UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

AUTOMATIZACION DE LA EXPEDICION, CONTABILIZACION Y CONTROL DE GIROS BANCARIOS.

TESIS PROFESIONAL

QUE PARA OBTENER EL TITULO DE :

INGENIERO EN COMPUTACION

P R E S E N T A N :

CISNEROS CASTAÑEDA HECTOR ERNESTO
MELESIO FUENTES ANGEL
SANTANA VIRGEN CLAUDIA GECELLE
TORRES MALDONADO HECTOR IVAN

DIRECTOR DE TESIS:
ING. ROCIO GEORGINA ROJAS MUÑOZ

MEXICO, D. F. 1993.

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

Indice

1 Introducción

1.1 Descripción General del Proyecto

El presente trabajo es la descripción de la aplicación de un área de la Ingeniería en Computación -*Ingeniería de Programación*- a una actividad específica dentro de una empresa.

En la actualidad existen diversas instituciones que ofrecen servicios bancarios tanto en el área nacional como internacional. Dichas empresas tienen la necesidad de mantener relaciones comerciales con empresas similares tanto en el ámbito nacional como en el extranjero.

Entre las diversas actividades de estas empresas, se encuentra la de expedición de giros o cheques hacia el extranjero, para satisfacer los requerimientos de sus clientes y los propios; es aquí en donde surge la necesidad de automatizar el proceso de expedición de giros, mediante la creación de un sistema que cubra las necesidades existentes en la empresa y coadyuve con mejoras al control del proceso.

1.2 Organización

El presente trabajo esta organizado en 8 capítulos, en los cuales se trata de abarcar todos los aspectos que deben considerarse e influyen en el desarrollo de un sistema de software. A continuación están los nombres de los capítulos y una breve explicación de lo que cada uno de ellos contiene.

Capítulo 1 Introducción. Es el presente capítulo.

Capítulo 2 ***Problemática Actual.*** En el se hace una descripción del procedimiento que realizan las empresas que ofrecen servicios bancarios para poder

expedir un giro que será enviado al extranjero. Se explican las cuatro etapas que se necesitan para la expedición de giros bancarios, las cuales son:

1.- Requisitación de la solicitud de expedición

2.- Elaboración del documento

3.- Trámite contable de la operación

4.- Conciliación de las operaciones

Se muestra también el formato general de un giro describiendo cada uno de los campos que lo conforman así como los controles necesarios para la seguridad de los recursos.

Para finalizar el capítulo se presenta la solución propuesta la cual considera una automatización en la expedición de giros para minimizar la intervención del personal en la elaboración y trámite de los giros. Dicha automatización se logrará mediante la realización de un sistema que permita controlar y procesar la información necesaria para la elaboración de los documentos. Las funciones que debe cumplir el sistema son las siguientes:

1.- Captura de la información para la elaboración del documento

2.- Elaboración del documento

3.- Control de las diferentes chequeras que maneja la empresa

4.- Generación de informes operativos

5.- Interfaz con el sistema contable de la empresa

6.- Interfaz con el sistema de conciliación de la empresa

Al lograr que el sistema propuesto realice dichas funciones se tendrá la automatización de la expedición de giros y controles seguros para la optimización de manera directa de los recursos.

El **Capítulo 3** ***Análisis del Sistema*** se divide en dos puntos de suma importancia para el buen desarrollo del sistema, estos puntos son *Análisis de Requerimientos* y *Estimación de Costos.*

En el *Análisis de Requerimientos* se desarrolla una especificación técnica de los requisitos del producto, mostrando el panorama general del producto y los ambientes de desarrollo y producción del sistema. Se describen las interfaces, flujos de datos, niveles de seguridad, opciones de los diferentes menús del sistema así como los datos utilizados en cada una de ellas y sus descripciones (diccionario de datos), se tienen

ejemplos de los diferentes reportes que proporciona el sistema y por último se tienen los requisitos de operación (tiempos de respuesta, tiempos de procesamiento, número de procesos ejecutados, restricciones de memoria y restricciones de seguridad) y manejo de excepciones.

El propósito de la *Estimación de Costos* es calcular los costos anticipados, no solo los de la construcción o desarrollo del sistema, sino también los de instalación, operación y mantenimiento. En esta sección se mencionan y describen los principales factores que influyen en el desarrollo de un sistema y en particular los que influyeron en el sistema de expedición de giros. Se presentan las dos técnicas más usadas en la práctica para poder estimar los costos del software: juicio experto y la técnica DELFI. Para finalizar se presenta un análisis de los beneficios tácticos y estratégicos que se tendrán con el producto así como la estimación de los costos de mantenimiento.

En el **Capítulo 4** ***Diseño Estructurado del Sistema*** se presentarán los módulos y criterios de modularización, para lo cual se describen las características, ventajas y desventajas del los diseños bottom-up, top-down y del desarrollo modular.

Dentro del tema de criterios de modularización se presentan diversos métodos para lograr la creación de los módulos que integrarán a un sistema haciendo énfasis en los criterios de acoplamiento y cohesión, los cuales fueron utilizados para nuestro sistema.

En el subtema de *Selección de la Plataforma de Desarrollo de Software y Hardware* se dan los criterios de selección que se utilizan en la práctica y los factores que influyeron en la selección de la plataforma para este proyecto.

El **Capítulo 5** ***Desarrollo del Sistema*** se divide en tres secciones importantes, la primera es la de estándares de programación, en la cual se hace mención de la importancia de generar código uniforme en un grupo de trabajo, para lo cual se mencionan diferentes aspectos que deben considerarse y tratar de adoptarlos para poder conseguir uniformidad en el código. La segunda parte consiste en la verificación, validación y pruebas del sistema; en está se hace mención a diferentes métodos y criterios para llevar acabo las pruebas y verificación del sistema.

El **Capítulo 6** ***Implantación y Mantenimiento del Sistema***, describe la estrategia de implantación del sistema en el ambiente de producción, considerando los planes y estrategias seguidas para la conversión de archivos, instalación del sistema y capacitación al usuario, la cual contempla el manual del usuario que sirve como referencia para la capacitación; así como el plan de mantenimiento en el cual se cubren tanto aspectos generales a considerar como aspectos administrativos involucrados en el mismo. Al finalizar esta sección se hace una descripción de las posibles aplicaciones que puede tener este sistema en escenarios diferentes a la empresa para el cual fue diseñado.

El **Capítulo 7** ***Conclusiones***, se hace una recapitulación del contenido del trabajo y se detallan diferentes aspectos relevantes que se observaron durante las diversas etapas del desarrollo del sistema, así como una evaluación de los logros obtenidos con el presente sistema y sus repercusiones en la empresa.

En el **Capítulo 8** ***Bibliografía***, se muestran las referencias de los textos utilizados como apoyo para documentar el presente trabajo, así como las referencias de los diferentes paquetes y lenguajes empleados para desarrollar el sistema.

Adicionalmente se presenta una sección de ***Anexos***, en el primero de los cuales se muestra el manual del usuario del sistema de expedición de giros. En el segundo anexo se muestran todos y cada uno de los programas en lenguaje de programación Pascal, que fueron generados como resultado de los requerimientos del sistema. Se listan en el siguiente orden: Programas ejecutables, Unidades de código compilables por separado con propósitos específicos, y Unidades de propósito general.

2 Problemática Actual

2.1 Antecedentes

Una institución que ofrezca servicios bancarios tiene la necesidad de mantener relaciones comerciales internacionales con empresas del mismo giro para satisfacer los requerimientos de sus clientes y las necesidades propias en el extranjero.

A estos requerimientos y necesidades se les conoce con el nombre de operaciones bancarias internacionales entre las que destacan: la realización de pagos en el extranjero, la compra-venta de divisas, el establecimiento de depósitos, el compromiso de cartas de crédito o pagarés, etc.

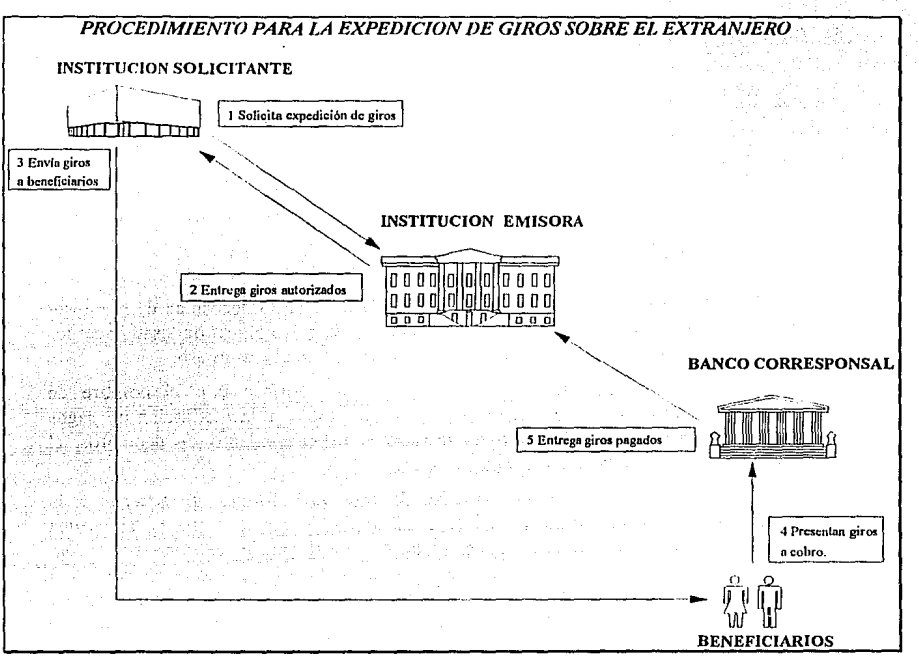
Estas operaciones pueden realizarse de diferentes formas de acuerdo a la naturaleza de la entidad con la cual se realizará la operación; esto es, si es una empresa perteneciente al ámbito financiero internacional, puede realizarse por medio de alguno de los sistemas que para este fin existen en el medio, como puede ser la red S.W.I.F.T. (Society for Worlwide Interbanking Financial Telecommunication) [SWIFT], la red pública de Telex o alguna de las redes locales de las Cámaras de Compensación (redes domésticas).

En el caso de pagos en el extranjero a una entidad beneficiaria, si esta no cuenta con ninguno de estos medios, o inclusive por acuerdo, el pago debe realizarse vía la expedición de un documento comercial (cheque o giro) en favor de la misma.

2.2 Situación Actual

En este tipo de empresas existen departamentos que tienen entre otras funciones el expedir los giros mencionados, para lo cual deben de realizar una serie de actividades encaminadas a producir el documento que será enviado al extranjero.

En la figura G.2.1 se muestra el procedimiento para la expedición de los giros hacia el extranjero y la dinámica que siguen.



G.2.1 **Procedimiento para la expedición de giros**

Como se observa en la figura G.2.2.1. la institución solicitante envía una solicitud de expedición del giro a la institución emisora, en esta se realiza la expedición vía una serie de actividades que se explican posteriormente.

Una vez elaborados los documentos, la institución emisora los entrega al solicitante, para que éstos sean distribuidos a los beneficiarios y puedan presentar el documento para su cobro en el extranjero.

Cuando el banco corresponsal de la institución emisora ha pagado el documento, lo devuelve a ésta para que el emisor realice sus operaciones de conciliación.

Las actividades que debe realizar el departamento responsable en la institución emisora pueden dividirse en cuatro etapas:

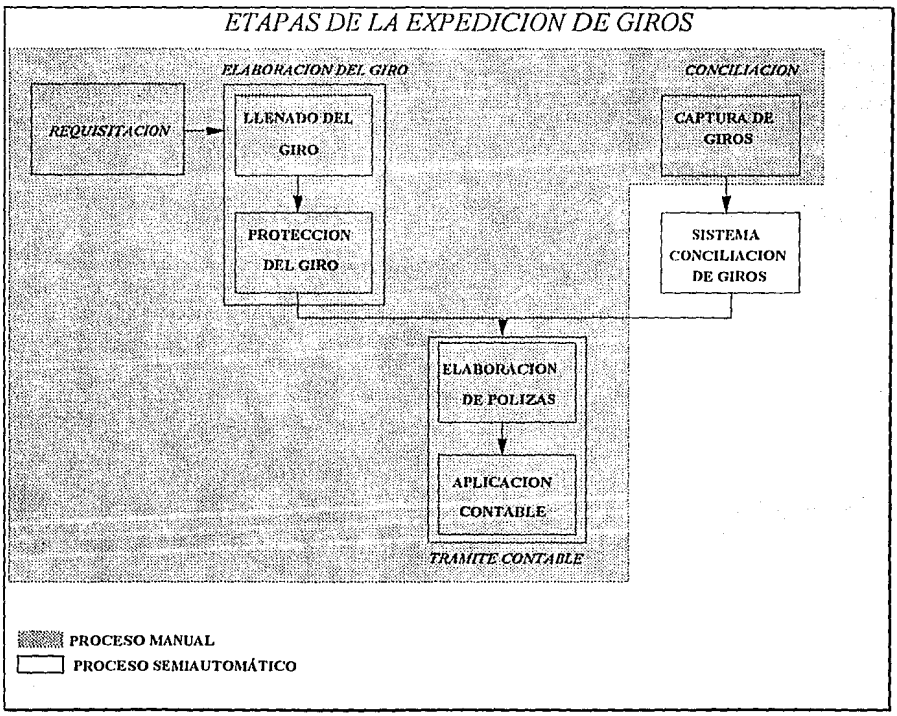
1.- Requisitación de la solicitud de expedición

2.- Elaboración del documento

3.- Trámite contable de la operación

4.- Conciliación de las operaciones

Actualmente las tres primeras fases se realizan manualmente y la cuarta se realiza en forma semiautomática; estas fases se pueden observar en la figura G.2.2, y cada una de ellas se describe a continuación.



