      en n4action, de acuerdo con la opción seleccionada */
n4activate( main_menu ).

/* Desactiva las ventanas definidas, liberando la memoria ocupada por ellas */
w4deactivate(main_menu).          w4close(main_menu),
w4deactivate(w_pie).              w4close(w_pie),
w4deactivate(w_tit).              w4close(w_tit).

d4close_all().          /* Cierra los archivos de dBASE abiertos */
w4clear( -1 ).          /* limpia la pantalla */
w4cursor_size( 7,7 ).   /* Devuelve al cursor su tamaño normal */
w4exit(0);              /* Termina la ejecución del programa */
}
```

Esta parte del programa define el menu principal y permite la selección de un sector para la consulta de información, desplegando la pantalla que se muestra a continuación:



Los dos niveles de menus posteriores se despliegan de la misma forma, pero en el cuarto, la información mostrada se extrae de la base de datos; tomando del archivo fpcvdes.dbf el conjunto de series conceptuales correspondientes a la selección hecha de acuerdo a los niveles superiores elegidos.

La función que lleva a cabo esa tarea se muestra a continuación·

```
static int funcion(int val)
{
        int menu4,i,num;
        char cmpser[8],titulos[5],*ap1;

        /* Limpia el arreglo en donde almacenará las series extraídas del archivo */
        for(i = 0; i < 370; i++)
            limpia(arr[i] renglon);

        /* Selecciona la tabla de series conceptuales y su archivo de índice */
        d4select(b_ref);  i4select(i_ref);

        /* Guarda en la variable serie la clave de las series a buscar para la selección hecha */
        sprintf(serie,"%s    ",nivel3[val]);
        sprintf(titulos," %s ",nivel3[val]);

        /* Busca la serie indicada.
            Si la encuentra, mueve el apuntador al archivo a ese registro; si no, regresa */
        if ((num = d4seek_str(serie)) != 0) {
            aviso(NO_EXISTE);
            return(num);
        }
        menu4 = w4define(-1,-1,-1,-1);
        w4border (SINGLE,F_WHITE);
        n4skip_over( n4( "" ), 1 );

        /* Asigna el apuntador ap1 al contenido del registro */
        ap1 = f4record();
        /* Copia el contenido del archivo al arreglo mientras la serie pertenezca a la clasificación*/
        for ( i = 0;; i++) {
            d4skip(1L);
            strncpy(arr[i].renglon, ap1 ,68);  /* Copia el contenido del */
            arr[i] renglon[68]='\0';           /* registro al arreglo   */
            substr(cmpser,arr[i] renglon,1,3);
            streat(cmpser,"    ");
            if (strcmp(serie,cmpser) != 0)
                break;
            n4( arr[i] renglon );          /* Define la serie almacenada como parte del menu  */
            n4action( funcion2 );          /* Especifica la llamada a funcion2 */
            n4parm(i);                     /* Pasa como parámetro de entrada a */
                                           /* funcion2 la i seleccionada en el menu  */
```

```
        /* El número de elementos que pueden almacenarse en el arreglo,  es de 370 */
        if (i == 369)    break;
    }
    n4skip_over( n4( "" ), 1 );
    if (i == 0) {
        aviso(NO_ITEMS), return 0;
    }
    n4calc( menu4, 1,10 );
    w4select( menu4 );
    w4border( SINGLE, F_WHITE );
    w4title( 0,-1, titulos, F_WHITE | F_INTENSE );
    n4activate( menu4 );

    w4deactivate(menu4);    w4close(menu4);
    return(0);
}
```

En el último nivel de consulta, la información presentada es extraida del archivo de series fpdgrals.dbf, como muestra la función de abajo.

```
static int funcion2(int val)
{
        char priser[4],segser[4],renglon[60],reg_base[70],regbase2[62];
        char renglon3[70],titulos[7],clave[8],descrip[61];
        char *ap1,sercmp[6];
        int  cont=0,i,num,menu_t;

        for(i = 0; i < 500; i++) {
                limpia(arre[i] reng);
                arre[i] lotu = 0;
        }
        substr(serieb,arr[val] renglon,1,5);
        sprintf(titulos," %s ",serieb),

        menu_t = w4definet(1,6,6,79),        w4popup(),
        w4border (SINGLE,F_WHITE);
        w4title(0,-1, titulos, F_WHITE | F_INTENSE );

        d4select(b_ref);        i4select(i_ref);
        d4seek_str(serie),
        d4go( d4recno() + (long) val + 1L);
        ap1 = d4record();
        strncpy (reg_base, ap1 ,68); reg_base[68]='\0',
        substr(serie3,reg_base,1,7);
        strcat(serie3," ");
        substr(regbase2,reg_base,8,60);
        strcat(serie3,regbase2),

        d4select(b_ref2),       i4select(i_ref2);
        num =d4seek_str(serieb),
```

```
switch(num) {
      case -1:
      case  2:
      case  3:
            aviso(NO_EXISTE);
            return(num);
};
if (sector == 'A' || sector == 'B') {
      w4select(menu_1);
      substr(priser,serieb,0,1);
      for(i = 0; i < 4; i++) {
            substr(segser,letsec[i],1,1);
            if(priser[0]==segser[0]) {
                  strcpy(renglon3,priser);
                  strcat(renglon3,"       ");
                  substr(renglon,sectores[i],0,40);
                  strcat(renglon3,renglon);
                  strcat(renglon3,"                    ");
                  break;
            }
      }
      w4(0,1,renglon3);
      substr(priser,serieb,0,2);
      for(i = 0; i < 27; i++) {
            substr(segser,sect2[i],1,2);
            if(strcmp(priser,segser)==0) {
                  strcpy(renglon3,priser);  strcat(renglon3,"    ");
                  substr(renglon,sect2[i],3,56);
                  strcat(renglon3,renglon);
                  strcat(renglon3,"       ");
                  break;
            }
      }
      w4(1,1,renglon3);
      substr(priser,serieb,0,3);
      for(i = 0; i < 99; i++) {
            substr(segser,sect3[i],1,3);
            if(strcmp(priser,segser)==0) {
                  strcpy(renglon3,priser); strcat(renglon3,"    ");
                  substr(renglon,sect3[i],4,51); strcat(renglon3,renglon);
                  strcat(renglon3,"          ");
            }
      };
      w4(2,1,renglon3);
      w4(3,1,serie3);
} else {
      switch(sector)  {
            case 'B': strcpy(arre[0] reng,"B       ");
                  strcat(arre[0] reng," INGRESOS (TOTAL CONTABLE)    ");
                  break;
            case 'D': strcpy(arre[0] reng,"D       ");
                  strcat(arre[0] reng," FUENTES DE FINANCIAMIENTO    ");
                  break;
```

```
            case 'F': strcpy(arre[0].reng,"F       ");
                      strcat(arre[0].reng," DEFICIT FINANCIERO TOTAL     ");
                      break;
        };
        strcat(arre[0].reng,"                              ");
        substr(priser,serieb,0,2);
        for(i = 0; i < 27; i++) {
            substr(segser,sect2[i],1,2);
            if (strcmp(priser,segser) == 0) {
                strcpy(arre[1].reng,priser);
                strcat(arre[1].reng,"    ");
                substr(renglon,sect2[i],3,56);
                strcat(arre[1].reng,renglon);
                strcat(arre[1].reng,"       ");
                cont++;
                break;
            }
        }
        substr(priser,serieb,0,3);
        for(i = 0; i < 99; i++) {
            substr(segser,sect3[i],1,3);
            if (strcmp(priser,segser) == 0) {
                strcpy(arre[2].reng,priser);
                strcat(arre[2].reng,"    ");
                substr(renglon,sect3[i],4,51);
                strcat(arre[2].reng,renglon);
                strcat(arre[2].reng,"           ");
                cont++;
                break;
            }
        }
};
ap1 = f4record();
i=cont;
while(1) {
        strncpy(renglon3 ,ap1,70);          renglon3[70]='\0';
        strncpy(sercmp,&renglon3[1],5); sercmp[5]='\0';
        substr(clave,renglon3,1,7);
        substr(descrip,renglon3,8,60);
        strcpy(arre[i].reng,clave);
        strcat(arre[i].reng,"   ");
        strcat(arre[i].reng,descrip);
        if (strcmp(serieb,sercmp) != 0)
            break;
        d4skip(1L);
        i++;
        /* El número de elementos que pueden almacenarse en el arreglo es de 500 */
        if (i==499)     break;
}
if (i == 0) { aviso(NO_ITEMS);  return 0; }

menu_prin(1,i,titulos);
w4deactivate(menu_t);   w4close(menu_t);
```

```
        return(0);
}
```

sector es una variable globlal de tipo caracter, que indica cual fue el sector elegido en el primer nivel para la consulta. Esta variable toma los valores mostrados a continuación:

sector = 'A' Para el Sector Paraestatal
sector = 'F' Para el Sector Público Consolidado
sector = 'B' Para el Sector Gobierno Federal
sector = 'D' Para el Sector Fuentes de Financianiento
sector = 'E' Para el Sector Banca de Desarrollo

En el Sector Paraestatal y Banca de desarrollo las series de nivel superior solo se utilizan para clasificar las series del último nivel. Sin embargo para los otros sectores estas series además representan totales y pueden ser consultadas como tales.

La función anterior almacena en un arreglo el conjunto de series que se encuentran en la base de datos para la selección hecha de acuerdo a los niveles elegidos. Para los dos sectores mencionados, las series de nivel superior se muestran en una ventana aparte a manera de título, a diferencia de los otros tres sectores en los que estas series son almacenadas dentro del mismo arreglo que se utiliza para la consulta de series.

Una vez que se ha llenado el arreglo, la función anterior llama a la función menu_prin(), la cual muestra en pantalla las series contenidas en el arreglo y permite manipularlas de diversas maneras. Dicha función se muestra a continuación:

```
menu_prin(in,fi,titu)
int in,fi;
char *titu;
{
        struct tm *ptr;
        time_t lt;
        int ban,reg,pos,items,reg_aux,menu5,pie4,i,cont,fppres,num_el;
        char seriebis[8],serietri[8],hora[15],repsal[15];
        FILE *lot_ser,*archbat;

        if (sector=='B' || sector=='D' || sector=='F') {
            menu5 = w4define(1,6,24,79);
            w4title(0,-1, titu, F_WHITE | F_INTENSE );
            num_el = 22;
        } else {
            menu5 = w4define(7,6,24,79);
```

```
    num_el = 16;
}
w4popup();      w4border( SINGLE, F_WHITE );
w4activate(menu5);

pie4 = w4define(24,36,24,49).     w4popup();
w4attribute( B_WHITE );
w4activate(pie4);
w4(0,0," <F1> - Ayuda ");

w4select(menu5);
w4attribute( F_WHITE );
items = (fi < num_el) ? fi : num_el;
for (i = 0; i < items; i++)
    normal(i,i);
inverso(0,0);
ban = 1; reg = 0; pos = 0;
while (ban == 1) {
    switch( g4char()) {
    case 18432          /* flecha arriba    */
        if(reg >= in) {
            normal(pos,reg);
            if (pos > 0)  pos--;
            else          w4scroll(-1);
            reg--;
            inverso(pos,reg);
        };
        break;
    case 20480 :        /* flecha abajo     */
        if (reg < fi-1) {
            normal(pos,reg);
            if (pos < num_el-1)
                pos++;
            else
                w4scroll(1);
            reg++;
            inverso(pos,reg);
        };
        break;
    case 18688 .        /* PgUp */
        if(reg-num_el >= in-1) {
            w4scroll(num_el);
            reg_aux = reg;
            for(i = pos; i < num_el; i++) reg_aux++;
            items = (((reg_aux-num_el*2) >= in-1) ? reg_aux-num_el*2 : in-1);
            if (items == in-1) {
                for (i = items; i < num_el; i++)
                    normal(i-items,i);
                inverso(pos,pos);
                reg = pos;
            } else {
                for (i = items; i < reg_aux-num_el; i++)
                    normal(i-items,i);
```