G.2.2 Etapas de la expedición de giros.

2.2.1 Requisitación de la Solicitud de Expedición

La solicitud de expedición se realiza mediante el envío por parte de algún cliente de una carta u oficio, o en el caso de algún departamento de la misma institución vía un memorándum solicitando la expedición.

Es en esta etapa en donde se revisa que la solicitud provenga de un cliente o departamento al cual se le proporciona este servicio, y que la solicitud se encuentre debidamente autorizada, es decir, debe contener las firmas autorizadas, del personal del departamento o institución solicitante, para este efecto.

Se revisa que las solicitudes contengan los datos necesarios para poder expedir el giro, que cumplan con los requisitos de firmas autorizadas y sean entregadas con el tiempo necesario para su trámite de acuerdo a los datos que en ella se proporcionan.

2.2.2 Trámite Contable de la Operación

Una solicitud de expedición de giro se convierte en una operación más para la oficina responsable que debe ser tramitada para su debido control y asiento contable, es decir, realizar el registro de la operación en el sistema contable de la empresa mediante la elaboración de pólizas que permitan realizar cargos y abonos a las diferentes cuentas del sistema.

En esta etapa la oficina expendedora lleva a cabo la realización de pólizas y formatos de operación para pasarlos al sistema contable de la empresa y se efectúen los asientos contables pertinentes.

2.2.3 Elaboración del Documento (Giro)

El objetivo fundamental de la solicitud recibida en la institución emisora es la elaboración de un giro que será enviado al extranjero.

Para elaborar este giro, la oficina debe vaciar los datos contenidos en la solicitud hacia el giro y posteriormente pasarlo por una serie de procesos de protección que permiten evitar la falsificación o alteración del mismo.

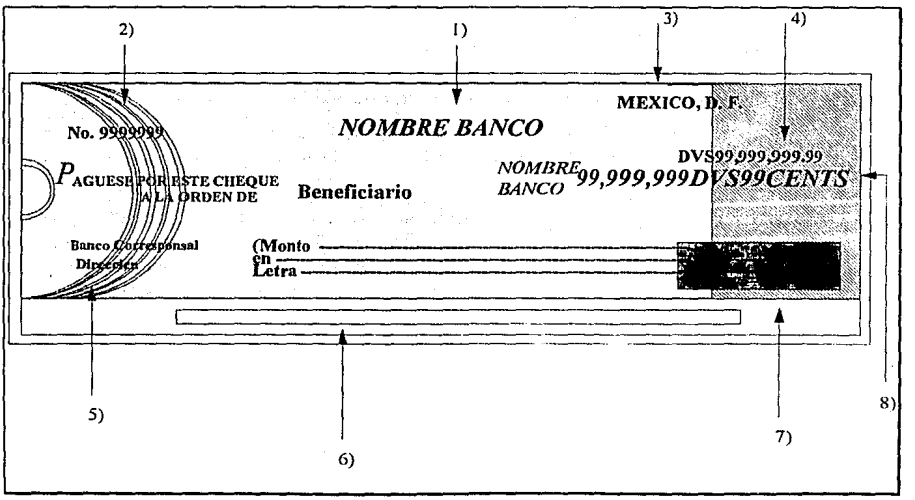
El giro debe ser llenado manualmente por el personal de la oficina, y una vez elaborado se pasa a protección en dos equipos destinados para esto, los cuales son equipos mecánicos que por su naturaleza tienen una serie de fallas, además de ser equipo antigüo y obsoleto.

Estos equipos de protección son utilizados para proteger el monto por el cual se expide el giro y una firma autorizada. La protección consiste en imprimir en el giro la firma autorizada por medio de una máquina protectora que mediante impacto imprima la firma que se encuentra grabada en un dado metálico y quede marcado físicamente el documento, de igual manera se debe de imprimir el monto en otra máquina para que quede marcado físicamente el giro y no pueda ser alterado el monto por el cual se expide.

Es necesario llevar un control de las firmas que son impresas y los montos de los giros que han sido protegidos. Estos equipos de protección son antigüos y, como ya se mecionó, tienen fallas mecánicas que provocan que el giro sea protegido erróneamente, las cifras de control deben ser modificadas y el giro debe ser elaborado una vez más por lo que un error en esta etapa implica utilizar nuevos recursos.

Los giros son formatos preimpresos en los cuales se tienen datos como son el número de documento y la institución en el extranjero que proporciona este servicio (corresponsal), por lo cual es necesario llevar un control rígido de cada uno de los formatos que son utilizados y de las chequeras correspondientes.

En la figura G.2.3 se muestra un formato general de los giros que se utilizan.



G.2.3 **Formato general de un giro**

Como se puede observar en la fig. G.2.3., el formato contiene una serie de datos los cuales algunos son preimpresos y otros son llenados por personal de la oficina expendedora, a continuación se hace una breve descripción de cada uno de ellos:

1.-Corresponde al nombre del banco emisor y se encuentra preimpreso en el giro.

2.-El número corresponde al folio del formato de cada una de las chequeras. La oficina cuenta con tres chequeras diferentes, dos de ellas son para giros en dólares, pero con diferente corresponsal y la tercera es una chequera para cualquier divisa, cualquier corresponsal.

3.-Es el lugar y fecha de emisión, en el formato se encuentra preimpreso el lugar, que siempre es México, D.F. y la fecha es llenada manualmente.

4.-Corresponde a la divisa y el monto por el cual se expide el giro. Debe de colocarse en la zona de protección derecha como lo muestra la figura. Este dato debe ser llenado manualmente.

5.-Este dato corresponde al nombre del banco corresponsal, así como la dirección y la plaza en la que se encuentra. Este dato se encuentra preimpreso en dos de las chequeras, y en los formatos de la tercer chequera es necesario llenarlo manualmente.

6.-Corresponde a una banda magnética que contiene información para el banco corresponsal y del giro en sí: cuenta en el banco corresponsal y número de giro (2). Esta banda magnética sólo la tienen las dos primeras chequeras, que son específicas para cada corresponsal mientras que la tercera no la contiene.

7.-Este elemento dentro del giro es el resultado de uno de los procesos de protección, se imprime una firma facsímil de la persona autorizada para girar sobre la chequera.

8.-Este es también resultado de la protección del giro, se imprime el monto en una forma predeterminada: el nombre del banco al inicio, la parte entera del monto, como separador decimal se usa la divisa y finalmente la parte fraccionaria y la leyenda CENTS (los giros se imprimen en idioma inglés o en casos específicos en español).

2.2.4 Conciliación de las Operaciones

Esta etapa consiste en proporcionar la información de cada uno de los giros expedidos por la oficina al departamento de conciliaciones de la institución para que se introduzca al sistema de conciliaciones, el cual se encarga de llevar un control del estado en que se encuentra el giro; un giro solamente puede tener tres estados:

- *Expedido*:
  Se considera que un giro se encuentra en estado expedido cuando ha sido elaborado y entregado a la institución solicitante.
- *Pagado*:
  Una vez que el banco corresponsal entrega los documentos, éstos han sido pagados.
- *Cancelado*:
  Cuando existe algún error en la elaboración del documento, cuando las instituciones solicitantes los devuelven, o después de un tiempo de vigencia sin ser cobrados, los documentos deben ser cancelados.

2.3 Solución Propuesta

Se pretende realizar un sistema que permita automatizar la expedición de giros, esto es, minimizar la intervención del personal en la elaboración y trámite de los giros.

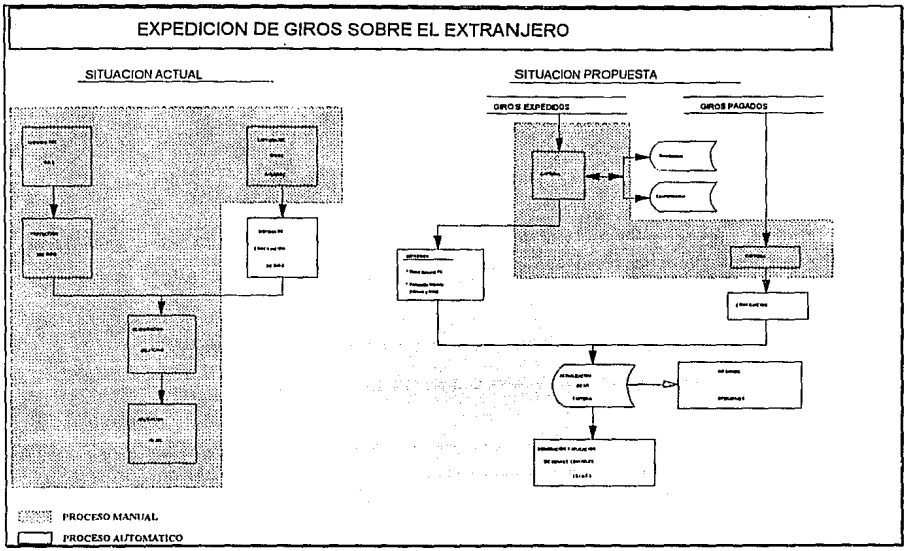
En el punto 2.2. se mencionó que existen cuatro etapas de las cuales no es posible automatizar la primera, ya que es una etapa de revisión la cual debe ser realizada por personal de la oficina encargada.

La automatización deseada puede lograrse mediante un sistema que permita controlar y procesar la información con el objetivo principal de elaborar giros, además de eliminar el uso de equipo obsoleto.

El sistema debe realizar las siguientes funciones:

- Captura de información para la elaboración del documento
- Elaboración del documento
- Control de las diferentes chequeras que maneja la empresa
- Generación de informes operativos
- Interfaz con el sistema contable de la empresa
- Interfaz para el sistema de conciliación de la empresa

La siguiente gráfica muestra un esquema comparativo general de la situación actual contra la solución propuesta.



Al tener un sistema que realice los procesos mencionados, se logra la automatización de la expedición de giros, se debe implementar un control seguro y obtener de manera directa la optimización de los recursos.

3 Análisis del Sistema

3.1 Análisis de Requerimientos

En esta sección se desarrolla una especificación técnica de los requisitos o requerimientos que debe cumplir el producto. La descripción del Sistema de Expedición y Control de Giros está basada en la propuesta de Richard Fairley para la planeación de un proyecto de programación [FAIRLEY 1988].

3.1.1 Panorama del Producto

El sistema propuesto tiene como objeto automatizar el proceso de expedición de giros. Dada la delicadeza de dicha operación, es necesario tener controles estrictos durante todo el proceso.

El sistema deberá garantizar la seguridad y el control de los recursos (formatos y firma facsímil), en base a una definición de niveles de seguridad.

3.1.2 Ambientes

Existen diferentes etapas en el ciclo de vida de un producto de software como son: análisis, desarrollo, operación y mantenimiento. De la misma forma, pueden existir diferentes ambientes para cada etapa. En el presente sistema se tiene que el ambiente de desarrollo será una red de área local y el lenguaje a utilizar será el de Turbo Pascal.

El ambiente de producción será también en una red similar a la de desarrollo. El ambiente de mantenimiento es el mismo de desarrollo, puesto que ambas tareas serán realizadas dentro de la empresa y por el mismo personal.

El porque de la selección de éstos ambientes se explicará a detalle en capítulos posteriores, pero se puede adelantar que se debe principalmente al aprovechamiento de los recursos actuales con los que cuenta la empresa.

3.1.3 Interfaces y Flujos de Datos

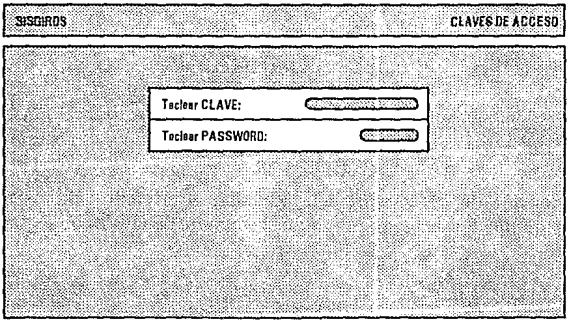
El sistema deberá ser amigable, por lo tanto debe estar basado en menús para la rápida comprensión de su funcionamiento por parte de los usuarios.

A continuación se muestran las pantallas de cada uno de los procesos, así como una breve explicación del funcionamiento de dichos procesos y el flujo de datos que se realiza en ellos.

3.1.3.1 Interfaces

Entrada al sistema

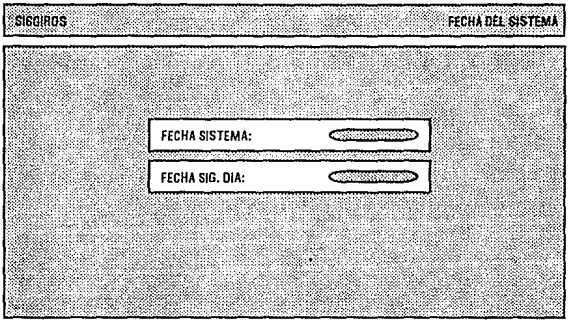
En esta etapa se debe requisitar la entrada al sistema a través de una petición de clave del usuario y con una palabra de acceso o password. La pantalla es la siguiente:



G 3.1 Pantalla de entrada al sistema.

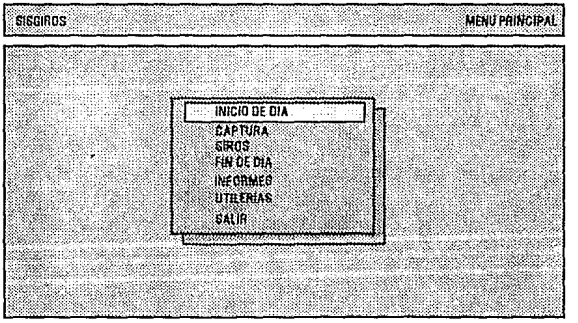
Los datos utilizados en esta etapa son USUARIOS.

La siguiente pantalla muestra la fecha del sistema, y si es la primera vez en el día que se entra al sistema, se tendrá la posibilidad de actualizar la fecha. En esta etapa se valida que la fecha registrada sea posterior a la fecha del sistema. El usuario tiene tres intentos para continuar. En esta etapa se utilizan los datos CALENDARIO y CONTROL.



G 3.2 **Pantalla de fecha del sistema.**

La siguiente pantalla es el menú principal del sistema y dependiendo del nivel del usuario podrá realizar diferentes procesos como muestra la tabla 3.1:



Proceso de inicio de día

G 3.3 **Pantalla del menú principal.**

NIVEL	PROCESOS AUTORIZADOS
Operador	Inicio de día
	Captura
	Fin de día
	Utilerías
	Informes
Supervisor	Inicio de día
	Giros
	Fin de día
	Utilerías
	Informes
Interventor	Fin de día

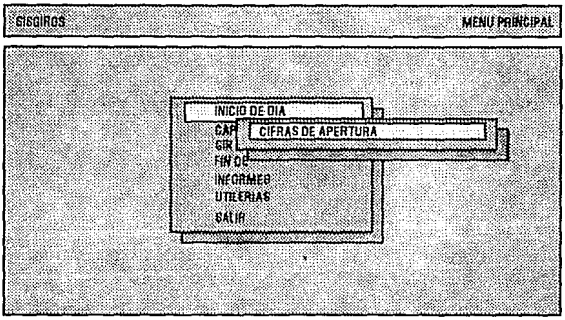
TABLA 3.1 Niveles de seguridad y procesos autorizados en el sistema.

Para navegar en este menú y los subsecuentes se utilizan las teclas de dirección y la tecla <ENTER>, o digitando directamente la tecla de la letra resaltada en la opción del menú.

Inicio de día

Como ya se mencionó anteriormente, existen diversos niveles de usuarios, los cuales podrán realizar diferente tipo de tareas, dependiendo del grupo al que pertenezcan.

La opción de inicio de día presenta un submenú con la opción de Cifras de Apertura. Este proceso debe ser lanzado siempre al inicio del día para poder continuar con los demás procesos y tiene como fin controlar los recursos con los que cuenta el sistema y asegurar la integridad de los mismos; es decir, SISGIROS es un sistema que debe tener continuidad, las cifras de cierre del día anterior deben coincidir con las cifras de apertura.



Cifras de apertura del sistema

G 3.4 **Pantalla de inicio de día.**

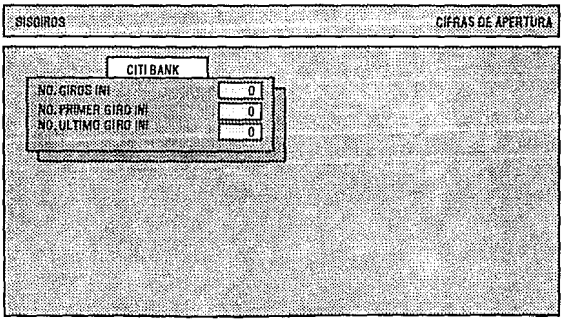
SISGIROS solicitará en el proceso de cifras de apertura tres datos para cada una de las chequeras con que cuenta. Los datos son los siguientes:

1.-Número de formatos en blanco con que cuenta la chequera

2.-El número de folio del primer formato en blanco de la chequera

3.-El número de folio del último formato en blanco de la chequera

Esta información es solicitada al usuario por medio de la siguiente pantalla:



G 3.5 **Pantalla de cifras de apertura.**

Una vez que el usuario digita la información, SISGIROS verifica que ésta sea correcta y en caso de ser errónea desplegará un mensaje de error y regresará al menú de inicio de día sin permitir que el usuario ejecute otro proceso hasta que las cifras de apertura sean las correctas.

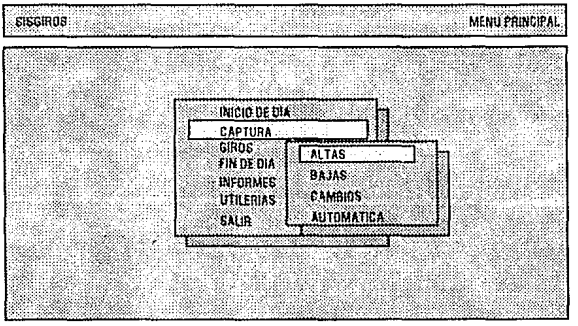
Los datos utilizados en este proceso son NUM_GIR.

Captura

Esta parte del sistema es donde el usuario alimenta las operaciones que dan origen a los giros.

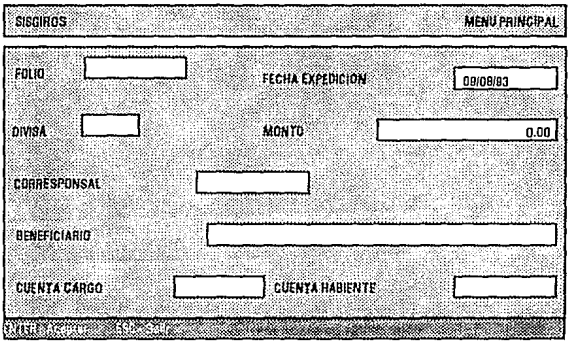
El usuario puede dar de alta nuevas operaciones, hacer modificaciones a las previamente capturadas o dar de baja operaciones.

La pantalla del menú es la siguiente:



G 3.6 Pantalla de menú de captura.

El usuario puede elegir una de las opciones del menú y pasar directamente a la pantalla que presenta la información de las operaciones. Esta pantalla es la misma para las tres primeras opciones, pero en cada una de ellas los campos que la componen tienen diferentes atributos.



G 3.6 Pantalla de captura de información.

El campo por medio del cual se accesa información previamente capturada o se introducen nuevas operaciones es el número de folio. En este campo se valida que el folio no exista, o que no haya sido procesado (dependiendo del tipo de operación que se desee realizar -altas, bajas o modificaciones-).

Cuando la opción elegida es la de altas, en cada uno de los campos se realiza una validación de la información capturada; la divisa, el corresponsal y el cuenta habiente deben estar dados de alta en los catálogos correspondientes.

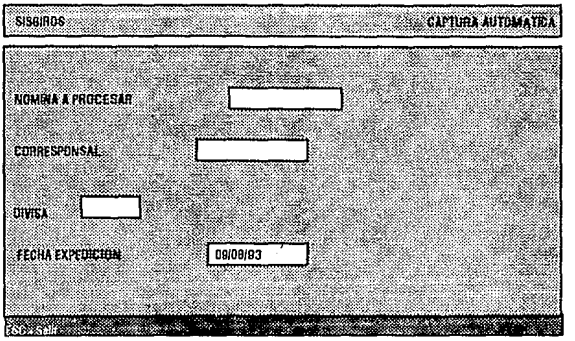
SISGIROS asigna por omisión la fecha del día de operación a la fecha del giro, ésta puede ser modificada por una fecha posterior lo cual permite que las operaciones sean capturadas en la fecha que el usuario desee. Esta fecha no podrá ser anterior a la del sistema.

Cuando se elige la opción de bajas, SISGIROS despliega la información del folio que se desea dar de baja sin permitir la modificación de los campos.

La opción de captura automática permite al usuario introducir operaciones en lote, esto es, capturar las operaciones originadas por nóminas.

Para realizar esta captura, el usuario sólo debe introducir la nómina que desea procesar, el corresponsal sobre el cual desea expedir los giros, la divisa y la fecha de expedición, con lo cual el sistema genera automáticamente los registros para cada uno de los beneficiarios de la nómina para ser procesada posteriormente.

La pantalla por medio de la cual se genera la captura automática es la siguiente:



G 3.7 **Pantalla de captura automática.**

Durante el proceso de captura los datos utilizados son BENEFGIR, CTAHABTE, CONTROL, CORRESP, DIVISA y GIROS.

Giros

Este es el proceso principal del sistema, tiene como fin la elaboración y protección de los giros que se derivan por cada operación.

Como lo indica la tabla 3.1, este proceso solamente puede ser realizado por el supervisor, el cual es el responsable de los recursos del sistema y del uso de ellos.

Cuando el supervisor elige esta opción, SISGIROS solicitará se introduzca en la unidad A: el disco llave, el cual está en posesión de una persona diferente al supervisor y no tiene acceso a SISGIROS.

Al detectar el disco llave se procede a ejecutar el proceso de impresión de giros, el cual requiere que el disco llave esté montado en la unidad para poder configurar la impresora.

Después de configurar la impresora se solicita al usuario que indique el tipo de formato que será alimentado (en forma manual ó continua).

SISGIROS realiza la impresión de los giros en tres ciclos (uno para cada chequera). Es posible generar los giros de una sola chequera presionando la tecla ESC cuando solicita que se introduzcan los formatos para las chequeras que no se desean imprimir.

En el momento en que SISGIROS imprime cada giro, verifica que existan formatos disponibles (de acuerdo con las cifras de control) para realizar la impresión, en caso de que se agoten los formatos en limpio de que

dispone, el supervisor deberá salir de esta opción y entrar a la opción de modificación de control (la cual se describe posteriormente).

Una vez que el sistema terminó de imprimir todos los giros, es necesario que el disco llave permanezca en la unidad A: y la impresora se encuentre en línea para proceder a configurarla a su estado original. Si la impresora se encuentra fuera de línea no será posible terminar con este proceso.

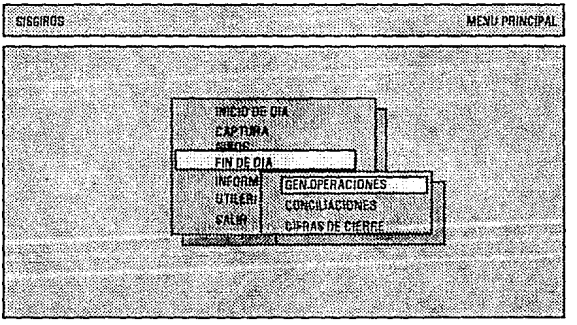
Al final de esta etapa se deberá retirar el disco llave de la unidad A:.