```
                    inverso(pos,reg-num_el);
                    reg -= num_el;
                }
            };
            break;
        case 20736 :        /* PgDn */
            if (reg+num_el < fi) {
                w4scroll(num_el);
                reg_aux = reg;
                for(i = pos; i > 0; i--)
                    reg_aux--;
                items = (((reg_aux+num_el*2) < fi-1) ? reg_aux+num_el*2 : fi);
                for (i = reg_aux+num_el; i < items; i++)
                    normal(i-reg_aux+num_el,i);
                inverso(pos,reg+num_el);
                reg += num_el;
            };
            break;
        case 18176 :        /* Home */
            w4clear(0);
            items = (fi < num_el) ? fi : num_el;
            for (i = 0; i < items; i++)
                normal(i,i);
            inverso(0,0);
            reg = pos = 0;
            break;
        case 20224 :        /* End */
            w4clear(0);
            items = (fi < num_el) ? fi : num_el;
            for (i = 0; i < items; i++)
                normal(i,fi-(items-i));
            reg = fi-1; pos = items-1;
            inverso(pos,reg);
            break;
case 27 :       /* Escape */
            ban = 0;
            break;
case 13 :       /* Enter */
            tppres=menu_pres();
            if (tppres==0 || tppres==1 || tppres==2) {
                sub_str(seriebis,arre[reg],reng,0,7);
                d4close();
                menu2(seriebis,op);
                w4select(menu5);
                b_ref = d4use("F:\\EJE\\FPUBLI\\DATOS\\FPCVDES");
                i_ref  = i4open("F:\\EJE\\FPUBLI\\DATOS\\FPCVDES");
                b_ref2 = d4use("F:\\EJE\\FPUBLI\\DATOS\\FPDGRALS");
                i_ref2  = i4open("F:\\EJE\\FPUBLI\\DATOS\\FPDGRALS");
            }
            break;
case 15104          /* F1 */
            w4display(" OPCIONES : ",
                "FLECHA ARRIBA : serie anterior",
```

```
                "FLECHA ABAJO   : serie siguiente",
                "HOME           : primera serie",
                "END            : última serie",
                "PG UP          : página anterior",
                "PG DN          : página siguiente",
                "RETURN         : mostrar datos y valores",
                "F1             : ayuda",
                "F2             : marcar/desmarcar",
                "F3             : reportes",
                "F4             : exportar a lotus",
                "ESC            : salida",
                (char *) 0),
        break;
case 15360 :                    /* F2 */
        arre[reg].lotu = !arre[reg].lotu;
        inverso(pos,reg);
        break;
case 15616 :                    /* F3 */
        tppres=menu_pres();
        if (tppres==0 || tppres==1 || tppres==2) {
                tppres=menu_ops();
                if (tppres==0 || tppres==1) {
                        lt = time('\0');
                        ptr = localtime(&lt);
                        sprintf(hora,"SE%d%d%d%d.dat",ptr->tm_hour,ptr->tm_min,ptr->tm_sec);
                        if ((lot_ser = fopen(hora,"w+")) == NULL) {
                            aviso(2);
                            break;
                        };
                        cont = 0;
                        for(i = 0; i < 500; i++) {
                            if ( arre[i].lotu == 1) {
                                    substr(serietri,arre[i].reng,0,7);
                                    fprintf(lot_ser,"%s\n",serietri);
                                    cont++;
                            };
                        };
                        fclose(lot_ser);
                        if (cont > 0) {
                            if ((archbat = fopen("c:\\fprpt.bat","w")) == NULL) {
                                    w4display(" DISCO LLENO ",(char *) 0);
                                    break;
                            }
                            d4close_all();
                            nombre_sal(hora,repsal);
                            reporte(hora,repsal,op);
                            if(tppres==0)   {
                                    fprintf(archbat,"@echo off\n");
                                    fprintf(archbat,"type %s>PRN",repsal);
                            } else {
                                    fprintf(archbat,"@echo off\n");
                                    fprintf(archbat,"F:\\EJE\\FPUBLI\\we %s",repsal);
                            };
```

```
                        fclose(archbat);
                        w4clear( -1 );
                        w4exit(0);
                    } else
                        w4display(" No existen series marcadas ",
                                "sólo se imprimen las series marcadas", (char *)0);
                },  /* menu_ops */
            }; /* menu_pres */
            w4select(menu5);
            break;
        case 15872 :        /* F4 */
            tppres=menu_pres();
            if (tppres==0 || tppres==1 || tppres==2) {
                lt = time('\0');
                ptr = localtime(&lt);
                sprintf(hora,"SE%d%d%d.dat",ptr->tm_hour,ptr->tm_min,ptr->tm_sec);
                if ((lot_ser = fopen(hora,"w+")) == NULL) {
                    aviso(2);
                    break;
                };
                cont = 0;
                for(i = 0; i < 500; i++) {
                    if ( arre[i].lotu == 1 ) {
                        substr(serietri,arre[i].reng,0,7);
                        fprintf(lot_ser,"%s\n",serietri);
                        cont++;
                    };
                };
                fclose(lot_ser);
                if (cont > 0) {
                    if ((lot_ser = fopen("c:\\fpexpo bat","w")) != NULL) {
                        fprintf(lot_ser,"@echo off\n");
                        fprintf(lot_ser,"F:\\EJE\\FPUBLI\\FPTO123 %s %d %c",hora,cont,op);
                        d4close_all();
                        fclose(lot_ser);
                        w4clear(-1);
                        w4exit(0);
                    } else w4display(" Disco lleno ",(char *)0);
                } else
                    w4display(" No existen series marcadas ",
                            "sólo se exportan las series marcadas", (char *)0);
            };
            w4select(menu5);
        }
    }
    w4select(pie4);    w4deactivate(pie4);    w4close(pie4);
    w4select(menu5); w4deactivate(menu5); w4close(menu5);
    w4cursor_size(-1,-1);
    return(0);
}
```

Esta función despliega el conjunto de series y permite escrolear la ventana para poder observarlas todas. Se puede desplazar la barra iluminada utilizando las teclas de flechas, Pg Up, Pg Dw, Home y End.

Si la tecla que se pulsa es la F1, se muestra una ventana de ayuda en la que se indican las opciones con que se cuenta.

Para poder observar las valores para los diferentes períodos de tiempo correspondientes a una serie, se debe posicionar la barra sobre la serie deseada y pulsar la tecla de Return. La función en todo momento valida la tecla pulsada y si es la antes mencionada, llama a la función menu_pres(), que pregunta por el tipo de información deseada (Flujo nominal, presupuesto original o presupuesto modificado) y a continuación llama a la función menu2, la cual extrae del archivo de valores los correspondientes a la serie y los muestra en pantalla.

Los datos mostrados pueden ser manipulados de dos formas:

1) Pueden ser vaciados en un archivo para ser consultados en pantalla o impresos en papel a manera de reporte.

2) Pueden ser vaciados en una hoja de cálculo de Lotus 1-2-3.

Para poder generar un reporte simple de las series, deben marcarse las elegidas presionando la tecla F2 cuando la barra se encuentre sobre cada una de ellas y a continuación pulsar la tecla F3. Cuando esto ocurre se genera un archivo temporal en el que se graban las claves de las series marcadas y se llama a la función reporte, la cual genera el reporte deseado.

Para generar una hoja de Lotus con los valores de las series, de igual forma que para la generación de reportes deben marcarse las elegidas y en seguida pulsar la tecla F4.

Cuando esto ocurre se genera un archivo temporal en el que se graban las claves de las series marcadas, se genera un archivo batch que hace la llamada al programa que genera la hoja y se termina la ejecución del primero para que continue el de exportación.

El control del sistema lo lleva el programa menufp.bat, que ejecuta el batch generado en este punto.

Las rutinas de generación de reporte y de exportación a Lotus se muestran a continuación:

Generación de reporte

```
double ar[2000];
union u {
      double doble;
      char   cadena[8];
} uni_on;
char c=-1;

int reporte(par1,par2,par3)
char *par1, *par2;
char  par3;
{
          char serieb[8],clave[3];
          int i,p,t,k,rep;

          valtipo = par3;
          /* Abre el archivo que contiene la clave de las series */
          if ((impr_ser = fopen(par1,"r")) == NULL) {
               w4display(" ERROR DE ARCHIVO ","Error al abrir el archivo",(char *) 0);
               return(1);
          }
          /* Abre el archivo de salida */
          if ((salida = fopen(par2,"w")) == NULL) {
               w4display(" ERROR DE ARCHIVO ","Error al crear el archivo",(char *) 0);
               fclose(impr_ser);
               return(1);
          };
          d4lock_code(2);
          b_ref2 = d4use("F:\\EJE\\FPUBLI\\DATOS\\FPDGRALS");
          i_ref2 = i4open("F:\\EJE\\FPUBLI\\DATOS\\FPDGRALS");
          if (b_ref2 < 0 || i_ref2 < 0 ) {
               w4display(" No se puede abrir el catálogo de series ",(char *) 0);
               fclose(impr_ser);  fclose(salida);
               return(0);
          }
          d4select(b_ref2),      i4select(i_ref2),
          switch(valtipo) {
               case 'N':  strcpy(archval,"F:\\EJE\\FPUBLI\\DATOS\\FPVALR.DAT");
                          break;
               case 'O':  strcpy(archval,"F:\\EJE\\FPUBLI\\DATOS\\FPVALO.DAT");
                          break;
               case 'M':  strcpy(archval,"F:\\EJE\\FPUBLI\\DATOS\\FPVALM.DAT");
          }
          /* Abre el archivo de valores para extraer la información
               de acuerdo al tipo de información requerida */
          if ((fp = fopen(archval,"rb")) == NULL) {
               w4display(" ERROR DE ARCHIVO ",
               "No se pudo abrir archivo de valores",(char *) 0);
               fclose(impr_ser);  fclose(salida);      d4close();
               return(0);
          }
```

```
/* lee el primer dato del archivo, que corresponde al rango
     de años para los cuales se tiene información almacenada */
fread(&valordbl,sizeof(valordbl),1,fp);
anioin  = (int)(valordbl/10000);
gblaniof = (int)(valordbl-(anioin*10000));
anioin-=1900; gblaniof-=1900;
aniosdif = (gblaniof - anioin) + 1;
for (i=0;i<8;i++)
     uni_on.cadena[i] = c;
/* Extrae del archivo de series temporal las claves de cada
     una de las series a incluir en el reporte */
while(fgets(serieb,9,impr_ser) != NULL) {
/* Hace la busqueda en la tabla de series para extrer sus datos */
num =d4seek_str(serieb);
switch(num){
     case -1:
     case  1:
     case  2:
     case  3:
          break;
     case  0:
          ap1 = f4record();
          strncpy(valores,ap1,168); valores[168] = '\0';
          substr(dgclv_v,valores,1,7);
          substr(dgdes_v,valores,8,60);
          substr(dguni_v,valores,68,25);
          dgper_v = valores[93];
          dgtip_v = valores[94];
          substr(dgani_v,valores,95,2);
          substr(dgpei_v,valores,97,3);
          substr(dganf_v,valores,100,2);
          substr(dgpef_v,valores,102,3);
          substr(dgans_v,valores,105,2);
          substr(dgps_v, valores,107,3);
          substr(dgtue_v,valores,110,20);
          substr(dgres_v,valores,130,4);
          substr(dgfua_v,valores,134,9);
          substr(dgarea_v,valores,143,2);
          substr(rini_v,valores,145,8);
          substr(oini_v,valores,153,8);
          substr(mini_v,valores,161,7);
          switch(dgtip_v) {
               case 'C': strcpy(vec_tipo,"CONCEPTUAL");   break;
               case 'G': strcpy(vec_tipo,"GENERADA");     break;
               case 'B': strcpy(vec_tipo,"BASICA ");
          };
          switch(dgper_v) {
               case 'D': strcpy(vec_per,"DIARIA");        break;
               case 'S': strcpy(vec_per,"SEMANAL");       break;
               case 'Q': strcpy(vec_per,"QUINCENAL");     break;
               case 'M': strcpy(vec_per,"MENSUAL");       break;
               case 'B': strcpy(vec_per,"BIMESTRAL");     break;
               case 'T': strcpy(vec_per,"TRIMESTRAL");    break;
```