Los datos utilizados son CORRESP, CONTROL, DIVISA, BENEFGIR, GIROS y NUM_GIR.

Fin de día

Esta opción lleva a un submenú en el cual se encuentran los procesos que deben ser ejecutados al final de un día de operación, y una vez ejecutados el sistema no permite que se vuelvan a realizar el mismo día.

La pantalla del menú es la siguiente:



G 3.8 Pantalla de menú de fin de día.

Interfaz con el sistema contable

Esta opción ejecuta el proceso con el cual SISGIROS genera los registros para poder realizar la contabilización de las operaciones a través del sistema general contable de la empresa. Este proceso genera los registros de las operaciones del día y debido a esto, SISGIROS no permite que se vuelva a lanzar el proceso de generación de giros.

Los datos utilizados son CTAHABTE y GIROS.

Actualiza conciliaciones

Este proceso es la interfaz entre SISGIROS y el sistema de conciliaciones, con lo cual se tiene la información oportuna de la expedición de giros, sin necesidad de que la información sea capturada nuevamente.

Con esta información se alimenta el sistema de conciliaciones para actualizar las bases de datos correspondientes.

Cifras de cierre

Este es el último proceso que debe realizarse diariamente, y tiene como fin verificar que las cifras internas de sistema (las cuales son confidenciales) coincidan con los datos que la oficina responsable calcula.

La opción de cifras de cierre se encuentra habilitada sólo para el nivel de interventor, siendo este quien debe lanzar la ejecución de este proceso alimentando los datos que fueron previamente calculados de manera manual.

Los datos que se solicitan para cada una de las chequeras en este proceso son:

1.-Número de formatos utilizados

2.-Número de folio del primer formato utilizado

3.-Número de folio del último giro utilizado

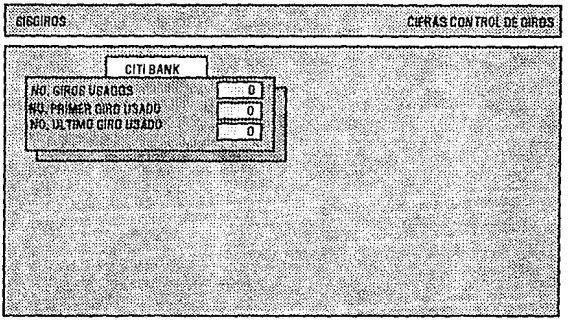
4.-Número de formatos limpios al final del día

5.-Número de folio del primer formato en limpio

6.-Número de folio del último formato en limpio

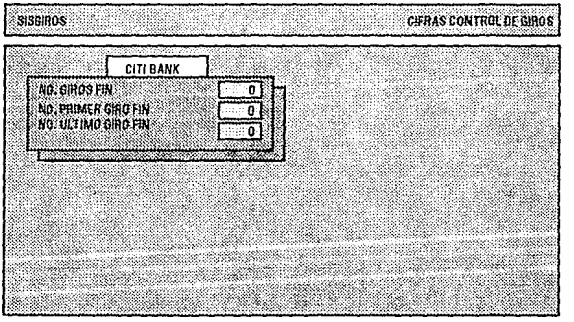
7.-Número de formatos cancelados

El sistema utiliza tres pantallas en este proceso. La primera es para los datos de los formatos utilizados y es la siguiente:



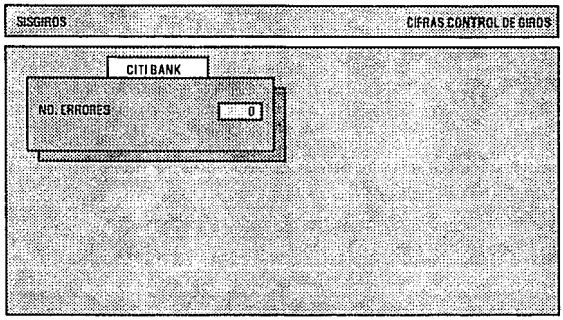
G 3.9 **Pantalla de formatos usados.**

La segunda pantalla es para los formatos en limpio al final del día:



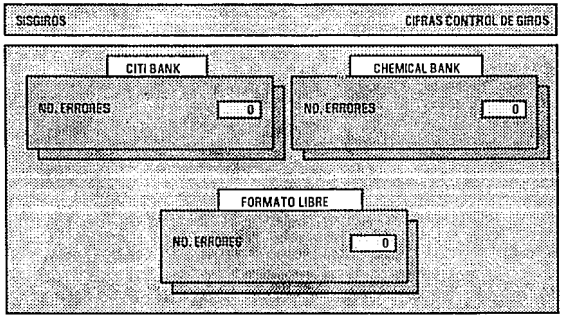
G 3.10 **Pantalla de formatos en limpio.**

Y la tercera es la pantalla de giros cancelados:



G 3.11 **Pantalla de giros cancelados.**

Estas pantallas se presentan para cada chequera, quedando finalmente la siguiente pantalla:

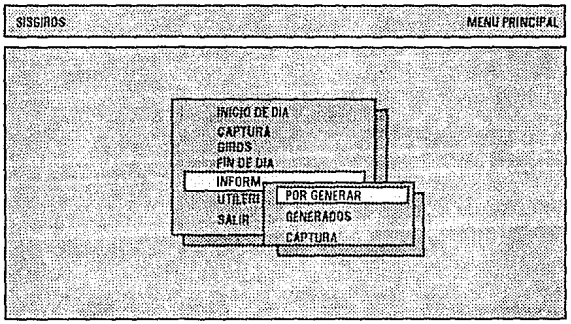


G 3.12 **Pantalla final de cifras de cierre.**

Los datos utilizados son NUM_GIR.

Informes

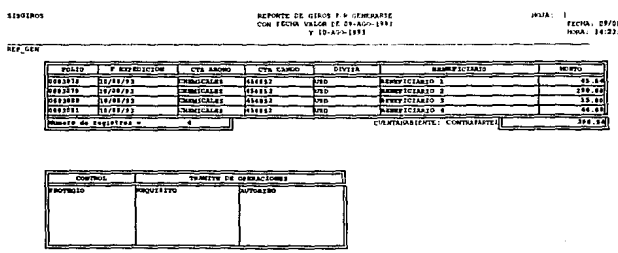
Esta opción nos lleva a un submenú de informes cuya pantalla es la siguiente:



G 3.13 **Pantalla de menú de Informes.**

Este submenú contiene los informes solicitados por la oficina responsable. Al seleccionar una de estas opciones se ejecuta el proceso para generar el informe deseado; actualmente sólo existen tres reportes: **giros por generarse, giros generados** y **capturados**.

Un ejemplo de dichos reportes se presenta en las siguientes figuras:



G 3.14 **Reporte de giros por generarse.**

SISGIROS

REPORTE DE GIROS GENERADOS
CON FECHA VALOR DE 09-AGO-1993
Y 10-AGO-1993

HOJA: 1
FECHA: 09/08/93
HORA: 15:37:05

REP_GEN

FOLIO	F EXPEDICION	NUM. DE GIRO	CTA ABONO	CTA CARGO	DIVISA	BENEFICIARIO	MONTO
[illegible]	10/08/93	[illegible]	COMERCIALES	[illegible]	USD	BENEFICIARIO 1	[illegible]
[illegible]	10/08/93	[illegible]	COMERCIALES	[illegible]	USD	BENEFICIARIO 2	[illegible]
[illegible]	10/08/93	[illegible]	COMERCIALES	[illegible]	USD	BENEFICIARIO 3	[illegible]
[illegible]	10/08/93	[illegible]	COMERCIALES	[illegible]	USD	BENEFICIARIO 4	[illegible]

Numero de registros = 4

CUENTAHABIENTE: CONTRAPARTE1 [illegible]

CONTROL	TRAMITE DE OPERACIONES	
PROTEGIO	REQUISITO	AUTORIZO

G 3.15 **Reporte de giros generados.**

SISGIROS

REPORTE DE GIROS CAPTURADOS
EL DIA 02-AGO-1993

HOJA: 1
FECHA: 02/08/93
HORA: 10:10:56

FOLIO	F EXPEDICION	CORRESPONSAL	CTA CARGO	DIVISA	BENEFICIARIO	MONTO
[illegible]	10/08/93	[illegible]	[illegible]	USD	BENEFICIARIO 10	3,360.20?
[illegible]	10/08/93	[illegible]	[illegible]	USD	BENEFICIARIO 11	[illegible]
[illegible]	10/08/93	BARCLAYLON?	[illegible]	USD	BENEFICIARIO 12	160.00?

Numero de registros = 3

CUENTAHABIENTE: CONTECH? [illegible]

G 3.16 **Reporte de giros capturados.**

Existen además otros reportes los cuales se generan al ejecutar el proceso de cifras de cierre. Los informes son el **reporte de cifras de control de giros** (que es generado en cuatro tantos: para acuse, para la oficina de control, para la de conciliaciones y para el supervisor) y el **reporte de giros con error**, de los cuales se muestran ejemplos a continuación.

SISGIROS — REPORTE DE CIFRAS DE CONTROL DE GIROS — HOJA: 1
GIROS CON FECHA VALOR 16-FEB-1993 — FECHA: 09/08/93
Y 19-FEB1993 — HORA: 13:22:15

REP_CTRL

CITI BANK

GIROS	TOTAL	NUM. PRIMER GIRO	NUM. ULTIMO GIRO
INICIO DIA	178	88118823	88118288
TRADOS	0	88888888	88888888
FIN DIA	178	88118823	88118288
NUMERO DE GIROS CON ERROR			0

CHEMICAL BANK

GIROS	TOTAL	NUM. PRIMER GIRO	NUM. ULTIMO GIRO
INICIO DIA	388	88234132	88234611
TRADOS	48	88234132	88234178
FIN DIA	336	88234154	88234888
NUMERO DE GIROS CON ERROR			0

FORMATOS LIBRES

GIROS	TOTAL	NUM. PRIMER GIRO	NUM. ULTIMO GIRO
INICIO DIA	51	88847618	88847988
TRADOS	0	88847618	88847618
FIN DIA	51	88847611	88847988
NUMERO DE GIROS CON ERROR			0

TRANSITO	INTERVENTOR

G 3.17 **Reporte de cifras de control de giros.**

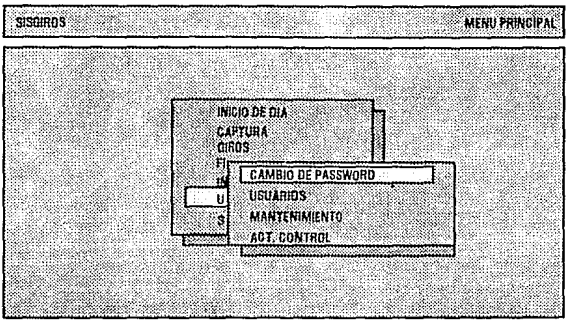
SISGIROS — REPORTE DE GIROS CON ERROR — HOJA: 1
PARA: CAJA DE VALORES — FECHA: 09/08/93
HORA: 15:32:05

FOLIO	CORRESPONSAL	DIVISA	NUMERO DE GIRO	CAUSA
88881883	CITYBK	USD	88238841	ERROR DE IMPORTEINC

G 3.18 **Reporte de giros con error.**

Utilerías

Esta opción nos lleva a un submenú que contiene las diferentes utilerías para trabajar con SISGIROS. La pantalla es la siguiente:



G 3.19 Pantalla de utilerías.

Cambio de password

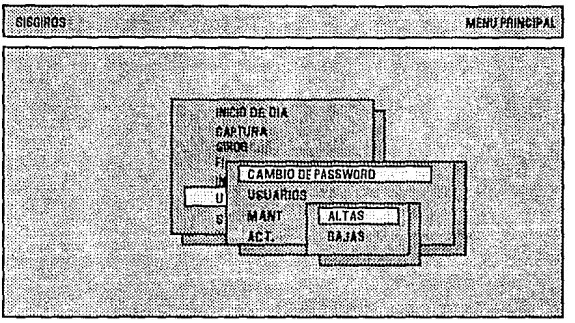
Esta opción permite al usuario modificar la palabra de acceso a SISGIRO (password), en el momento que se desee.

Para esto SISGIRO solicitará el password anterior y posteriormente el nuevo password dos veces para verificarlo.

Únicamente se puede modificar el password del usuario en sesión.