```
        case 'C': strcpy(vec_per,"CUATRIMESTRAL");break;
        case 'X': strcpy(vec_per,"SEMESTRAL");      break;
        case 'A': strcpy(vec_per,"ANUAL");
};
switch(valtipo) {
        case 'N': regini = atol(rini_v);  break;
        case 'O': regini = atol(oini_v);  break;
        case 'M': regini = atol(mini_v);
}
for(i = 0; i < 10; i++) {
        substr(clave,areas[i],0,2);
        if( strcmp(clave,dgarea_v) == 0)
                break;
}
if (i < 9)      substr(vec_area,areas[i],3,52);
else            substr(vec_area,areas[9],3,52);
recorta(dgdes_v);
/* Vacia en el archivo de salida, los datos leídos en la tabla  */
fprintf(salida,"\nSerie      : %s  %s\n",dgclv_v,dgdes_v);
fprintf(salida,"Area       : %s\n",vec_area);
fprintf(salida,"Fuente     : %s\n",dgfue_v);
fprintf(salida,"Unidades   : %s\n",dguni_v);
fprintf(salida,"Periodicidad : %s\n",vec_per);
fprintf(salida,"Tipo       : %s\n",vec_tipo);
fprintf(salida,"PERIODOS          Año    Periodo\n");
fprintf(salida,"          LIBERADO     %s      %s\n",dgans_v,dgps_v);
fprintf(salida,"          INICIAL      %s      %s\n",dgani_v,dgpei_v);
fprintf(salida,"          FINAL        %s      %s\n",dganf_v,dgpef_v);
fprintf(salida,"Fecha de la última actualización : %s\n",dgfua_v);
fprintf(salida,"VALORES :\n");
aniosdif = (gblaniof - anioin) + 1;
/* llama a la función que lee los valores correspondientes a la serie */
if ( bus_valores()==1 )
        break;
/* Si encontró valores los graba en el archivo de salida */
fprintf(salida,"    Año  Periodo    valor     Año  Periodo     valor\n");
t=ban=0;
anioseg = atoi(dgans_v);
perseg  = atoi(dgps_v); aniofin = gblaniof;
if(aniofin > anioseg)   aniofin= anioseg;
aniosdif=aniofin-anioin+1;
if (anioseg%2==0)       ban=1;
for (k=anioin ; k < aniofin; k+=2) {
        for(p=1 ; p < per+1 ; p++) {
                if ((esvalor(ar[t]) == 1) || (esvalor(ar[t+per]) == 1)) {
                        if (esvalor(ar[t]) == 1)
                                fprintf(salida,"      %d   %2d %15.2f",k,p,ar[t]);
                        else    fprintf(salida,"                        ");
                        if (esvalor(ar[t+per]) == 1) {
                          if ((ban == 0) || (k!=anioseg-1) || ((k==anioseg-1) && (p<=perseg)))
                                fprintf(salida,"      %d   %2d %15.2f\n",k+1,p,ar[t+per]);
                          else
                                fprintf(salida,"\n");
```

```
                        } else
                                fprintf(salida,"\n");
                        };
                        t++;
                    };
                    t+=peri;
                };
                if (ban == 0){
                    for(p=1;p<=perseg;p++) {
                        if(esvalor(ar[t])==1)
                            fprintf(salida,"     %d   %2d %15.2f\n",k,p,ar[t]);
                        t++;
                    }
                };
                fprintf(salida,"%c",12);
            };
        }
        fclose(fp);     fclose(salida);    fclose(impr_ser);
        d4close();
        return(0);
}

int bus_valores()
{
        int z;

        /* Calcula el número de periodos por año de acuerdo a la periodicidad de la serie */
        peri= maxobs(anioin,dgper_v);
        /* Calcula el número de datos a leer del archivo de valores (el máximo es 2000) */
        dif = (int) aniosdif * peri;
        if (dif > 2000) {
            w4display(" MUCHOS DATOS ","SE CARGARAN SOLO 2000 DATOS",
            (char *) 0);
            dif = 2000;
        };
        /* Mueve el apuntador al archivo de valores, a la posicion  indicada por el
        apuntador al rango de valores para esa serie que se encuentra en la tabla */
        if (fseek(fp,(long) (sizeof(ar[0]) * regini), SEEK_SET)) {
            w4display(" ERROR EN BUSQUEDA ",
            "No se localizaron datos",(char *) 0);
            return(1);
        }
        for (z=0;z<2000;z++)
            ar[z] = uni_on_doble;
        /* lee dif elementos de tipo double ( longitud sizeof(ar[0]) ),
            del archivo apuntado por fp y los almacena en ar */
        fread(&ar,sizeof(ar[0]),dif,fp);

        return(0);
}
```

Exportación a Lotus 1-2-3

Como se mencionó en el capítulo dos, todo archivo de Lotus, cuenta con una serie de códigos de operación que le dan el formato a la hoja. Dichos códigos de operación deben existir para que Lotus 1-2-3 pueda reconocerla como propia.

Para poder almacenar estos códigos se crea un archivo binario en el que deben grabarse los códigos en el orden que Lotus lo hace al generar una hoja.

Lo anterior se hizo empleando dos estructuras; una para grabar los códigos de definición de la hoja y la otra para poder grabar las celdas de tipo numéricas.

```
/* Variables para generacion de la hoja */
struct tlotus {
        int codigo;
        int lon;
        char datos[1200];
} reglotus;

struct tcelda {
        int codigo;
        int lon;
        char formato;
        int ncol;
        int nren;
        union v {
                int   entero;
                double doble;
        } valor;
} celda;

long savepos;
int  i,iaux,series,ren,numval,maxval; char saux[256],arc[60];
unsigned int waux;
FILE *fph;

main(argc,argv)
int argc;
char *argv[];
{
        int num,p,k,t;
        char valores[170],*ap1;

        maxval=0;
        saux[0]='\0';
        if(argc<4) {
            w4display(" Datos incompletos ",
                "El número de argumentos es inválido",argv[1],argv[2],argv[3], (char *) 0);
            w4exit(0);
        }
```

```
/* Abre el archivo que contine la lista de series a grabar en la hoja */
if ((fps = fopen(argv[1],"r")) == NULL) {
      w4display(" No existe el archivo o está protegido",argv[1], (char *) 0);
      w4exit(0);
}
strcpy(cadena,argv[3]);
switch(cadena[0]) {
      case 'N': strcpy(archval,"F:\\EJE\\FPUBLI\\DATOS\\FPVALR.DAT");
            break;
      case 'O': strcpy(archval,"F:\\EJE\\FPUBLI\\DATOS\\FPVALO.DAT");
            break;
      case 'M': strcpy(archval,"F:\\EJE\\FPUBLI\\DATOS\\FPVALM.DAT");
};
/* Abre el archivo de valores de acuerdo al tipo de información solicitada */
if ((fpv = fopen(archval,"rb")) == NULL) {
      w4display(" No se puede abrir el archivo de valores",archval,(char *) 0);
      fclose(fps);
      w4exit(0);
}
/* Lee el primer valor grabado en el archivo, que indica al rango de años
      para los cuales existe información */
fread(&valordbl,sizeof(valordbl),1,fpv);
anioin  = valordbl/10000;
gblaniof = valordbl-(anioin*10000); anioin-=1900; gblaniof-=1900;
aniosdif = (gblaniof - anioin) + 1;

/* Abre la tabla de series para extraer la descripción y el
      apuntador al conjunto de valores correspondientes a la serie */
b_ref = d4use("F:\\EJE\\FPUBLI\\DATOS\\FPDGRALS");
i_ref = i4open("F:\\EJE\\FPUBLI\\DATOS\\FPDGRALS");
if (b_ref < 0 || i_ref < 0) {
      w4display(" No existe el catálogo de datos generales o está protegido ",(char *) 0);
      fclose(fps); fclose(fpv);
      w4exit(0);
}
d4select(b_ref),      i4select(i_ref),
series=atoi(argv[2]);
if(series > 30)
      series=30;
generahoja();   /* Genera la hoja con el formato de Lotus */
i=1;
while (fgets(serie,9,fps)!=NULL && (i<=30)){
      serie[7]='\0';
      val = nv = 0;
      num=d4seek_str(serie);
      switch(num) {
            case -1:
            case  1:
            case  2:
            case  3:
                  nv=1;
                  val=1;
      };
```

```
ap1 = f4record();
strncpy(valores,ap1,168);      valores[168] = '\0';
substr(reg_des,valores,8,60);
reg_per[0] = valores[93];      reg_per[1]='\0';
substr(dgans_v,valores,105,2);
substr(dgps_v, valores,107,3);
dgans = atoi(dgans_v);
dgps  = atoi(dgps_v);
switch(cadena[0]) {
      case 'N': substr(regimi_v,valores,145,8); break;
      case 'O': substr(regimi_v,valores,153,8); break;
      case 'M': substr(regimi_v,valores,161,7);
};

/* Si la serie existe, coloca el apuntador al archivo de valores en la
     posición inicial del conjunto de valores correspondientes a esta */
val = (val == 1) ? val : pos_apunta();

/* Código de una etiqueta
   Graba la clave de la serie */
reglotus.codigo=15;
reglotus.lon=15;
reglotus.datos[0]=113;
iaux=(i-1)*4;    memcpy(&reglotus.datos[1],&iaux,2);
iaux=0;          memcpy(&reglotus.datos[3],&iaux,2);
reglotus.datos[5]=39;
strcpy(saux,serie);
memcpy(&reglotus.datos[6],saux,8);
reglotus.datos[14]=0;
fwrite(&reglotus,reglotus.lon+4,1,fph);

/* Graba la descripción de la serie */
reglotus.lon=67;
iaux=(i-1)*4+2;
memcpy(&reglotus.datos[1],&iaux,2);
/* Si la serie no existe le pone por descripción NO DISPONIBLE */
if(val == 1) && (nv == 1))
      strcpy(saux," NO DISPONIBLE ");
else  strcpy(saux,reg_des);
memcpy(&reglotus.datos[6],saux,60);  reglotus.datos[66]=0;
fwrite(&reglotus,reglotus.lon+4,1,fph);

reglotus.lon=7;
iaux=(i-1)*4+3;
memcpy(&reglotus.datos[1],&iaux,2);
reglotus.datos[6]=0;
fwrite(&reglotus,reglotus.lon+4,1,fph);
reglotus.lon=41;
iaux=(i-1)*4;    memcpy(&reglotus.datos[1],&iaux,2);
iaux=1;          memcpy(&reglotus.datos[3],&iaux,2);
reglotus.datos[5]=39;
strcpy(saux,"-------- ----------------------");
memcpy(&reglotus.datos[6],saux,34);
```

```
reglotus.datos[40]=0;
fwrite(&reglotus,reglotus.lon+4,1,fph);
if (val==0) {
      ren=2,     t=0;
      /*  celdas  */
      peri++;
      aniofin = gblaniof;
      if (aniofin > dgans)    aniofin = dgans;
      for (k=anioin; k < aniofin ; k++) {
            for(p=1 ; p < peri ; p++) {
                  fread(&valordbl,sizeof(valordbl),1,fpv);
                  if (esvalor(valordbl) == 1) {
                  /* Código de un valor de tipo entero */
                              celda.codigo=13;
                              celda.lon=7;
                              celda.formato=255; celda.ncol=(i-1)*4;
                              celda.nren=ren;
                              celda.valor.entero=k;
                              /* Graba el año */
                              fwrite(&celda,11,1,fph);
                              celda.ncol++;
                              celda.valor.entero=p;
                              /* Graba el periodo */
                              fwrite(&celda,11,1,fph);
                              /* Código de un valor de punto flotante */
                              celda.codigo=14;
                              celda.lon=13;
                              celda.formato=66;
                              col++;
                              ren=ren;
                              celda.valor.doble=valordbl;
                              /* Graba el valor de la serie correspondiente
                              al año y periodo,  anteriormente grabados */
                              fwrite(&celda,17,1,fph);
                              ren++;
                              t++;
                        }; /* if */
      };  /* for p */
};  /* for k */
/* Escribe los valores del último año (año de seguridad) */
for(p=1 ; p <= dgps ; p++) { fread(&valordbl,sizeof(valordbl),1,fpv);
      if (esvalor(valordbl) == 1) {
            celda.codigo=13;
            celda.lon=7;
            celda.formato=255;
            celda.ncol=(i-1)*4;
            celda.nren=ren;
            celda.valor.entero=k;        /* año */
            fwrite(&celda,11,1,fph);
            celda.ncol++;
            celda.valor.entero=p;        /*periodo*/
            fwrite(&celda,11,1,fph);
```

```
                        celda.codigo=14;
                        celda.lon=13;
                        celda.formato=66;
                        celda.ncol++;
                        celda.nren=ren;
                        celda.valor.doble=valordbl;
                        fwrite(&celda,17,1,fph); ren++;
                        t++;} /* if */
                    }; /* for p */
                    numval=t;
                    if(maxval<numval)
                        maxval=numval;
                };
                i++;
            };    /* end while */
            savepos = ftell(fph);
            maxval++;
            fseek(fph,16,0);
            fwrite(&maxval,2,1,fph);
            fseek(fph,savepos,0);

            /* Código de fin de archivo */
            reglotus.codigo=01;
            reglotus.lon=0;
            fwrite(&reglotus,4,1,fph);
            d4close_all();
            fclose(fps);        fclose(fpv);        fclose(fph);
            w4display("","                    Exportación realizada          ",
                        " Presione cualquier tecla para continuar... ", (char *)0);
            w4exit(0);
}

int pos_apunta()
{
        int j;

        regini = atol(regini_v);
        if (regini == 0)
            return(1);
        peri= maxobs(anioin,reg_per[0]);
        dif = (int) aniosdif * peri;
        if (fseek (fpv, (long) (sizeof(valordbl) * regini), SEEK_SET))
            return(1);
        return(0);
}

generahoja()
{
        /* Se lee el nombre que se le dará a la hoja */
        lectura();
        strcat(arc,".wk1");
```

```
/* Se crea el archivo como binario */
if ((fph=fopen(arc,"wb"))== 0) {
     /* Si no se puede crear, se cierran los archivos abiertos y termina la
     ejecución del programa */
     w4display(" No se puede crear la hoja",(char *) 0);
     d4close_all();   fclose(fps);  fclose(fpv);
     w4exit(0);
}

reglotus.codigo=0;  /* Se almacena en la estructura, el código de */
reglotus.lon=2;        /* inicio de la hoja y se graba en el archivo */
iaux=1030;
memcpy(&reglotus.datos[0],&iaux,2);
fwrite(&reglotus,reglotus.lon+4,1,fph);

reglotus.codigo=6;  /* Se almacena en la estructura, el código que */
reglotus.lon=8;    /* define el rango de celdas a utilizar en la hoja */
iaux=0;
memcpy(&reglotus.datos[0],&iaux,2);                  /*columna inicial 0 */
memcpy(&reglotus.datos[2],&iaux,2);                  /* renglón inicial 0 */
iaux = series*4-1;                                   /* columna final */
/* Por cada serie se utilizan 4 columnas (año, periodo, valor
y una celda de separación con la siguiente */
memcpy(&reglotus.datos[4],&iaux,2);
iaux=3660;                                  /* Renglón final  Este valor se actualiza al final */
memcpy(&reglotus.datos[6],&iaux,2);
fwrite(&reglotus,reglotus.lon+4,1,fph);     /* Se graba la información */

/* Código de la lista de columnas que contienen una o más celdas activas */
reglotus.codigo=150;
reglotus.lon=series*3*6;
/* La longitud de la información depende del numero de columnas a utilizar
y es una triada de valores de tipo entero para cada una de ellas */
for(i=0;i<reglotus.lon;i++) {
     reglotus.datos[i]= 0;
}       /* estos valores pueden ajustarse a cero */
fwrite(&reglotus,reglotus.lon+4,1,fph);

/* Todos los códigos se explican en el capítulo dos */
reglotus.codigo=47;
reglotus.lon=1;
reglotus.datos[0]=1;
fwrite(&reglotus,reglotus.lon+4,1,fph);

reglotus.codigo=2;
reglotus.datos[0]=255;
fwrite(&reglotus,reglotus.lon+4,1,fph);

reglotus.codigo=3;
reglotus.datos[0]=0;
fwrite(&reglotus,reglotus.lon+4,1,fph);
```

```
reglotus.codigo=4;
reglotus.datos[0]=0;
fwrite(&reglotus,reglotus_lon+4,1,fph);

reglotus.codigo=5;
reglotus.datos[0]=255;
fwrite(&reglotus,reglotus_lon+4,1,fph);

reglotus.codigo=7;
reglotus_lon=31;
iaux=0; memcpy(&reglotus.datos[0],&iaux,2);
iaux=2; memcpy(&reglotus.datos[2],&iaux,2);
reglotus.datos[4]=113;
reglotus.datos[5]=0;
iaux=9; memcpy(&reglotus.datos[6],&iaux,2);
iaux=7; memcpy(&reglotus.datos[8],&iaux,2);
iaux=20; memcpy(&reglotus.datos[10],&iaux,2);
iaux=0; memcpy(&reglotus.datos[12],&iaux,2);
iaux=2; memcpy(&reglotus.datos[14],&iaux,2);
iaux=0; memcpy(&reglotus.datos[16],&iaux,2);
iaux=2; memcpy(&reglotus.datos[18],&iaux,2);
iaux=0; memcpy(&reglotus.datos[20],&iaux,2);
iaux=0; memcpy(&reglotus.datos[22],&iaux,2);
iaux=4; memcpy(&reglotus.datos[24],&iaux,2);
memcpy(&reglotus.datos[26],&iaux,2);
iaux=72; memcpy(&reglotus.datos[28],&iaux,2);
reglotus.datos[30]=0;
fwrite(&reglotus,reglotus_lon+4,1,fph);

/* Define el ancho de cada una de las columnas utilizadas */
reglotus.codigo=8;
reglotus_lon=3;
for (i=1; i<=series; i++) {
      waux=(i-1)*4;
      memcpy(&reglotus.datos[0],&waux,2);
      reglotus.datos[2]=5;          /* Ancho de la celda para almacenar año */
      fwrite(&reglotus,7,1,fph);
      waux=(i-1)*4+1;
      memcpy(&reglotus.datos[0],&waux,2);
      reglotus.datos[2]=4;          /* Ancho de la celda para almacenar periodo */
      fwrite(&reglotus,7,1,fph);
      waux=(i-1)*4+2;
      memcpy(&reglotus.datos[0],&waux,2);
      reglotus.datos[2]=25;         /* Ancho de la celda para almacenar valor */
      fwrite(&reglotus,7,1,fph);
      waux=(i-1)*4+3;
      memcpy(&reglotus.datos[0],&waux,2);
      reglotus.datos[2]=4;          /* Ancho de la celda de separación */
      fwrite(&reglotus,7,1,fph);
}
```

```
reglotus.codigo=100;
reglotus.lon=32;
for(i=0;i<reglotus.lon;i++) {
      reglotus.datos[i]=0;
}
fwrite(&reglotus,reglotus.lon+4,1,fph);

reglotus.codigo=24;
reglotus.lon=25;
for(i=0;i<reglotus.lon;i++) {
      reglotus.datos[i]=0;
}
iaux=-1;
for(i=0;i<=5;i++)
memcpy(&reglotus.datos[i*4+1],&iaux,2);
fwrite(&reglotus,reglotus.lon+4,1,fph);

reglotus.codigo=25;
reglotus.lon=25;
for(i=0;i<reglotus.lon;i++) {
      reglotus.datos[i]=0;
}
for(i=1;i<=6;i++)
memcpy(&reglotus.datos[(i-1)*4],&iaux,2);
fwrite(&reglotus,reglotus.lon+4,1,fph);

reglotus.codigo=26;
reglotus.lon=8;
for(i=0;i<reglotus.lon;i++) {
      reglotus.datos[i]=0;
}
memcpy(&reglotus.datos[0],&iaux,2);
memcpy(&reglotus.datos[4],&iaux,2);
fwrite(&reglotus,reglotus.lon+4,1,fph);

reglotus.codigo=48;
reglotus.lon=1;
reglotus.datos[0]=0;
fwrite(&reglotus,reglotus.lon+4,1,fph);

reglotus.codigo=28;
reglotus.lon=8;
for(i=0;i<reglotus.lon;i++) {
      reglotus.datos[i]=0;
}
memcpy(&reglotus.datos[0],&iaux,2);
memcpy(&reglotus.datos[4],&iaux,2);
fwrite(&reglotus,reglotus.lon+4,1,fph);

reglotus.codigo=27;
fwrite(&reglotus,reglotus.lon+4,1,fph);
```

```
reglotus.codigo=29;
reglotus.lon=9; reglotus.datos[reglotus.lon-1]=0;
fwrite(&reglotus,reglotus.lon+4,1,fph);

reglotus.codigo=35;
fwrite(&reglotus,reglotus.lon+4,1,fph);

reglotus.codigo=103;
reglotus.lon=25;
for(i=0;i<reglotus.lon;i++)
    reglotus.datos[i]=0;
for(i=1;i<=6;i++)
    memcpy(&reglotus.datos[(i-1)*4],&iaux,2);
fwrite(&reglotus,reglotus.lon+4,1,fph);

reglotus.codigo=105;
reglotus.lon=40;
for(i=0;i<reglotus.lon;i++)
    reglotus.datos[i]=0;
for(i=1;i<=10;i++)
    memcpy(&reglotus.datos[(i-1)*4],&iaux,2);
fwrite(&reglotus,reglotus.lon+4,1,fph);

reglotus.codigo=32;
reglotus.lon=16;
for(i=0;i<reglotus.lon;i++)
    reglotus.datos[i]=0;
for(i=1;i<=4;i++)
    memcpy(&reglotus.datos[(i-1)*4],&iaux,2);
fwrite(&reglotus,reglotus.lon+4,1,fph);

reglotus.codigo=102;
fwrite(&reglotus,reglotus.lon+4,1,fph);

reglotus.codigo=76;
reglotus.lon=1;
reglotus.datos[0]=0;
fwrite(&reglotus,reglotus.lon+4,1,fph);

reglotus.codigo=37;
reglotus.lon=242;
for(i=0;i<reglotus.lon;i++)
    reglotus.datos[i]=0;
fwrite(&reglotus,reglotus.lon+4,1,fph);

reglotus.codigo=38;
fwrite(&reglotus,reglotus.lon+4,1,fph);

reglotus.codigo=39;
reglotus.lon=40;
fwrite(&reglotus,reglotus.lon+4,1,fph);
```

```
reglotus.codigo=40;
reglotus.lon=10;
iaux=4;
memcpy(&reglotus.datos[0],&iaux,2);
iaux=76;
memcpy(&reglotus.datos[2],&iaux,2);
iaux=66;
memcpy(&reglotus.datos[4],&iaux,2);
iaux=2;
memcpy(&reglotus.datos[6],&iaux,2);
memcpy(&reglotus.datos[8],&iaux,2);
fwrite(&reglotus,reglotus.lon+4,1,fph);