Usuarios

Al ingresar a esta opción se despliega otra pantalla con el menú siguiente:



Altas de Usuario

G 3.20 **Pantalla de utilerías.**

Alta de usuario

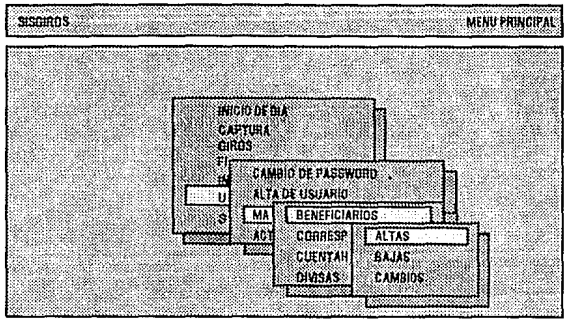
Esta opción se encuentra habilitada exclusivamente para el supervisor y tiene como fin habilitar más usuarios para trabajar con SISGIROS. Únicamente se pueden dar de alta usuarios de nivel Operador o Supervisor.

Baja de usuario

Esta opción se encuentra habilitada exclusivamente para el supervisor y tiene como fin deshabilitar a usuarios para trabajar con SISGIROS.

Mantenimiento

Esta opción permite dar mantenimiento a los diferentes archivos utilizados por SISGIROS, al seleccionarla lleva al usuario al siguiente submenú:



Alta al archivo de Beneficiarios

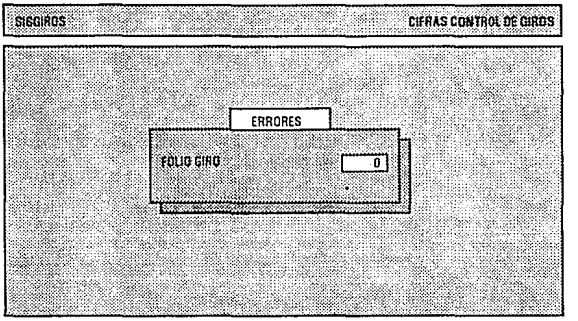
G 3.21 **Pantalla de menú de mantenimiento.**

Como se puede observar, en submenú cada opción lleva a un segundo submenú que permite al usuario indicar el tipo de movimiento que desea realizar. Este submenú se presenta en cada uno de los diferentes archivos y sólo los usuarios con nivel operador pueden accesar esta opción.

Actualiza control

Esta opción se encuentra habilitada únicamente para el supervisor. Por medio de ésta se puede realizar la cancelación de los giros que fueron expedidos y es necesario cancelarlos.

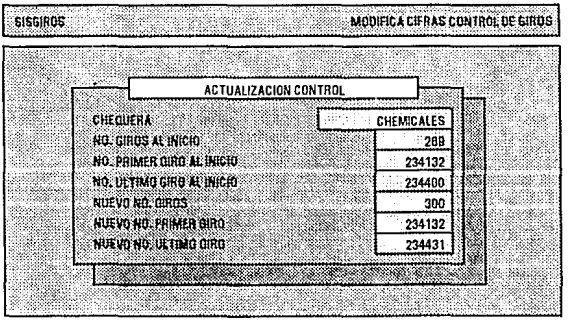
Para realizar las cancelaciones, SISGIROS solicita el folio de la operación que se desea cancelar, así como el número de folio del formato que se utilizó y el monto por el cual se expidió el giro; además del motivo por el cual se desea cancelar el giro.



G 3.22 **Pantalla de cancelaciones de giros.**

El sistema realiza las validaciones para la cancelación y en caso de que los datos que se introdujeron sean correctos, se procede a realizar la cancelación.

También se pueden modificar las cifras de control cuando se agotan los formatos que tiene la oficina responsable y se abastecen con más.

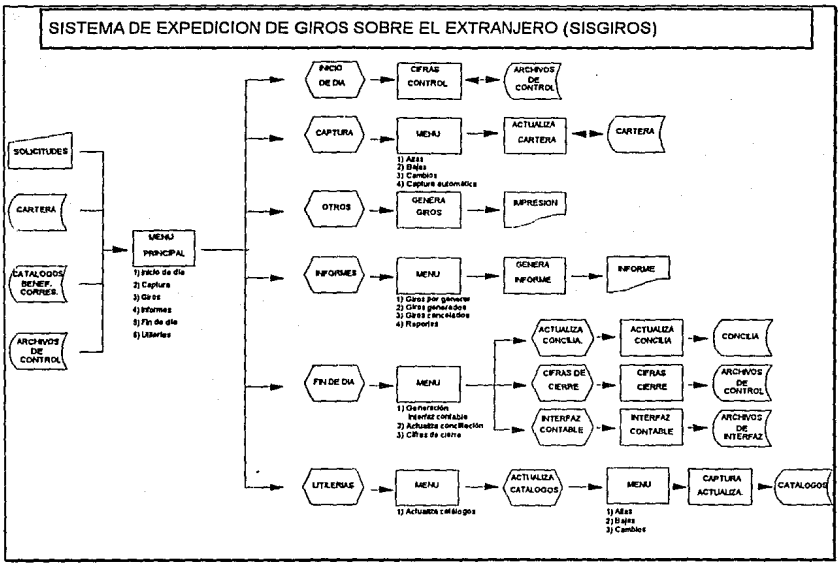


G 3.23 **Pantalla de actualización de cifras.**

3.1.3.2 Flujo de Datos

Diagrama de flujo de datos

La siguiente figura muestra el diagrama de flujo de datos general de SISGIROS.



G 3.24 **Diagrama de Flujo de Datos de SISGIROS.**

Diccionario de datos

A continuación se presenta una breve descripción de los datos utilizados en los procesos antes mencionados.

BENEFGIR Contiene los datos de los beneficiarios (nombre, monto y un campo extra que se utiliza para hacer referencia a la nómina - en capturas automáticas-), y se accesan por medio de una llave la cual es el campo clave.

CORRESP Contiene la información de los corresponsales que efectúan los pagos de los giros expedidos. Clave, nombre y campos de cuentas integran este tipo de dato.

CONTROL Se utiliza para llevar el control de todo el proceso del sistema y se compone de un identificador del proceso, estado, descripción del proceso, hora y fecha en que se realizó la última actualización al archivo de control.

DIVISA Contiene la información de las divisas utilizadas por el sistema. Los campos son clave, y nombres en español y en inglés de la divisa.

GIROS Es el dato que contiene la información de la cartera de operaciones capturadas. Los campos son folio, fecha valor, fecha de captura, divisa, corresponsal, monto, beneficiario, número de cheque, cuenta habiente, cuenta de cargo y un campo para el estado actual del giro.

NUM_GIR Contiene la información para el control de las cifras del sistema. Los campos son identificador de control, descripción, numero de control y un campo auxiliar para el número de errores.

USUARIOS Contiene la información de los usuarios del sistema. Sus campos son clave, password, grupo al que pertenece y nombre del usuario.

3.1.4 Requisitos de Operación

En esta sección especificaremos las características de operación del Sistema de Expedición y Control de Giros, estas características son:

1.- Tiempos de respuesta.

2.- Tiempos de procesamiento.

3.- Número de procesos ejecutados.

4.- Restricciones de memoria.

5.- Restricciones de seguridad.

Para las dos primeras características, tiempos de respuesta y tiempos de procesamiento, se agregó (de forma temporal) a cada proceso, imprimiera la hora en que inicia y la hora en que termina, por lo tanto los datos proporcionados son mucho muy exactos. A continuación tenemos cada una de estas características:

3.1.4.1 Tiempos de respuesta

Se refiere al tiempo que tarda el sistema en empezar a ejecutar un proceso determinado, en la tabla siguiente tenemos los tiempos de respuesta para los procesos correspondientes a cada opción del menú de SISGIROS, los tiempos están en mili segundos:

TIEMPOS DE RESPUESTA PARA LOS PROCESOS DEL SISTEMA DE EXPEDICION DE GIROS

PROCESO	TIEMPO DE RESPUESTA
Proceso de CIFRAS DE APERTURA	0.43
Proceso de CAPTURA--ALTAS	0.01
Proceso de CAPTURA--BAJAS	0.01
Proceso de CAPTURA--CAMBIOS	0.06
Proceso de CAPTURA--AUTOMATICA	0.65
Proceso de REPORTE DE CAPTURA	0.67
Proceso de GIROS	0.60
Proceso de GENERACION DE MOVIMIENTOS	0.20
Proceso de ACTUALIZA CONCILIACIONES	0.20
Proceso de INFORMES POR GENERAR	0.33
Proceso de INFORMES GENERADOS	0.33
Proceso de CAMBIO DE PASSWORD	0.02
Proceso de ALTA DE USUARIO	0.11
Proceso de BAJA DE USUARIO	0.11
Proceso de ALTA BENEFICIARIO	0.04
Proceso de BAJA BENEFICIARIO	0.03
Proceso de CAMBIO BENEFICIARIO	0.02
Proceso de REPORTE DE BENEFICIARIOS	0.28
Proceso de ALTA CTAHABTE	0.02
Proceso de BAJA CTAHABTE	0.01
Proceso de CAMBIO CTAHABTE	0.02
Proceso de REPORTE DE CTAHABIENTES	0.28
Proceso de ALTA DIVISA	0.22
Proceso de BAJA DIVISA	0.06
Proceso de REPORTE DE DIVISAS	0.27
Proceso de ALTA CORRESPONSALES	0.01
Proceso de BAJA CORRESPONSALES	0.01
Proceso de CAMBIO CORRESPONSALES	0.02
Proceso de REPORTE DE CORRESPONSALES	0.28
Proceso de CANCELACION DE GIROS	0.32
Proceso de CIFRAS CONTROL	0.25
Proceso de CIFRAS DE CIERRE	0.51

TABLA 3.2 Tiempos de respuesta

3.1.4.2 Tiempos de Procesamiento

Se refiere al tiempo que tarda el sistema en ejecutar un proceso determinado, la mayoría de los procesos de SISGIROS dependen de la cantidad de información a procesar y en ocasiones a la velocidad con que sean capturados los datos, en promedio la actualización de los archivos se encuentra entre 0.01 y 0.12 milisegundos.

3.1.4.3 Número de Procesos Ejecutados

Es el número de programas que se mandan a ejecutar para cada opción del menú:

NÚMERO DE PROCESOS EJECUTADOS

PROCESO	NUMERO Y PROCESOS EJECUTADOS
Proceso de CIFRAS DE APERTURA	1.- CIF_APER
Proceso de CAPTURA--ALTAS	1.- CAPTURA_GIROS (movimiento)
Proceso de CAPTURA--BAJAS	1.- CAPTURA_GIROS (movimiento)
Proceso de CAPTURA-CAMBIOS	1.- CAPTURA_GIROS (movimiento)
Proceso de CAPTURA--AUTOMATICA	1.- CAP_AUT
Proceso de REPORTE DE CAPTURA	1.- REP_CAP
Proceso de GIROS	1.- GIROS
Proceso de GENERACION DE MOVIMIENTOS	1.- GEN_FORM
Proceso de ACTUALIZA CONCILIACIONES	1.- ACT_CONC
Proceso de INFORMES POR GENERAR	1.- REP_GEN (N)
Proceso de INFORMES GENERADOS	1.- REP_GEN (G)
Proceso de CAMBIO DE PASSWORD	1.- CAMBIO_PASSWORD_SISGIROS
Proceso de ALTA DE USUARIO	1.- ALTA_CLAVE (grupo)
Proceso de BAJA DE USUARIO	1.- BAJA_CLAVE (grupo)
Proceso de ALTA BENEFICIARIO	1.- CAPTURA_BENEFICIARIO (movimiento)
Proceso de BAJA BENEFICIARIO	1.- CAPTURA_BENEFICIARIO (movimiento)
Proceso de CAMBIO BENEFICIARIO	1.- CAPTURA_BENEFICIARIO (movimiento)
Proceso de REPORTE DE BENEFICIARIOS	1.- REP_BEN
Proceso de ALTA CTAHABTE	1.- CAPTURA_CTAHABTE (movimiento)
Proceso de BAJA CTAHABTE	1.- CAPTURA_CTAHABTE (movimiento)
Proceso de REPORTE DE CTAHABIENTES	1.- REP_CTAH
Proceso de ALTA DIVISA	1.- CAPTURA_DIVISA (movimiento)
Proceso de BAJA DIVISA	1.- CAPTURA_DIVISA (movimiento)
Proceso de CAMBIO DIVISA	1.- CAPTURA_DIVISA (movimiento)
Proceso de REPORTE DE DIVISAS	1.- REP_DIV
Proceso de ALTA CORRESPONSALES	1.- CAPTURA_CORR (movimiento)
Proceso de BAJA CORRESPONSALES	1.- CAPTURA_CORR (movimiento)
Proceso de CAMBIO CORRESPONSALES	1.- CAPTURA_CORR (movimiento)
Proceso de REPORTE DE CORRESPONSALES	1.- REP_CORR
Proceso de CANCELACION DE GIROS	1.- MOD_CTRL
Proceso de CIFRAS CONTROL	1.- MOD_NUM
Proceso de CIFRAS DE CIERRE	1.- CIF_FIN
	2.- REP_CTRL
	3.- REP_GEN
	4.- REP_ERR

TABLA 3.4 Número y procesos ejecutados

3.1.4.5 Restricciones de Memoria

El sistema de Expedición de Giros, debido a la red en que se encontrará trabajando (sus características se mencionan en el capítulo 4), el tipo de microcomputadoras conectadas a la red y el tamaño del sistema necesita un mínimo de 512KB de memoria principal libres. En relación a la red y en general al hardware y software se presenta en el capítulo 4 el tema PLATAFORMA DE DESARROLLO, lo cual va íntimamente relacionado a las características que aquí hemos mencionado.