reglotus.codigo=41;
reglotus.lon=1;
reglotus.datos[0]=27;
fwrite(&reglotus,reglotus.lon+4,1,fph);

reglotus.codigo=42;
reglotus.lon=16;
iaux=-1;
for(i=0;i<reglotus.lon;i++)
      reglotus.datos[i]=0;
for(i=1,i<=4;i++)
      memcpy(&reglotus.datos[(i-1)*4],&iaux,2);
fwrite(&reglotus,reglotus.lon+4,1,fph);

reglotus.codigo=45;
reglotus.lon=437;
for(i=0;i<reglotus.lon;i++)
      reglotus.datos[i]=0;
for(i=0,i<104,i++)
      reglotus.datos[i]=-1;
reglotus.datos[104]=4;
for(i=107;i<=112;i++)
      reglotus.datos[i]=1;
reglotus.datos[433]='q';
 reglotus.datos[434]='q';
iaux=1;
memcpy(&reglotus.datos[435],&iaux,2);
fwrite(&reglotus,reglotus.lon+4,1,fph);
}
```

CONSULTA Y GENERACIÓN DE CUADROS ESTADÍSTICOS

Para la generación de cuadros estadisticos se desarrolló el programa FPEMITE.C, el cual genera una hoja de calculo (Lotus) en base a otra similar, en la cual se sustituyen valores de series.

La sustitución se realiza aún en fórmulas mezcladas de celdas y observaciones de series.

Este programa recibe como entradas:

a) Clave de Usuario (Username).
b) Archivo de Lotus (extension .WK1) con la definición de las observaciones requeridas.
c) Nombre del archivo a generar de Lotus (opcional).

Entrega como salida un archivo de Lotus con los valores sustituidos.

A continuación se muestran las partes más importantes del programa:

```
#define     UCONSULTA   'C'     /* Para usuarios de Consulta */
#define     UMANTENIM   'P'     /* Para usuarios Propietarios */
#define     TRUE        1
#define     FALSE       0

struct tlabel {
      char align;
      char etiqueta[1194];
};

struct tformula {
      double val;
      int longform;
      char sformula[1185];
};

struct tcelda {
      char formato;
      int ncol
      int nren;
      union {
          struct tlabel etq;
          struct tformula formula;
      } formlab;
};

union tdatos {
      char resto[1200];
      struct tcelda celda;
} datos;
```

```
static char *areas[] = {
        "A  Dir  de Estadistica Hacendaria                    ",
        "AA Subdir. de Estadisticas del Sector Paraestatal    ",
        "AB Subdir. de Egresos del Gobierno Federal           ",
        "AC Subdir  de Ingresos del Gobierno Federal          ",
        "AD Subdir. de Estadisticas de la Banca de Desarrollo ",
        "AE Subdir. de Análisis y Eval. de las Finanzas Públicas ",
        "B  Dir  de Planeación Financiera                     ",
        "C  Dir  de Análisis Macro Económico                  ",
        "D  Dir  de Análisis y Evaluación Hacendaria          ",
        "H  Unidad de Sistemas                                ",
        "XX                                                   "
};

main(argc,argv)
int  argc;
char *argv[];
{
        char strtmp[60],tmp[60],*sp,getustipo();
        int  I1,I2;

        /*  Lee clave  */
        if (argc > 1) strcpy(Mgbluser,argv[1]);
        else strcpy(Mgbluser,getenv("USUARIO"));
        /* si la clave no es proporcionada como parámetro, se toma de una
           de las variables de ambiente (variable USUARIO ) */
        sp=strchr(Mgbluser,'$');
        if (sp != NULL) {
                I2=strlen(sp)-1;
                strncpy(Mgbluser,sp+1,I2); Mgbluser[I2]='\0';
        }
        /* La longitud de la variable donde se guarda la clave de usuario es de
         cuatro caracteres, ya que asi es como se almacena en el catálogo de usuarios */
        Mgbluser[4]='\0';
        /*  Archivo de entrada (hoja de calculo patrón)  */
        if (argc > 2)
                strcpy(FIn,argv[2]);
        else {
                printf("Archivo de entrada: ");
                gets(FIn);
        };
        /*  Archivo de salida  (hoja de calculo generada)  */
        if (argc > 3)
                strcpy(FOut,argv[3]);
        else {
                strcpy(FOut,"G");   strcat(FOut,FIn);
                do {
                    printf("Archivo de salida [%s]: ",FOut); gets(strtmp);
                    if (strlen(strtmp) != 0) strcpy(FOut,strtmp);
                } while (strcmp(FIn,FOut)==0);
        };

        abre_arch(); /* Abre todos los archivos a utilizar */
```

```
/* Modo de la información  A/D (Acumulado/Desacumulado) */
if (argc > 4)
        gblmodo=toupper(argv[4][0]);
else {
        printf("Modo de la información: ");
        gets(strtmp); gblmodo=toupper(strtmp[0]);
};
if ( !((gblmodo == 'A') || (gblmodo == 'D')) )
        gblmodo = 'D';
/* Periodo base de series  (AA:PP)  */
if (argc > 5)
        strcpy(strtmp,argv[5]);
else {
        printf("Periodo base de series (AA:PP): ");
        gets(strtmp);
};
gblperiodo=1;
sp=strchr(strtmp,':');
if ( sp!=0 ) {
        /* Si el periodo no es un valor entero aborta el programa */
        if ( checkint(sp+1)==FALSE )
                abort_prg(3);
        gblperiodo=atoi(sp+1);
        *sp = '\0';
};
/* Si el año no es un valor entero aborta el programa */
if ( checkint(strtmp)==FALSE )
        abort_prg(4);
gblanio=atoi(strtmp);
/* Periodo y periodicidad del cuadro (AA:PP/M) */
if (argc > 6)
        strcpy(strtmp,argv[6]);
else {
    printf("Periodo y periodicidad del cuadro (AA:PP/P): ");
    gets(strtmp);
};
gblperiodoC=1;
gblperC='M';
sp=strchr(strtmp,'/');
if ( sp!=0 ) {
        gblperC=toupper(*(sp+1));
        *sp = '\0';
};
/* Si la periodicidad no es una periodicidad
 válida aborta el programa */
if (checkperiod(gblperC)==FALSE)
        abort_prg(5);
sp=strchr(strtmp,':');
if ( sp!=0 ) {
        if ( checkint(sp+1)==FALSE )
                abort_prg(5);
        gblperiodoC=atoi(sp+1);
```

```
            *(sp) = '\0';
        };
        if ( checkint(strtmp)==FALSE )
            abort_prg(5);
        gblanioC=atoi(strtmp);
        /*  Acción a tomar si no hay valor (C/M/A) */
        if (argc > 7)
            gblaccion=toupper(argv[7][0]);
        else {
            printf("Acción a tomar si no hay valor: (C)eros/(M)arcar/(A)bortar: ");
            gets(strtmp);
            gblaccion=toupper(strtmp[0]);
        };
        /*  Strings de sustitucion global (&0..&9)   */
        for (i=8; i<18; i++) {
            if (argc > i)
                strcpy(wild[i-8],argv[i]);
            else {
                printf("String de sustitución %d (\&%d): ",i-8,i-8);
                gets(wild[i-8]);
            };
        };
        /* Verifica cual es el tipo de usuario (de Consulta o Propietario) */
        gblustipo=getustipo();

        w4cursor_size(-1,-1);

        proceso();  /* Proceso de lectura y generación de hojas de cálculo */

        d4close_all();
        exit(0);
};

abre_arch()
{
        if ((f1=fopen(FIn,"rb"))==0)
            abort_prg(1);
        if ((f2=fopen(FOut,"wb"))==0)
            abort_prg(2);
        d4lock_code(2);
        if ((b_ref2=d4use("F:\\EJE\\FPUBLI\\DATOS\\FPDGRALS")) < 0)
            abort_prg(7);
        if ((i_ref2=i4open("F:\\EJE\\FPUBLI\\DATOS\\FPDGRALS")) < 0)
            abort_prg(8);
        if ((b_ref3=d4use("F:\\EJE\\FPUBLI\\DATOS\\FPUSER")) < 0)
            abort_prg(9);
        if ((i_ref3=i4open("F:\\EJE\\FPUBLI\\DATOS\\FPUSER")) < 0)
            abort_prg(10);
        if ((b_ref4=d4use("F:\\EJE\\FPUBLI\\DATOS\\FPCVDES")) < 0)
            abort_prg(14);
        if ((i_ref4=i4open("F:\\EJE\\FPUBLI\\DATOS\\FPCVDES")) < 0)
            abort_prg(15);
```

```
    if ((f1r=fopen("F:\\EJE\\FPUBLI\\DATOS\\FPVALR.DAT","rb")) == NULL)
        abort_prg(11);
    if ((f1o=fopen("F:\\EJE\\FPUBLI\\DATOS\\FPVALO.DAT","rb")) == NULL)
        abort_prg(12);
    if ((f1m=fopen("F:\\EJE\\FPUBLI\\DATOS\\FPVALM.DAT","rb")) == NULL)
        abort_prg(13);

    fread(&valordbl,sizeof(valordbl),1,f1r);
    anioinr=valordbl/10000; aniofinr=(int)(valordbl-anioinr*10000);
    anioinr-=1900; aniofinr-=1900;
    fread(&valordbl,sizeof(valordbl),1,f1o);
    anioino=valordbl/10000; aniofino=(int)(valordbl-anioino*10000);
    anioino-=1900; aniofino-=1900;
    fread(&valordbl,sizeof(valordbl),1,f1m);
    anioinm=valordbl/10000; aniofinm=(int)(valordbl-anioinm*10000);
    anioinm-=1900; aniofinm-=1900;
};

char getustipo()
{
    char *ap1,*ap2,cont[141];
    int num;

    d4select(b_ref3); i4select(i_ref3);
    num=d4seek_str(Mgbluser);
    switch(num){
        case -1:
        case  1:
        case  2:
        case  3:
            return(UCONSULTA);
    };
    ap1 = f4record();
    ap2 = cont;
    memcpy(ap2,ap1,140);
    /* Extrae para el usuario que está utilizando el sistema
       el área a la cual pertenece y por lo tanto a que series tiene acceso.
       Si el area asignada a éste es *** el usuario tiene acceso a todas las series */
    substr(Musclv,cont,1,4);
    substr(Musarea,cont,137,3);
    strcpy(gblareaini,Musarea);
    strcpy(gblareafin,Musarea);
    if ( gblareaini[0]=='*' ) {
        gblareaini[0]=' '; gblareafin[0]='~';
    };
    if ( gblareaini[1]=='*' ) {
        gblareaini[1]=' '; gblareafin[1]='~';
    };
    if ( gblareaini[2]=='*' ) {
        gblareaini[2]=' '; gblareafin[2]='~';
    };
    return(UMANTENIM);
};
```

```
proceso()
{
        int codigo=-1,lon=0;

        d4select(b_ref2);  i4select(i_ref2);
        /*  Lee del archivo patrón cada uno de los códigos de operación
            y los analiza, hasta encontrar el fin de archivo */
        do {
                fread(&codigo,sizeof(codigo),1,f1);
                fread(&lon,2,1,f1);
                fread(&datos,lon,1,f1);
                /* Si el código leído es el de una etiqueta, el contenido de la
                   celda debe ser analizado para verificar si se sustituiran valores */
                if(codigo==15) {
                        wildcards(&lon);
                        sustituir(&codigo,&lon);
                }
                /* Todos los códigos leídos en la hoja patron con la información
                   correspondiente a ellos es grabada en el archivo de salida.
                   Cuando en una celda de tipo etiqueta existen claves de series a sustituir,
                   el código es cambiado por el código correspondiente a una celda  de tipo
                   numérica sin alterar el formato, para lo cual deben grabarse todos los datos
                   correspondientes a la transformación */
                fwrite(&codigo,sizeof(codigo),1,f2);
                fwrite(&lon,2,1,f2);
                fwrite(datos resto,lon,1,f2);
        } while (codigo!=1);
}

wildcards(lon)
int     *lon;
{
        char  tmp[512],wn,*p1,*s1;
        int    l;