3.1.4.6 Restricciones de Seguridad

Para este caso se tienen varios puntos a considerar:

a) De acceso al sistema

1- Como primer nivel de seguridad se tiene un password en cada microcomputadora de la red, el cual es conocido únicamente por los empleados de la oficina respectiva.

2.-Como segundo nivel de seguridad se requiere una clave y su respectiva llave para poder ingresar a la red.

3.-El tercer nivel de seguridad se encuentra en el SISGIROS, el cual al ser invocado pide se capture la clave y su respectiva palabra llave.

b) Derechos a usuarios

1.-Como ya se mencionó en el punto 3.1.3.1 se tienen niveles de usuario dentro del SISGIROS para poder controlar las diferentes opciones que pueden realizar dentro del mismo.

c) Seguridad de recursos

1.-El sistema contiene procesos que controlan los recursos del mismo, como por ejemplo el controlar el número de formatos y la secuencia correspondiente para cada chequera.

2.-Se tiene la generación de la firma facsímil, la cual se encuentra en un disco flexible, en poder del interventor del sistema.

3.-Por último tenemos las cifras de control, tanto de inicio de día como de fin de día, que aseguran que no se alteraron los archivos del sistema durante el tiempo transcurrido entre un cierre y la apertura.

3.1.5 Excepciones

En lo que se refiere al manejo de excepciones para el SISGIROS se tienen las siguientes tablas, en las cuales se especifican los errores o fallas que se pueden presentar en el sistema. Estos errores se han dividido en dos categorías, la primera contiene las posibles fallas que ocasionan que el sistema corte su ejecución, así como su respectivo mensaje.

MENSAJES DE ERRORES O FALLAS QUE OCASIONAN QUE EL SISTEMA CORTE SU EJECUCIÓN

PROCESO	MENSAJE
-PROCESO DE CAPTURA AUTOMATICA	
" Al buscar el cuenta habiente correspondiente, si se detecta un error en la operación de lectura en el archivo btrieve.	CAP_AUT.CUENTA LEE_CUENTAHABIENTE + (cuentahabiente buscado) + el status de btrieve
" Al abrir los archivos de: BEN_GIR GIROS CTAHBTE CORRESP DIVISAS si se detecta un error en la operacion de btrieve.	CAP_AUT.ABRE_BENEF + el status de btrieve CAP_AUT.GIROS+ el status de btrieve CAP_AUT.CTAHBTE + el status de btrieve CAP_AUT-.CORRESPONSAL + el status de btrieve CAP_AUT.DIVISA + el status de btrieve
" Al tomar la fecha de captura en la variable fecha_hoy si la operación no es exitosa.	CAP_AUT.FECHA_HOY + el status de btrieve
" Al leer el archivo de GIROS para obtener el último capturado, si el resultado de la operación es diferente a fin de archivo o registro leído correctamente.	CAP_AUT.LEE1_ULTIMO_GIROS + el status de btrieve
" Detecta cuántos registros se tienen en el archivo de GIROS, antes y después de escribir los nuevos si no es exitoso.	CAP_AUT.REG_ARCH_1 status de btrieve CAP_AUT.REG_ARCH_2 status de btrieve
" Al generar número de nuevo folio, al convertir de string a numérico y no se realiza con éxito	CAP_AUT.NUM_NVO_FOLIO + (número.fol) + 999
" Al generar el nuevo registro checa que el beneficiario exista en el catálogo si la operación regresa con error.	CAP_AUT.LEE_BENEFICIARIO + status de btrieve
" Al escribir el nuevo registro a GIROS, si no es exitoso.	CAP_AUT.ESCRIBE:GIROS + status de btrieve
" Al finalizar el proceso, cierra los archivos usados, si la operación CIERRAS no se realiza con éxito.	CAP_AUT.CIERRAS + el status de btrieve.
-PROCESO DE CIFRAS DE CONTROL (APERTURA Y FIN DE DIA)	
" Al abrir el archivo NUM_GIR si no es con éxito.	CIFRAS_CONTROL: ABRE_NUM + status de btrieve
" Al crear las ventanas para la captura si no es exitoso o al desplegarlas.	ERROR EN INICIALIZA VENTANA + 8 ERROR AL DESPLEGAR VENTANA + 8
" Para checar los datos capturados convierto los valores de string a numérico, si no es exitoso.	CIFRAS_CONTROL, numeros giros + (banco) + status de btrieve
" Para checar los datos capturados de el archivo NUM_GIR si la operación no es exitosa.	CIFRAS_CONTROL: LEE1_NUM_GIR + (número) + status de btrieve
" Si los datos son correctos actualiza el archivo NUM_GIR si la operación no es exitosa.	CIFRAS_CONTROL, ACTUALIZA_NUM_GIR
" Al terminar el proceso, cierra los archivos btrieve, no es exitoso.	CIFRAS_CONTROL: CIERRAS + status de btrieve
-PROCESO DE GENERACION DE FORMATOS CONTABLES	
" Al inicializar, checa que se le pasen parámetros para la ejecución, y el parámetro se convierte de string a numérico.	NUMERO DE PARAMETROS INVALIDO + NUMERO DEPARTAMENTOS PARAMETRO INVALIDO + PARAMETRO + 999
" Abre el archivo de GIROS si no es exitoso	GEN_FORM.ABRE_GIROS + status de btrieve
" Toma la fecha de captura (fecha del día) en la x fecha_hoy.	GEN_FORM.FECHA_HOY + status de btrieve
" Al leer el archivo de GIROS, si no es exitosa.	GEN_FORM.LEE3_PRIMER_GIROS + status de btrieve GEN_FORM.LEE_SIG_GIROS
" Abre el archivo de CTAHABTE si no es exitosa.	GEN_FORM.ABRE_CTAHABTE + status de btrieve
" Al leer el archivo CTAHABTE si no es exitosa.	GEN_FORM.OBTEN_CTA_CARGO LEE_CTAHABTE +(cuentahabiente) + status btrieve
" Al cerrar los archivos.	GEN_FORM.CIERRAS + status de btrieve

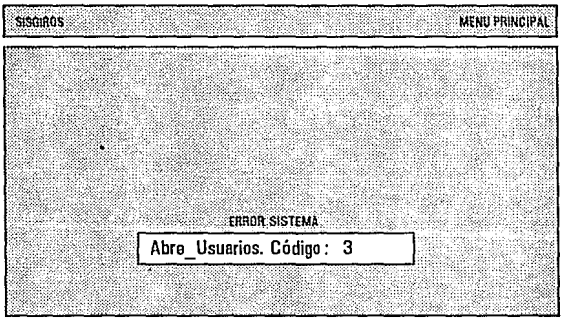
PROCESO	MENSAJE
-PROCESO DE CAPTURA AUTOMATICA	
" Al abrir el archivo GIROS.	GIROS.ABRE_GIROS + status de btrieve
" Al leer el archivo GIROS.	GIROS.LEE_GIROS + status de btrieve
" Al cerrar el archivo GIROS.	GIROS.CIERRAS + status de btrieve
-PROCESO DE MODIFICACION DE CIFRAS DE CONTROL (cancelación de giros)	
" Al inicializar la ventana si no es exitoso.	ERROR en inicializa ventana 8
" Al abrir archivo GIROS.	MOD_CTRL.ABRE_GIROS + status de btrieve
" Al abrir archivo CTRL_GIROS.	MOD_CTRL.ABRE_CTRL_GIROS + status de btrieve
" Al abrir archivo NUM_GIRO.	MOD_CTRL.ABRE_NUM_GIR + status de btrieve
" Al abrir archivo ERRORES.	MOD_CTRL.ABRE_ERRORES + status de btrieve
" Al activar ventana si no es exitosa.	MOD_CTRL.ACTIVA VENTANA 1008
" Al buscar el folio a cancelar y la operación se detecta error.	MOD_CTRL.LEE1_GE_GIROS + status de btrieve
" Al leer la fecha del sistema (fecha de captura).	MOD_CTRL.FECHA_HOY + status de btrieve
" Al actualizar la cancelación de el archivo de GIROS si no es exitoso.	MOD_CTRL.LEE1_GIROS + status de btrieve
" Al actualizar GIROS.	MOD_CTRL.ACTUALIZA1_GIROS + status de btrieve
" Al leer el control de giros.	MOD_CTRL.LEE2_GIROS + status de btrieve
" Al actualizar el control de GIROS.	MOD_CTRL.ACTUALIZA2_GIROS + status de btrieve
" Escribe al archivo de ERRORES.	MOD_CTRL.ESCRIBE_ERRORES + status de btrieve
" Al CERRAR los archivos.	MOD_CTRL.CIERRAS + status de btrieve
-PROCESO DE MODIFICACION DE NUMERO DE GIROS	
" Al inicializar la ventana de captura.	ERROR al inicializar ventana 8
" Al abrir NUM_GIR, si no es exitoso.	MOD_NUM.ABRE_NUM_GIR + status de btrieve
" Al activar la ventana.	ERROR al activar ventana 8
" Al actualizar datos, lee el archivo NUM_GIR, si no es exitoso.	MOD_NUM.LEE_NUM_GIR + status de btrieve
" Al actualizar el archivo NUM_GIR no es exitoso.	MOD_NUM.ACTUALIZA_NUM_GIR + status de btrieve
" Al cerrar archivos.	MOD_NUM.CIERRAS + status de btrieve
-PROCESO REPORTE DE BENEFICIARIOS	
" Al abrir el archivo BENEFICIARIOS.	REP_BEN.ABRE_BENEFICIARIOS + status de btrieve
" Al leer el archivo BENEFICIARIOS.	REP.BEN.LEE_BENEFICIARIOS + status de btrieve
" Al cerrar el archivo BENEFICIARIOS.	REP_BEN.CIERRAS + status de btrieve
°-PROCESO REPORTE DE CORRESPONSALES	
" Al abrir el archivo CORRESPONSALES.	REP_CORR.ABRE_CORRESP + status de btrieve
" Al leer el archivo CORRESPONSALES.	REP_CORR.LEE_CORRESP + status de btrieve
" Al cerrar el archivo CORRESPONSALES.	REP_CORR.CIERRAS + status de btrieve
-PROCESO REPORTE DE CONTROL	
" Al abrir el archivo NUMGIRO.	REP_CTRL.ABRE_NUMGIRO + status de btrieve
" Al leer la fecha del sistema.	REP_CTRL.FECHA_HOY + status de btrieve
" Al leer el archivo NUMGIRO.	REP_CTRL.LEE_NUMGIRO + status de btrieve
" Al cerrar el archivo NUMGIRO.	REP_CTRL.CIERRAS + status de btrieve
-PROCESO REPORTE DE GIROS GENERADOS	
" Al abrir el archivo GIROS.	REP_GEN.ABRE_GIROS + status de btrieve
" Al leer la fecha del sistema.	REP_GEN.FECHA_HOY + status de btrieve
" Al obtener la fecha siguiente.	REP_GEN.FECHA_SIG + status de btrieve
" Al leer el archivo GIROS.	REP_GEN.LEE_GIROS + status de btrieve
" Al cerrar el archivo GIROS.	REP_GEN.CIERRAS + status de btrieve
-PROCESO DE REPORTE DE CAPTURA	
" Al abrir el archivo GIROS.	REP_CAP.ABRE_GIROS + status de btrieve
" Al leer la fecha del sistema.	REP_CAP.FECHA_HOY + status de btrieve
" Al leer el archivo GIROS.	REP_CAP.LEE_GIROS + status de btrieve
" Al cerrar el archivo GIROS.	REP_CAP.CIERRAS + status de btrieve
-PROCESO DE REPORTE DE CTAHABIENTES	
" Al abrir el archivo CTAHABTE.	REP_CTAH.ABRE_CTAHABTE + status de btrieve
" Al leer el archivo CTAHABTE.	REP_CTAH.LEE_CTAHABTE + status de btrieve
" Al cerrar el archivo CTAHABTE.	REP_CTAH.CIERRAS + status de btrieve
-PROCESO DE REPORTE DE DIVISAS	
" Al abrir el archivo DIVISAS.	REP_DIV.ABRE_DIVISAS + status de btrieve
" Al leer la fecha del sistema.	REP_DIV.FECHA_HOY + status de btrieve
" Al leer el archivo DIVISAS.	REP_DIV.LEE_DIVISAS + status de btrieve
" Al cerrar el archivo de DIVISAS.	REP_DIV.CIERRAS + status de btrieve
-PROCESO DE REPORTE DE ERRORES	
" Al abrir el ARCHIVO ERRORES.	REP_ERR.ABRE_ERRPRES + status de btrieve
" Al leer la fecha del sistema.	REP_ERR.FECHA_HOY + status de btrieve
" Al leer el archivo errores.	REP_ERR.LEE_ERRORES + status de btrieve
" Al cerrar el archivo errores.	REP_ERR.CIERRAS + status de btrieve