        /* Lo primero que debe verificarse es si existen caracteres de sustitución */
        s1=p1=datos.celda.formlab etiq.etiqueta;        l=strlen(p1);
        do {
                p1=strchr(p1,'&');
                if ( p1!=0 ) {
                        wn= (char) *(p1+1);
                        if ( (wn>='0') && (wn<='9') ) {
                                strcpy(tmp,p1+2);
                                *(p1)='\0';
                                strcat(s1,wild[wn-48]);
                                strcat(s1,tmp);
                                l=l-2+strlen(wild[wn-48]);
                        } else
                                p1++;
                };
        } while (p1!=0 );
        *lon=l+7; /* formato + coordenadas + align + string + nulo */
};
```

```
sustituir(codi,loni)
int *loni;
int *codi;
{
        char    *p1,*p2,tmp[1024],stmp[512],sval[512];
        int     codt;

        strcpy(tmp,datos.celda.formlab.etiq.etiqueta);
        /* Por default se elige el código para una celda de tipo fórmula */
        codt=16;
        bandera=FALSE;
        do {
                p1=strchr(tmp,'[');
                if ( p1!=0 ) {
                        p2=strchr(p1,']');
                        if ( p2!=0 ) {
                                strncpy(sval,p1+1,p2-p1-1); sval[p2-p1-1]='\0';
                                /* Se analiza el contenido de la celda extraído de entre corchetes.
                                   La función sust_valor() devuelve el código que corresponde a la celda
                                   de acuerdo al tipo de información sustituida */
                                codt=sust_valor(sval);
                                *codi=codt;
                                strcpy(stmp,p2+1);      *p1='\0';
                                strcat((tmp,sval);      strcat(tmp,stmp);
                        };
                };
        } while (( p1!=0 ) && ( p2!=0));

        if (bandera==TRUE)      *codi=15;
        if (*codi==15){
                *loni=strlen(tmp)+7;
                strcpy(datos.celda.formlab.etiq.etiqueta,tmp);
        } else {
                datos.celda.formlab.formula.val=0.0;
                /* Las fórmulas almacenadas en la celda deben expresarse en notación polaca.
                   La función pol() lleva a cabo esa tarea */
                *loni=pol(tmp)+15;
        };
};

sust_valor(sv)
char    sv[];
{
        char stmp[512], *get_variable(),*get_dat_serie(),*get_prompt();

        strcpy(stmp,sv);
        switch ( sv[0] ) {
                case '_': strcpy(sv,get_prompt(&stmp[1]));
                        /* se trata de una variable de tipo prompt */
                        break;
                case '&': strcpy(sv,get_variable(&stmp[1]));
                        /* se trata de la sustitución de una de las varibles predefinidas */
                        break;
```

```
            case '!': strcpy(sv,get_dat_serie(&stmp[1]));
                    /* se trata de la sustitución de un dato de la serie */
                    break;
            default: if ( get_valor(sv) == TRUE )
                        return(16);     /* se trata de la sustitución de un valor de la serie */
                    if ( gblaccion=='A' )
                        abort_prg(6);
        };
        return(15);
};

char *get_prompt(svar)
char svar[];
{
        char sv1[67],sv2[67],sv3[67],sv4[67],svt[270], int i, w_ref1;

        /* Se define una ventana para la captura del enunciado
           que sustituira al que se encuentra actualmente */
        w_ref1 = w4define(9, 6, 16, 74), w4border(SINGLE, F_WHITE);
        w4popup(),
        w4title(0,-1, svar, B_WHITE), w4activate(w_ref1),
        w4select(w_ref1),
        for (i=0,i<65,i++)
                sv1[i] = sv2[i] = sv3[i] = sv4[i]=' ';
        sv1[0] = sv1[i] = sv2[i] = sv3[i] = sv4[i]='\0';
         g4(1, 1, sv1);  g4(2, 1, sv2),
        g4(3, 1, sv3);  g4(4, 1, sv4),
        g4read();       /* Se lee el enunciado */
        strcat(sv1,sv1);        strcat(sv1,sv2);
        strcat(sv1,sv3);        strcat(sv1,sv4),
        espacios(sv1),

        w4deactivate(w_ref1);       w4close(w_ref1);

        /* Se devuelve el enunciado leído para que sustituya al primero */
        return(sv1);
}

char *get_dat_serie(svar)
char svar[];
{
        char  *p1,stmp[512],stmp1[255],stmp2[255],

        strcpy(stmp,svar);
        p1=strchr(stmp,'.'),
        if ( p1!=0 ) {
                *(p1) = '\0';
                strcpy(stmp2,p1+1);
        } else
                strcpy(stmp2,"DES"),
        strcpy(stmp1,stmp);
        strcpy(stmp,svar);
```

```
        /* Se obtiene el dato que se pide de la serie indicado a continuación
           del punto y el cual se almacena en stmp2 para enviarlo como parámetro
           a la función que devuelve dicho dato */
        if ( dat_ser(svar,stmp,stmp1,stmp2) != FALSE )
                return(stmp);
        strcpy(stmp,"{");       strcat(stmp,svar);      strcat(stmp,"}");
        return(stmp);
};

int dat_ser(svar,st,st1,st2)
char  svar[],st[],st1[],st2[];
{
        char auxst1[255],auxst2[255],*get_periodicidad();
        int  band;

        strcpy(auxst1,st1);             strcpy(auxst2,st2);
        strupp(auxst1);                 strupp(auxst2);
        strcpy(Mvalclv,auxst1);
        if (strlen(Mvalclv) > 5) {
                if(leeserie(Mvalclv)!=0) {
                        if ( strcmp("DES",auxst2)==0 )    strcpy(st,Mdgdes);
                        if ( strcmp("UNI",auxst2)==0 )    strcpy(st,Mdguni);
                        if ( strcmp("FUE",auxst2)==0 )    strcpy(st,Mdgfue);
                        if ( strcmp("TIP",auxst2)==0 )    get_tipo(Mdgtip[0],st);
                        if ( strcmp("PER",auxst2)==0 )    strcpy(st,get_periodicidad(Mdgper[0]));
                        if ( strcmp("AÑOI",auxst2)==0 )   sprintf(st,"%d",Mdgani);
                        if ( strcmp("PERI",auxst2)==0 )   sprintf(st,"%d",Mdgpei);
                        if ( strcmp("RES",auxst2)==0 )    get_usdat(Mdgres,'R',st);
                        if ( strcmp("AREA",auxst2)==0 )   get_area(Mdgarea,st);
                        if ( strcmp("UBIC",auxst2)==0 )   get_usdat(Mdgres,'U',st);
                }
        } else      get_concep(Mvalclv,st);
        if ( (band=strcmp(st,svar))!=FALSE )
                VarCase(st,st2);
        return(band);
};

get_concep(cvser,st)
char cvser[];
char st[];
{
        char *ap1,*ap2,cont[70];
        int num;

        d4select(b_ref4);  i4select(i_ref4);
        num=d4seek_str(cvser);
        switch(num) {
                case -1:
                case  2:
                case  3:
                        d4select(b_ref2);  i4select(i_ref2);
                        strcpy(st,"NO REGISTRADA"); return(0);
        };
```

```
        ap1 = f4record();     ap2 = cont;
        memcpy(ap2,ap1,69);
        substr(st,cont,9,60);
        d4select(b_ref2);  i4select(i_ref2);
        return(1);
};

get_usdat(cveres,td,st)
char cveres[];
char td;
char st[];
{
        char *ap1,*ap2,cont[141];
        int num;

        d4select(b_ref3);  i4select(i_ref3);
        num=d4seek_str(cveres);
        switch(num) {
            case -1:
            case  1:
            case  2:
            case  3:
                    d4select(b_ref2); i4select(i_ref2);
                    if (td=='R') {
                            strcpy(st,"NO REGISTRADO"); return(0);
                    } else {
                            strcpy(st,"NO REGISTRADA"); return(0);
                    }
        };
        ap1 = f4record();     ap2 = cont;
        memcpy(ap2,ap1,140);
        d4select(b_ref2);  i4select(i_ref2);
        if (td=='P')    substr(st,cont,5,60);
        else            substr(st,cont,77,60);
        return(1);
};

get_area(cvearea,vec_area)
char cvearea[];
char vec_area[];
{
        int i;
        char clave[3];

        vec_area[0]='\0';
        for(i = 0; i < 10; i++) {
                substr(clave,areas[i],0,2);
                if( strcmp(clave,cvearea) == 0)
                        break;
        }
        if (i < 9)   substr(vec_area,areas[i],3,52);
        else         substr(vec_area,areas[9],3,52); return(1);
}
```

```
char *get_variable(svar)
char svar[];
{
        struct tm *fechorn;
        char  stmp[512],
        strcpy(stmp,svar),
        fechora = localtime(&segundos_ini);

        /* Obtiene fecha, hora, dias, meses, años, modos de emisión (acumulado o desacumulado)
           o rangos de la fecha de emisión de acuerdo a lo indicado en el caudro patrón */
        if ( fechas(svar,stmp,fechora) !=FALSE ) return(stmp);
        if ( horas(svar,stmp,fechora)  !=FALSE ) return(stmp);
        if ( dias(svar,stmp) !=FALSE )           return(stmp);
        if ( meses(svar,stmp,fechora)  !=FALSE ) return(stmp);
        if ( anios(svar,stmp,fechora)  !=FALSE ) return(stmp);
        if ( modos(svar,stmp)      !=FALSE )     return(stmp);
        if ( emision(svar,stmp)    !=FALSE )     return(stmp);

        strcpy(stmp,"(&"),    streat(stmp,svar),    streat(stmp,")"),
        return(stmp),
};

int get_valor(sv)
char        sv[];
{
        char  tmp[512],tk1[20],tk2[20],tk3[20],stmp[512],*gettoken(),tipo;
        int cp,tmpinc,i1,i2,i3,valanio,valperiodo,cmperiodo(),inceriodo(),lectura;
        unsigned char valmodo;

        strcpy(tmp,sv);
        /* Extrae la clave de la serie contenida en la sintaxis
           del tipo  ( AAA01010.M.99.99 ) */
        strcpy(Mvalclv,gettoken(&tmp[0],' '));
        lectura=TRUE;
        tipo=Mvalclv[7];
        /* Lee los datos generales de la serie, asi como el apuntador al conjunto de valores
           para el tipo de información indicada por el último caracter de la clave de la serie
           contenida en Mvalclv */
        if(leeserie(Mvalclv)==0)
                lectura=FALSE;
        i1 = strlen(Mvalclv)+1;
        if ( strlen(Mvalclv)==strlen(tmp) ) {
                /* Si no se definen el modo en que se requiere la información,
                   el año y el periodo, se toman los dado como parámetros de entrada */
                valmodo=gblmodo;
                valanio=gblanio,
                valperiodo=gblperiodo,
        } else {
                /* De otro modo se extrae el modo en que se requiere la información;
                 si se indica un '@', entonces se considera el modo dado como parámetro */
                strcpy(tk1,gettoken(&tmp[i1],' '));
                i2 = strlen(tk1)+1+i1;
```