PROCESO	MENSAJE
-PROCESO DE CAPTURA AUTOMATICA	
Al leer la fecha del sistema.	CAPTURA_GIROS.FECHA_HOY + status de brieve
Al escribir a GIROS.	CAPTURA_GIROS.ESCRIBE_GIROS+ status de brieve
Al actualizar de GIROS.	CAPTURA_GIROS.ACTUALIZA_GIROS + status de brieve
Al cerrar el archivo GIROS.	CAPTURA_GIROS.CIERRAS + status de brieve
-PROCESO DE CAPTURA DE BENEFICIARIOS	
Al abrir el archivo GIROS.	CAPTURA_BENEF.ABRE_BENEF + status de brieve
Al escribir a BENEF.	CAPTURA_BENEF.ESCRIBE_BENEF + status de brieve
Al borrar del BENEF.	CAPTURA_BENEF.BORRA_BENEF + status de brieve
Al actualizar el BENEF.	CAPTURA_BENEF.ACTUALIZA_BENEF + status de brieve
Al cerrar el BENEF.	CAPTURA_BENEF.CIERRAS + status de brieve
-PROCESO DE CAPTURA DE CORRESPONSALES	
Al abrir el archivo CORRESP.	CAPTURA_CORRESP.ABRE_CORRESP + status de brieve
Al escribir al archivo CORRESP.	CAPTURA_CORRESP.ESCRIBE_CORRESP + status de brieve
Al borrar del archivo CORRESP.	CAPTURA_CORRESP.BORRA_CORRESP + status de brieve
Al actualizar el archivo CORRESP.	CAPTURA_CORRESP.ACTUALIZA_CORRESP + status de brieve
Al cerrar el archivo CORRESP.	CAPTURA_CORRESP.CIERRAS + status de brieve
-PROCESO DE CAPTURA DE CTAHABTE	
Al abrir el archivo CTAHABTE.	CAPTURA_CTAHABTE.ABRE_CTAHABTE + status de brieve
Al escribir al archivo CTAHABTE.	CAPTURA_CTAHABTE.ESCRIBE_CTAHABTE + status de brieve
Al borrar del archivo CTAHABTE.	CAPTURA_CTAHABTE.BORRA_CTAHABTE + status de brieve
Al actualizar el archivo CTAHABRE.	CAPTURA_CTAHABTE-ACTUALIZA_CTAHABTE + status de brieve
Al cerrar el archivo CTAHABTE.	CAPTURA_CTAHABTE.CIERRAS + status de brieve
-PROCESO DE CAPTURA DE DIVISAS	
Al abrir el archivo de DIVISAS.	CAPTURA_DIVISAS.ABRE_DIVISAS + status de brieve
Al escribir al archivo de DIVISAS.	CAPTURA_DIVISAS.ESCRIBE_DIVISAS + status de brieve
Al borrar del archivo de DIVISAS	CAPTURA_DIVISAS.BORRA_DIVISAS + status de brieve
Al actualizar el archivo de DIVISAS.	CAPTURA_DIVISAS.ACTUALIZA_DIVISAS + status de brieve
Al cerrar el archivo de DIVISAS.	CAPTURA_DIVISAS.CIERRAS + status de brieve
-PROCESO DE INGRESO AL SISTEMA	
Al leer el archivo texto en disco duro.	LEE_CLAVE_USUARIO.RESET (arch_uso) + status de brieve LEE_CLAVE_USUARIO.READLN1 (clave + status de brieve LEE_CLAVE_USUARIO.READLN2 (grupo) + status de brieve LEE_CLAVE_USUARIO.CLOSE (arch_uso) + status de brieve
Al actualizar la clave del usuario.	LEE_CLAVE_USUARIO.REWRITE (arch_uso) + status de brieve LEE_CLAVE_USUARIO.WRITELN1 (clave)+ status de brieve LEE_CLAVE_USUARIO.WRITELN2 (grupo) + status de brieve LEE_CLAVE_USUARIO.CLOSE (arch_uso) + status de brieve
-PROCESOS GENERALES	
Al ingresar al sistema pide teclear password, si el password es incorrecto y los intentos exceden el limite (3)	INTENTOS EXCEDIDOS, ejecución del sistema suspendida
Función REDONDEA, un extendad a los decimales iniciados.	Rutina REDONDEA A FRACCION RUTNTIME 207
Función EJECUTA.	Rutina EJECUTA + status de EXECDOS
-PROCESOS DE IMPRESION	
Al ejecutar ENDCAP.	Rutina PONE LOCAL
Al actualizar el status de la impresora.	Actualiza_status_impresora.REWRITE Actualiza_status_impresora.WRITELN Actualiza_status_impresora.CLOSE
Al ejecutar LASER.BAT o ATI.BAT.	EJECUTA_LASER.BAR EJECUTA_ATI.BAT
Al leer el status de la impresora	Lee_status_impresora.REWRITE Lee_status_impresora.RESET
-PROCESOS DE LECTURA	
Función fec_fecha para leer fecha de captura.	ULECTURA.LEE_FECHA
-PROCESOS DE ACTUALIZACION DE PASSWORD	
Pide CLAVE, abre USO_GIRO.	PIDE_CLAVE.ABRE_USO_GIRO + status brieve
al leer USO_GIRO.	PIDE_CLAVE.LEE_USO_GIRO + status de brieve
al CERRAR USO_GIRO.	PIDE_CLAVE.CIERRAS + status de brieve
Al cambiar Password abre el archivo USUARIOS	CAMBIA_PASSWORD.ABRE_USUARIOS + status de brieve CAMBIA_PASSWORD.LEE_USUARIOS + status de brieve
ESCRIBE al archivo USUARIOS.	CAMBIA_PASSWORD.ACTUALIZA_USUARIOS + status de brieve
CIERRA el archivo USUARIOS.	CAMBIA_PASSWORD.CIERRAS + status de brieve

PROCESO	MENSAJE
-PROCESO DE CAPTURA AUTOMATICA	
" ESCRIBE el archivo USUARIOS.	ALTA_CLAVE.ESCRIBE_USUARIOS.
" CIERRA el archivo USUARIOS.	ALTA_CLAVE.CIERRAS + status de btrieve
-AL DAR DE BAJA UN USUARIO.	
" Abre el archivo usuarios.	BAJA_CLAVE.ABRE_USUARIOS + status de btrieve
" Lee del archivo USUARIOS.	BAJA_CLAVE.LEE_USUARIOS + status de btrieve
" Borra del archivo USUARIOS.	BAJA_CLAVE.BORRA_USUARIOS + status de btrieve
" CIERRA del archivo USUARIOS.	BAJA_CLAVE.CIERRAS + status de btrieve.

TABLA 3.5 Mensajes de error del sistema.

En todos los casos expuestos en la tabla anterior, el sistema esperará a que el usuario teclee RETURN o ESC para después cerrar todos los archivos que en ese momento estén abiertos y luego regresará al sistema operativo, un ejemplo de la pantalla que se despliega en estos casos es la siguiente:



G 3.25 Pantalla de error sistema

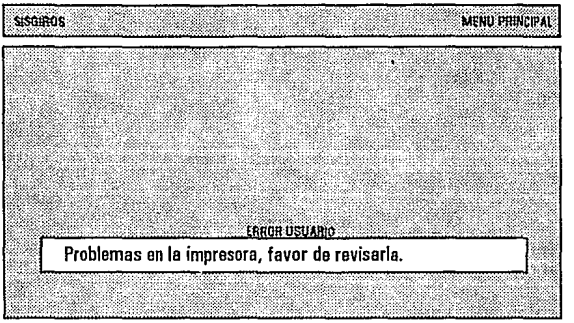
En la siguiente tabla tenemos los mensajes para errores que principalmente se dan en la captura de algún dato o con alguna falla de un dispositivo, estos errores o fallas no ocasionan que el sistema corte su ejecución, simplemente esperan a que sea corregido el error o que el dato sea capturado correctamente.

MENSAJES DE ERRORES QUE NO NECESITAN QUE EL SISTEMA CORTE SU EJECUCIÓN

PROCESO	MENSAJE
-PROCESO DE CAPTURA AUTOMATICA	
Al ingresar los datos: CUENTAHABIENTE CORRESPONSAL DIVISA checa que existan en los respectivos catálogos, si no existe alguno de ellos.	EL CUENTAHABIENTE NO EXISTE EL CORRESPONSAL NO EXISTE LA DIVISA NO EXIXTE
Al ingresar la fecha de expedición, si es menor a la fecha de captura (fecha del sistema).	FECHA INVALIDA
PROCESO DE GENERACION DE FORMATOS	
Al escribir a la impresora, si la operación no es exitosa	PROBLEMAS CON LA IMPRESORA, FAVOR DE REVISARLA
PROCESO DE GIROS	
Al escribir a la impresora, si la operación no es exitosa	PROBLEMAS CON LA IMPRESORA, FAVOR DE REVISARLA
Convierte el número de giro a imprimir de string a numérico, si no es exitoso.	ERROR EN NUMERO DE GIRO
Si el siguiente número que se captura es mayor al ultimo giro del día.	ERROR NUMERO DE GIRO ERRONEO
Al imprimir y la alimentación es manual.	FAVOR DE ALIMENTAR GIRO
PROCESO DE MODIFICACION DE CIFRA$ DE CONTROL (cancelación de giros)	
Si el folio a cancelar no existe.	NO EXISTE EL FOLIO
Si el folio capturado no coincide con el asignado.	EL NUMERO DE FOLIO NO COINCIDE CON EL ASIGNADO
Al terminar las actualizaciones generadas por la cancelación	CANCELACION REALIZADA
PROCESO DE MODIFICACION DE NUMERO DE GIROS	
Al convertir de string a numérico el giro a modificar	CIFRA ERRONEA
PROCESOS GENERALES	
Cuando un programa manda ejecutar a otro o a un archivo batch , mientras se carga el respectivo archivo	ESPERAR, REALIZANDO + proceso
Al mandar a imprimir, si la impresora no esta lista	No esta lista la impresora. Favor de revisarla
Al ingresar al sistema, se pide teclear clave y el correspondiente password	Teclear CLAVE Teclear PASSWORD
Si la clave no existe en el respectivo archivo o el password no es correcto	ERROR, CLAVE INCORRECTA ERROR, PASSWORD INCORRECTO
En la utileria de cambio de password, pide teclear el nuevo password y para asegurar que este correcto pide se reteclee	Teclear PASSWORD NUEVO Reteclear PASSWORD NUEVO
Si el password nuevo no cambia con respecto al anterior	ERROR, PASSWORD NUEVO IGUAL AL ANTERIOR
Si al reteclear el password nuevo, no coincide con el tecleado la primera vez	ERROR, PASSWORDS DIFERENTES, CAMBIO CANCELADO
Al actualizar el password	CAMBIO REALIZADO
En la utileria de dar de alta una clave, pide la nueva clave	Teclear la NUEVA CLAVE
Si la clave a dar de alta ya existe	YA EXISTE CLAVE, ALTA CANCELADA
Cuando la operación de ALTA de un usuario finaliza	USUARIO + clave usuario +DADO DE ALTA
En la utileria de baja de clave	Teclear CLAVE DE BAJA
Si la clave a dar de baja no existe	NO EXISTE CLAVE, BAJA CANCELADA
Si la clave si existe, para confirmar, pide reteclear la clave, si no coinciden, manda el mensaje de error	Reteclear CLAVE BAJA ERROR, CLAVES DIFERENTES, BAJA CANCELADA
Si las claves coinciden, pide se teclee el área del usuario, si esta no existe , manda mensaje	Teclear AREA del USUARIO ERROR, SG + SUP GIROS, OG + OP GIROS
Si el arreo es correcta, se da de baja la clave	USUARIO + clave usuario + DADO DE BAJA
Al inicial el proceso diario, se pide dar la fecha del sistema y la fecha del siguiente dia hábil	Teclear FECHA SISTEMA Teclear FECHA SIG. DIA HABIL
Si la fecha tecleada en fecha del sistema	FECHA MENOR A LA DEL SISTEMA
Si la fecha del siguiente día hábil es menor a la fecha del sistema	FECHA SIG. ANTERIOR A FECHA SISTEMA
Si al ejecutar alguna opción del sistema, y no se a ejecutado alguna anterior (obligatoria)	ERROR, REALIZAR + proceso correspondiente
Si al ejecutar alguna opción del sistema, se detecta que ya ha sido ejecutada (para opciones que solo se pueden ejecutar una vez)	PROCESO REALIZADO
Al imprimir al gnu reporte o formato, mientras se efectúa la operación	AVISO, IMPRIMIENDO + reporte correspondiente
Para los procesos de cifras de apertura y cifras de fin de día, se detecta si son correctas o no	CIFRAS CORRECTAS CIFRAS INCORRECTAS

Tabla 3.6 Mensajes de error usuario

Estos mensajes se despliegan en una ventana con la leyenda correspondiente, esta ventana se borra al detectarse que la falla a sido corregida, un ejemplo de estos mensajes se presenta en la siguiente figura:



G 3.26 Pantalla de error usuario

3.2 Estimación de Costos

El propósito de esta actividad es calcular todos los costos anticipados asociados con el sistema, no sólo el costo de construirlo o desarrollarlo, sino también el costo de instalarlo, de operarlo y el de mantenimiento, ademas de los costos extras [YOURDON, 1993].