```
    if ( tk1[0]=='@' )
        valmodo=gblmodo;
    else
        valmodo=tk1[0];
    /*  Se extraen el año y periodo para los cuales se requiere la información.
        En estos pueden incluirse incrementos o decrementos que en
        caso de existir son calculados */
    if ( i2 > strlen(tmp) ) {
        valanio=gblanio;
        valperiodo=gblperiodo;
    } else {
        strcpy(tk2,gettoken(&tmp[i2],':'));
        i3 = strlen(tk2)+1+i2;
        if ( i3 > strlen(tmp) ) {
            valanio=gblanio;
            valperiodo=gblperiodo;
            if ( checkint(tk2)==FALSE ) {
                sprintf(sv,"{%s}",tmp);
                return(CheckCeros(sv));
            }
            tmpinc=atoi(tk2);
            incperiodo(&valanio,&valperiodo,tmpinc,Mdgper[0]);
        } else {
            if ( tk2[0]=='i' )
                valanio=gblanio;
            else {
                if ( checkint(tk2)==FALSE ) {
                    sprintf(sv,"{%s}",tmp);
                    return(CheckCeros(sv));
                }
                if ( (tk2[0]=='+')  || (tk2[0]=='-') )
                    valanio = gblanio+atoi(tk2);
                else
                    valanio  = atoi(tk2);
                if ( checkint(&tmp[i3])==FALSE ) {
                    sprintf(sv,"{%s}",tmp);
                    return(CheckCeros(sv));
                }
                if ( (tmp[i3]=='+') || (tmp[i3]=='-') ) {
                    valperiodo=gblperiodo;
                    incperiodo(&valanio,&valperiodo,atoi(&tmp[i3]),Mdgper[0]);
                } else
                    valperiodo = atoi(&tmp[i3]);
            }
        }
    }
}
sprintf(sv,"{%s,%c,%d:%d}",Mvalclv,valmodo,valanio,valperiodo);
if (lectura==FALSE)
    return(CheckCeros(sv));
if ( (cmperiodo(valanio,valperiodo,Mdgani,Mdgpei)) < 0 )
    return(CheckCeros(sv));
```

```
    if ( gblustipo==UCONSULTA ) {
            if ( (cmperiodo(valanio,valperiodo,Mdgans,Mdgps) ) > 0 )
                    return(CheckCeros(sv));
    }
    /* Se verifica si el usuario tiene acceso a toda la información de la serie */
    cp=CheckProp();
    if ( cp!=TRUE ) {
            if ( (cmperiodo(valanio,valperiodo,Mdgans,Mdgps) ) > 0 )
                    return(CheckCeros(sv));
    }
    /* Se lee el valor de la serie para el modo, año y período
    indicados y el valor leido se devuelve en sv */
    if ( leevalor(valanio,valperiodo,valmodo,tipo,sv)!=TRUE )
            return(CheckCeros(sv));
    else
            return(TRUE);
}

leeserie(Mval)
char Mval[9];
{
    char  valores[170],*ap1,*ap2,Mclv[9];
    char  Mdgani_v[3], Mdgpei_v[4], Mdganf_v[3],Mdgpef_v[4],
          Mdgans_v[3], Mdgps_v[4],regini_v[9];
    int   num;

    strcpy(Mclv,Mval); Mclv[7]='\0';
    num=d4seek_str(Mclv);
    switch(num){
            case -1:
            case  1:
            case  2:
            case  3:
                    return(0);
    };
    ap1 = f4record();     ap2 = valores;
    memcpy(ap2,ap1,168);
    substr(Mdgdes,      valores,8,60);
    substr(Mdgtni,      valores,68,25);
     substr(Mdgper,     valores,93,1);
    substr(Mdgtip,      valores,94,1);
    substr(Mdgani_v,    valores,95,2);
     substr(Mdgpei_v,   valores,97,3);
    substr(Mdganf_v,    valores,100,2);
    substr(Mdgpef_v,    valores,102,3);
    substr(Mdgans_v,    valores,105,2);
    substr(Mdgps_v,     valores,107,3);
    substr(Mdgfue,      valores,110,20);
    substr(Mdgres,      valores,130,4);
    substr(Mdgarea,     valores,143,2);
    switch(Mval[7]) {
            case '0': substr(regini_v,valores,145,8); break;
            case 'A': substr(regini_v,valores,153,8); break;
```

```
            case 'M': substr(regini_v,valores,161,7),
        };
        regini=atol(regini_v),
        if (regini==0)
            return(0);
        Mdgani=atoi(Mdgani_v),   Mdgpei=atoi(Mdgpei_v),
        Mdganf=atoi(Mdganf_v),   Mdgpef=atoi(Mdgpef_v);
        Mdgans=atoi(Mdgans_v);   Mdgps=atoi(Mdgps_v);
        return(1),
};

int CheckProp()
{
        if ( strcmp(Mdgarea,gblareaini) < 0 )
            return(FALSE);
        if ( strcmp(Mdgarea,gblareafin) > 0 )
            return(FALSE);
        return(TRUE),
};

int CheckCeros(sv)
char    sv[];
{
        char cad[250];
        if ( gblaccion=='C' ) {
            strcpy(sv,"0.0");
            return(TRUE);
        } else {
            bandera=TRUE;
            return(FALSE);
        }
};

int leevalor(anio,per,modo,tp,sv)
int anio,per;
unsigned char tp,modo;
char sv[];
{
        int  num,i,dif,existe=0,
        char stmp[20];

        Mvalor=0L;
        switch(tp) {
        case '0': if((anio < anioinr) || (anio > aniofinr))
                    return(FALSE),
                dif = anio - anioinr;   num = 0;
                for (i=0,i<dif;i++)
                    num += maxobs(anioinr+i,Mdgper[0]);
                if (modo=='D') {
                   if (fseek(f3r,(long) ((sizeof(valordbl))*(regini+(long)num+(long)(per-1))), SEEK_SET))
                        return(FALSE),
                   fread(&Mvalor,sizeof(valordbl),1,f3r),
```

```
            if(esvalor(Mvalor)==FALSE)
                return(FALSE);
            existe=1;
        } else {
            if (fseek(f3r,(long) ((sizeof(valordbl))*(regini+(long)num)), SEEK_SET))
                return(FALSE);
            while (per>0) {
                fread(&valordbl,sizeof(valordbl),1,f3r);
                if(esvalor(valordbl)==TRUE) {
                    Mvalor+=valordbl;
                    existe=1;
                }
                per--;
            };
        };
        break;
case 'A': if((anio < anioino) || (anio > aniofino))
            return(FALSE);
        dif = anio - anioino; num = 0;
        for (i=0;i<dif;i++)
            num += maxobs(anioino+i,Mdgper[0]);
        if (modo=='D') {
            if (fseek(f3o,(long) ((sizeof(valordbl))*(regini+(long)num+(long)(per-1))), SEEK_SET))
                return(FALSE);
            fread(&Mvalor,sizeof(valordbl),1,f3o);
            if(esvalor(Mvalor)==FALSE)
                return(FALSE);
            existe=1;
        } else {
            if (fseek(f3o,(long) ((sizeof(valordbl))*(regini+(long)num)), SEEK_SET))
                return(FALSE);
            while (per>0) {
                fread(&valordbl,sizeof(valordbl),1,f3o);
                if(esvalor(valordbl)==TRUE) {
                    Mvalor+=valordbl;
                    existe=1;
                }
                per--;
            }
        };
        break;
case 'M': if((anio < anioinm) || (anio > aniofinm))
            return(FALSE);
        dif = anio - anioinm; num = 0;
        for (i=0;i<dif;i++)
            num += maxobs(anioinm+i,Mdgper[0]);
        if (modo=='D') {
            if (fseek(f3m,(long) ((sizeof(valordbl))*(regini+(long)num+(long)(per-1))), SEEK_SET))
                return(FALSE);
            fread(&Mvalor,sizeof(valordbl),1,f3m);
            if(esvalor(Mvalor)==FALSE)
                return(FALSE);
            existe=1;
```

```
            } else {
                if (fseek(f3m,(long) ((sizeof(valordbl))*(regini+(long)num)), SEEK_SET))
                    return(FALSE);
                while (per>0) {
                    fread(&valordbl,sizeof(valordbl),1,f3m);
                    if(esvalor(valordbl)==TRUE) {
                        Mvalor+=valordbl;
                        existe=1;
                    }
                    per--;
                };
            };
        };
        if (existe==0)
            return(FALSE);
        strcpy(sv,gcvt(Mvalor,15,stmp));
        return(TRUE);
}

int esvalor(dob)
double dob;
{
        union u {
            double doble;
            char   cadena[8];
        };
        union u uno, dos;
        char c=-1;
        int  band=0,i;

        setmem(&uno,doble, 8, c);
        dos.doble=dob;
        for (i=0;i<8;i++)
            if(uno.cadena[i] != dos.cadena[i]) {
                band=1;
                break;
            };
        return(band);
}

int pol(cad)
char cad[];
{
        char cad2[500];
        char *ap1,*ap2,sval[20];
        unsigned char codig;

        if (polaca(cad,cad2) == 1) {
            w4display(" ERROR EN LA EXPRESION", (char *) 0);
            return(0);
        } else {
            contglb=15;
            ap1=cad2;
```

```
            do{
                    ap2=strchr(ap1,',');
                    if (ap2 !=0) {
                            strncpy(sval,ap1,ap2-ap1);
                            sval[ap2-ap1]='\0';
                            analiza(sval);
                            ap1=ap2+1;
                    }
            } while ((ap2=strchr(ap1,',')) !=0);
            codig=3;
            memcpy(&datos.resto[contglb],&codig,1);
            datos.celda.formlab.formula.longform = contglb-14;
            return(datos.celda.formlab.formula.longform);
    }
}

analiza(sv)
char sv[];
{
    double num;
    unsigned char cod;
    int var1,var2,ren,col,ren_al,col_al;
    struct byte {
            unsigned a:1;
            unsigned b:1;
            unsigned c:1;
            unsigned d:1;
            unsigned e:1;
            unsigned f:1;
            unsigned g:1;
            unsigned h:1;
    };
    union bits {
            char ch;
            struct byte bit;
    } b;
    union u {
            unsigned int entero;
            char cadena[2];
    } numero;

    if (isdigit(sv[0])) {
            num = atof(sv),   cod=0;
            memcpy(&datos.resto[contglb],&cod,1);
            memcpy(&datos.resto[contglb+1],&num,8);
            contglb+=9;
            return(0);
    };
    if (strchr("+-*/^%\\",sv[0])!=NULL) {
            if (isdigit(sv[1])) {
                    num = atof(sv),  cod=0;
                    memcpy(&datos.resto[contglb],&cod,1);
                    memcpy(&datos.resto[contglb+1],&num,8);
```

```
                contglb+=9;
                return(0);
        } else {
                switch(sv[0]) {
                        case '+' : cod = 9;  break;
                        case '-' : cod = 10; break;
                        case '*' : cod = 11; break;
                        case '/' : cod = 12; break;
                        case '^' : cod = 13;
                }
                memcpy(&datos.resto[contglb],&cod,1); contglb++;
                return(0);
        }
    }
    if(sv[0] == '(') {
        cod = 4;
        memcpy(&datos.resto[contglb],&cod,1); contglb++;
        return(0);
    };
    if(isalpha(sv[0])) {
        if (isdigit(sv[1])) {
                var2 = sv[0] - 64;
                var1 = 0;
                if ( checkint(&sv[1])==FALSE ) return(1);
                ren = atoi(&sv[1]);
        } else {
                var1 = sv[0] - 64;
                var2 = sv[1] - 64;
                if ( checkint(&sv[2])==FALSE ) return(1);
                ren = atoi(&sv[2]);
        };
        col = var1*26+var2-1;
        cod = 1; memcpy(&datos.resto[contglb],&cod,1); contglb++;
        var1 = col - datos.celda.ncol;
        var2 = ren - datos.celda.nren-1;
        numero.entero=var1;
        b.ch=numero.caden[1];
        b.bit.h=1;
        b.bit.g=0;
        b.bit.f = (var1>=0) ? 0 : 1;
        memcpy(&datos.resto[contglb],&numero.caden[0],1);
        datos.resto[contglb+1]=b.ch;  contglb+=2;
        numero.entero=var2;
        b.ch=numero.caden[1];
        b.bit.h=1;
        b.bit.g=0;
        b.bit.f = (var2>=0) ? 0 : 1;
        memcpy(&datos.resto[contglb],&numero.caden[0],1);
        datos.resto[contglb+1]=b.ch;  contglb+=2;
        return(0);
    }
}
```

```
void abort_prg(nerror)
int  nerror;
{
    switch (nerror) {
        case 1:  w4display(" El archivo no existe o está protegido ",Fln, (char *)0);
                 break;
        case 2:  w4display(" ERROR "," Espacio insuficiente en disco o directorio protegido ", (char *)0);
                 break;
        case 3:  w4display(" ERROR "," Periodo base de series inválido ",(char *)0),
                 break;
        case 4:  w4display(" ERROR "," Año base de series inválido",(char *)0);
                 break;
        case 5   w4display(" ERROR "," Periodo de emisión inválido ",(char *)0);
                 break;
        case 6:  w4display(" ERROR "," No se encontró valor. Opción de ABORTAR ",(char *)0);
                 break;
        case 7:  w4display(" El archivo no existe o está protegido "," DATOS GENERALES ",(char *)0);
                 break;
        case 8:  w4display(" El archivo no existe o está protegido "," INDICE DATOS GENERALES ",
                 (char *)0),
                 break;
        case 9:  w4display(" El archivo no existe o está protegido "," USUARIOS ",(char *)0);
                 break;
        case 10: w4display(" El archivo no existe o está protegido "," INDICE USUARIOS ",(char *)0);
                 break;
        case 11: w4display(" El archivo no existe o está protegido "," VALORES P. REAL ",(char *)0);
                 break;
        case 12: w4display(" El archivo no existe o está protegido "," VALORES P. ORIGINAL ",
                 (char *)0),
                 break;
        case 13: w4display(" El archivo no existe o está protegido "," VALORES P. MODIFICADO ",
                 (char *)0),
                 break,
        case 14: w4display(" El archivo no existe o está protegido "," SERIES CONCEPTUALES ",
                 (char *)0),
                 break,
        case 15: w4display(" El archivo no existe o está protegido ",
                 " INDICE DE SERIES CONCEPTUALES ", (char *)0);
    },
    w4deactivate(w_ref),    w4close(w_ref);
    fclose(f1);    fclose(f2),
    fclose(f3r);   fclose(f3o),    fclose(f3m),
    d4close_all(),
    w4cursor_size(0,7);
    w4exit(0);
};
```

En la figura siguiente se muestra el diagrama de programas, en donde se pueden observar todos aquellos que integran el sistema y la forma como se relacionan entre si.

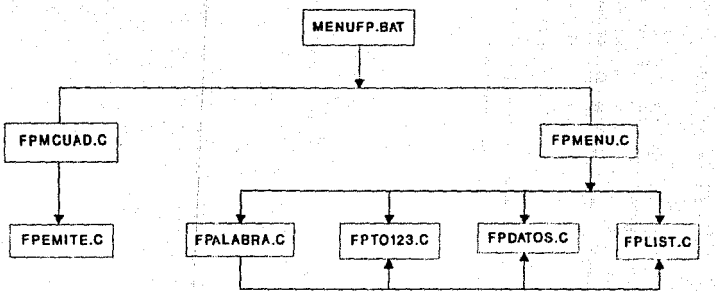


DIAGRAMA DE PROGRAMAS

CONCLUSIONES

CONCLUSIONES

Al terminar el desarrollo del Sistema, realizar pruebas y hacerlo funcionar, puedo decir que los objetivos que se habían planteado desde hace cuatro años, finalmente se cubrieron.

El Sistema satisface las necesidades de los usuarios:

- Tiene un tiempo de respuesta muy bueno.
- Comprime la información a un tamaño adecuado para su almacenamiento.
- La información que reporta es confiable, no es redundante y está descentralizada.
- Las diferentes direcciones de área tienen acceso al banco de datos, el cual es reestringido através de una clave propia.
- Permite la consulta de información de manera organizada.

El sistema se liberó formalmente en el mes de abril de este año, y se dió capacitación a los usuarios para el manejo del mismo. En el transcurso del mes los usuarios se adaptaron al sistema y comenzaron a utilizarlo. En general hubo comentarios satisfactorios y colaboración para la

puesta en marcha.

La actualización de la información, se lleva a cabo de manera trimestral, y por períodos de tiempo; cada período tarda alrededor de ocho horas efectivas. El proceso se lleva a cabo durante la noche, para no interferir con las actividades de los usuarios.

Elaboré dos manuales del Sistema :

- **Manual Técnico.** El cual contiene estructuras de las bases de datos, diagrama de bloques, diagrama de flujo de información e información técnica en general.

- **Manual de Usuario.** El cual contiene pantallas del sistema, su modo de operación y descripción de procesos generales.

Durante el desarrollo del Sistema, nos enfrentamos a diversos problemas técnicos, como el manejo de un gran volumen de información, administración de la memoria y los segmentos de datos, análisis del formato de archivos de Lotus, etc. Todos estos problemas fueron solventados, de lo cual considero que se obtuvo una gran experiencia técnica y personal.

Como todo Sistema, el SFPE es susceptible de mejorarse, quizás codificando en programación orientada a objetos, o bien agregándole nuevos módulos, como graficación, exportación a otras hojas de cálculo, desarrollo de un software de instalación, etc.Todas estas posibles mejoras se realizarían como una nueva versión del Sistema. También es cierto que el sistema tiene un ciclo de vida y que deben realizarse nuevas versiones para preservarlo.

La computación se aplica en todas las áreas, y pienso que la Facultad de Ingeniería proporciona al estudiante las herramientas necesarias para desenvolverse bien en cualquiera de ellas.

Es muy común encontrar sistemas de tipo administrativo y pienso que es un campo donde se puede aportar mucho, por lo que se necesita una buena preparación para lograrlo. Creo además que así como se logró este desarrollo, se pueden realizar sistemas de otro tipo trabajando en conjunto con especialistas del área.

Estoy satisfecha por el trabajo realizado, porque siento que se aportó algo bueno a un área importante y que era desconocida para mí. En lo personal me gusta mucho el desarrollo de software y pienso dedicarme a ello durante algún tiempo.

BIBLIOGRAFÍA

- Walden ,Jeff
"File formats for popular PC Software"
Wiley Press, U.S.A., 1986

- Schildt, Herbert
"C: manual de referencia"
Mc Graw Hill, España, 1991

- Schildt, Herbert
"Programación en lenguaje C"
Mc Graw Hill, México, 1989

- Schildt, Herbert
"C Guia para usuarios expertos"
Mc Graw Hill, España, 1989

- Kernighan, Brian; Ritchie Dennis
"El lenguaje de programación C"
Prentice Hall, México, 1986

- Ceballos, Francisco Javier
"Enciclopedia del lenguaje C"
Addison Wesley, E.U.A., 1993

- Anasagasti, Pedro de Miguel
"Fundamentos de los computadores"
Paraninfo, España, 1986

- Matthews, Martin y Carole
"Aplique 1-2-3 versión 3"
Mc Graw Hill, España, 1989

ANEXO A

En base a los requerimientos de los usuarios se creó la siguiente sintaxis.

TABLA DE SINTAXIS PARA INDICAR FECHAS

Sintaxis	Descripción	Sustitución
[&AÑOEMI1] [&AÑOEMI1+2] [&AÑOEMI1-1]	Muestra el año para el cual fue emitido el cuadro. Este se puede incrementar o decrementar.	1992 1994 1991
[&AÑOEMI2] [&AÑOEMI2+2 [&AÑOEMI2-1]	Realiza la misma acción que el anterior pero en formato abreviado.	92 94 91
[&AÑO1]	Indica el año para el cual fue emitido el cuadro.	1992
[&AÑO2]	Realiza la misma acción que el anterior, pero en formato abreviado.	92
[&NMESINI] [&NMESINI+2] [&NMESINI-1]	Indica el mes del año en que inicia el periodo de la información que se va a consultar (formato numérico). Este se puede incrementar o decrementar.	1 3 12
[&NMESFIN] [&NMESFIN+2] [&NMESFIN-1]	Indica el mes del año en que finaliza el periodo de la información a consultar (formato numérico). Este se puede incrementar o decrementar. Para un periodo trimestral	3 5 2
[&NMES] [&NMES+4] [&NMES-1]	Indica el mes del año en que fue emitido el cuadro(formato numérico). Este se puede incrementar o decrementar. Si la emisión del cuadro se lleva a cabo en el mes de julio.	7 11 6
[&mes1] [&Mes1+4] [&MES1-1]	Indica el mes en que fue emitido el cuadro (formato alfabético). Este se puede incrementar o decrementar	Julio Noviembre JUNIO

Sintaxis	Descripción	Sustitución
[&Mesini1+1]	Indica el mes en que inicia el período de emisión del cuadro (formato alfabético).	Enero
[&NDIAINI] [&NDIAINI+2] [&NDIAINI-1]	Indica el día del mes en que inicia la consulta de la información (formato numérico). Se puede incrementar o decrementar.	1 3 31
[&NDIAFIN] [&NDIAFIN+2] [&NDIAFIN-1]	Indica el día del mes en que termina el periodo de consulta de la información (formato numérico). Se puede incrementar o decrementar.	30 2 29
[&dia1] [&Dia1] [&DIA1] [&Dia1-1]	Indica el día de la semana en que fue emitido el cuadro (formato alfabético).También permite incrementos o decrementos.	viernes Viernes VIERNES Jueves
[&dia2] [&Dia2] [&DIA2] [&DIA2-1]	Realiza la misma accion que la anterior pero en formato abreviado. Se puede incrementar o decrementar.	vie Vie VIE JUE
[&NDIA] [&NDIA+1] [&NDIA-1]	Indica el día del mes en que fue emitido el cuadro (formato numérico). Se puede incrementar o decrementar.	31 1 30
[&NDIAÑO] [&NDIAÑO+10]	Indica los días transcurridos en el año hasta el día en que se emitió el cuadro. Se puede incrementar o decrementar.	213 223
[&EMISION1]	Indica el periodo para el cual fue emitido el cuadro (formato numérico). Se pueden incrementar o decrementar. Considerando una consulta de 90 periodo tres acumulado	90/01/01 - 90/03/31

Sintaxis	Descripción	Sustitución
[&EMISION2]	Indica el periodo para el cual fue emitido el cuadro (formato alfa numérico). Se pueden incrementar o decrementar.	01-Ene-1990: 31 Mar-1990

Tabla A.1

TABLA DE PARAMETROS PARA SUSTITUCION DE CAMPOS DE LA BASE DE DATOS

Sintaxis	Descripción
[!BBE0505]	Presenta la descripción de la serie indicada.
[!BBE0505.TIP]	Indica el tipo de serie que se presenta (Básica Generada Calculada o Desconocida)
[!BBE0505.Res]	Muestra el nombre del responsable de la serie.
[!BBE0505.per]	Indica el tipo de periodicidad que tiene la serie.
[!BBE0505.Ubic]	Muestra el domicilio del responsable de la serie.
[!BBE0505.FUE]	Indica la Fuente o procedencia de la información de la serie.
[!BBE0505.area]	Indica el área a la cual pertenece la información

Tabla A.2

TABLA DE SINTAXIS PARA OBTENER EL MODO DE INFORMACION

Sintaxis	Descripción	Resultados
[&MODOSM]	Describe el modo en el que se va a mostrar la información. SM para singular masculino.	ACUMULADA
[&Modosf]	sf para singular femenino	Acumulada
[&modopm]	pm para plural masculino	acumulados
[&Modopf]	pf para plural femenino	Acumuladas

Tabla A.3

Es importante señalar que si se desea que los campos a sustituir sean escritos en mayúsculas y minúsculas, solo en mayúsculas, o solo en minúsculas, es necesario indicarlo así en el formato elegido, como se mostró en las tablas anteriormente descritas.