Así, los **costos de desarrollo** de un sistema pueden ser variados y múltiples. La siguiente lista es un ejemplo de los factores que pueden influir:

- Los salarios y gastos extras para todo el personal relacionado con el proyecto.
- Costos de capacitación.
- Tiempo de computadora y herramientas de desarrollo para el personal.
- Costo de reclutamiento de personal nuevo.

En proyectos grandes, se debe considerar **costos de instalación**, entre los que destacan:

- Gastos de capacitación de usuarios.
- Gastos de conversión de Bases de Datos.
- Gastos de instalación comercial.
- Gastos de aprobación reglamentaria.
- Gastos de ejecuciones paralelas.

Y finalmente considerar los **costos de operación**, que principalmente son:

- Costos de hardware, materiales y equipos relacionados.
- Costos de software.
- Costos de personal.
- Costos de mantenimiento.

Muchos de estos factores, como se indicará más adelante, se deben evaluar con mucho cuidado, pues es importante destacar que la automatización de la expedición de los giros bancarios fue desarrollada por personal de la misma empresa.

Se debe tener en cuenta que la estimación de costos de un producto de programación es una tarea muy complicada puesto que es difícil hacer estimaciones exactas durante las fases de planeación y de análisis de un desarrollo, debido a la gran cantidad de factores desconocidos en ese momento, por lo que se considera un factor muy importante que contribuye a los retrasos de entrega y sobregiro en presupuestos tan comunes en los proyectos de programación.

En este capítulo primeramente se mencionan los factores principales que influyen en el costo del software, después se presenta información referente a la forma en que se evaluaron los costos y el porcentaje de ahorro al automatizar el proceso de la expedición de los giros bancarios así como de beneficios y finalmente se menciona la evaluación del mantenimiento.

3.2.1 Factores en el Costo de Software

Existen muchos factores que influyen en el costo de un producto de programación, pero los principales son:

- Capacidad del programador o programadores
- Complejidad del producto
- Tamaño del programa y tiempo disponible
- Confiabilidad requerida
- Nivel tecnológico

A continuación se describen estos factores y se da el enfoque para el proyecto en cuestión.

3.2.1.1 Capacidad del Programador

La producción y mantenimiento de productos de programación son tareas laboriosas, por lo que la productividad y la calidad son funciones directas de la capacidad y esfuerzo individual. Existen dos aspectos fundamentales relacionados con la capacidad del programador: **la competencia global del individuo y su familiaridad con el área particular de aplicación**.

Se entiende como **competencia global** a la capacidad para escribir programas de computadora correctos y considerar que las variaciones en la productividad de la programación es un factor significativo para la estimación de costos. Y se cree que en proyectos muy grandes las diferencias individuales tienden a compensarse, pero en proyectos de cinco programadores o menos la diferencia puede ser muy importante **[FAIRLEY, 1988]**.

Se debe tener en cuenta que para este proyecto el grupo de programadores cae dentro de la segunda clasificación, pero es un grupo que tiene tiempo trabajando junto, con lo que se facilitan tanto el análisis del sistema como la integración de los diferentes módulos que comprenden al sistema (los criterios de modularización se describen en el siguiente capítulo) por lo tanto este punto no se considera crítico para la estimación global del costo.

Debemos considerar que una falta de **familiaridad con el área de aplicación** puede implicar baja productividad y poca calidad pero el grupo de trabajo de este proyecto esta íntimamente relacionado con el tema, puesto que su área se trabajo comprende no sólo a la oficina encargada de la expedición y control de los giros bancarios sino de toda la gerencia correspondiente en la empresa.

3.2.1.2 Complejidad del Producto

Existen tres categorías para los productos de programación: **programas de aplicación**, en los que se incluyen procesamientos de datos y programas científicos; **programas de apoyo**, como compiladores, ligadores y sistemas de inventarios y por último **programas de sistema**, como sistemas de base de datos, sistemas operativos y sistemas de tiempo real **[FAIRLEY, 1988]**.

Por lo que podemos clasificar el proyecto dentro de la categoría **programas de aplicación** y consideramos que el nivel de complejidad, en lo que respecta al desarrollo general, no es grande, pero se debe tener mucho cuidado en lo que respecta a los controles de los procesos, por la importancia de la información la cual es confidencial como por ejemplo la generación de la firma facsímil y el control de las diferentes chequeras que se manejan.

3.2.1.3 Tamaño del Producto y Tiempo Disponible

Un proyecto grande de programación es generalmente más caro en su desarrollo que uno pequeño. Para nuestro caso es un producto que no es de gran tamaño, pero como va a interactuar con otros sistemas, se deben considerar algunos factores más que si fuese a trabajar en forma independiente.

El esfuerzo total del proyecto se relaciona con el calendario de trabajo asignado para la terminación del proyecto. Para este sistema no debemos de olvidar que su desarrollo se hace en forma paralela con el desarrollo, mantenimiento y soporte a otros, por lo que se estima un tiempo total de 2 meses, en el cual los programadores no estarán de tiempo completo en el.

Generalmente la estimación del tiempo de desarrollo de un programa se basa tanto en las líneas de código fuente y el esfuerzo total en meses del programador, pero uno de los errores comunes en la estimación de líneas de código fuente de un producto de programación es subestimar la cantidad de código de servicio requerido; este código es la parte del código fuente que permite el manejo de entradas y salidas, comunicación interactiva con el usuario, interfaces de comunicación humana, y la determinación y manejo de errores (información que se presentó al principio de este capítulo). En base a proyectos anteriores podemos mencionar que algunas veces el código de servicio llega a ser más del 50% e incluso hasta el 90% del código total del producto.

Podemos concluir que las estimaciones basadas en términos del código de cómputo suelen ser engañosas cuando se usan para estimar el código final del programa. En nuestra estimación de costos, podemos determinar que el tamaño del proyecto, en lo referente al código fuente y código de servicio, se calcula en 17,000 líneas de código, del cual el 80 % corresponderá al código

de servicio; este valor se calculó considerando 40 caracteres (en promedio) por línea y suponiendo que se tendrán aproximadamente 700,000 Kbits de programas fuente.

3.2.1.4 Nivel de Confiabilidad Requerido

La confiabilidad de un producto de programación puede definirse como la probabilidad de que un programa desempeñe una función requerida bajo ciertas condiciones específicas y durante cierto tiempo. La confiabilidad puede expresarse en términos de exactitud, firmeza, cobertura y consistencia del código fuente. Las características de la confiabilidad pueden instrumentarse en un producto de programación, pero existe un costo asociado con el aumento del nivel de análisis, diseño, instrumentación y esfuerzo de verificación y validación que debe aportarse para asegurar alta confiabilidad.

El nivel de confiabilidad deseado debe establecerse durante la fase de planeación al considerar el costo de las fallas del programa; en algunos casos, las fallas pueden causar al usuario pequeñas inconveniencias, mientras que en otros tipos de productos puede generarse gran pérdida financiera e incluso poner una vida en peligro.

Teniendo en cuenta los puntos anteriormente descritos, es obvio que el proceso de automatización de la expedición y control de los giros bancarios debe de tener un grado de confiabilidad mucho muy grande, pues, como ya se mencionó, se trabajará con información confidencial y afectará la contabilidad de la empresa.

3.2.1.5 Nivel Tecnológico

El nivel de tecnología empleado en un proyecto de programación se refleja en el lenguaje utilizado, la máquina abstracta (tanto el equipo como los programas de apoyo), las prácticas y las herramientas de programación utilizadas. Se sabe que el número de líneas de código fuente escritas por día es, por completo, independiente del lenguaje ocupado, y que las proposiciones escritas en un lenguaje de alto nivel como el Turbo Pascal suelen generar varias instrucciones a nivel de máquina. El uso de un lenguaje de alto nivel, en vez de un ensamblador, aumenta la productividad por un factor de 5 ó 10; además, las reglas de verificación de tipos de datos y los aspectos de auto documentación de estos lenguajes mejoran la confiabilidad y la capacidad de modificación de los programas. Los lenguajes modernos de programación brindan características adicionales para mejorar la productividad y confiabilidad del producto de programación; entre estas características están la verificación fuerte de tipos de datos, la abstracción de datos, la compilación

separada, el manejo de excepciones y de interrupciones, así como los mecanismos de concurrencia.

En el siguiente capítulo se describe el por que se utilizará el lenguaje Turbo Pascal (descripción de la plataforma de desarrollo, tanto software como hardware).

3.2.2. Técnicas de Estimación de Costos del Software

Dentro de la mayor parte de las organizaciones, la estimación de costos de la programación se basa en las experiencias pasadas. Los datos históricos se usan para identificar los factores de costo y determinar la importancia relativa de los diversos factores dentro de la organización. Lo anterior, por supuesto, significa que los datos de costos y productividad de los proyectos actuales deben ser centralizados y almacenados para un empleo posterior.

La estimación de costos puede llevarse a cabo en forma **jerárquica hacia abajo o en forma jerárquica hacia arriba**, estas dos formas se describen a continuación:

La **estimación jerárquica hacia abajo** se enfoca primero a los costos del nivel del sistema, así como a los costos de manejo de configuración, del control de calidad, de la integración del sistema, del entrenamiento y de las publicaciones de la documentación. Los costos del personal relacionado se estiman mediante el examen del costo de proyectos anteriores que resulten similares.

En la **estimación jerárquica hacia arriba**, primero se estima el costo del desarrollo de cada módulo o subsistema; tales costos se integran para obtener un costo total. Esta técnica tiene la ventaja de enfocarse directamente a los costos del sistema, pero se corre el riesgo de despreciar diversos factores técnicos relacionados con algunos módulos que se desarrollarán. La técnica subraya los costos asociados con el desarrollo independiente de cada módulo o componente individual del sistema, aunque puede fallar al no considerar los costos del manejo de la configuración o del control de calidad. En la práctica, ambas técnicas deben desarrollarse y compararse para que iterativamente se eliminen las diferencias obtenidas.

En la actualidad se tienen técnicas como el llamado **Juicio Experto** y la **Técnica DELFI**, los cuales se basan en:

3.2.2.1 Juicio Experto

La técnica más utilizada para la estimación de costos es el uso del juicio experto, que además es una técnica de tipo **jerárquica hacia abajo**. El juicio experto se basa en la experiencia, en el conocimiento anterior y en el sentido comercial de uno o más individuos dentro de la organización.

La mayor ventaja del juicio experto, que es la experiencia, puede llegar a ser su debilidad; el experto puede confiarse de que el proyecto sea similar al anterior; pero bien puede suceder que haya olvidado algunos factores que ocasionan que el sistema nuevo sea significativamente diferente; o quizás, el experto que realiza la estimación no tenga experiencia en ese tipo de proyecto; por lo que generalmente se forman grupos de personas para que entre todos se establezca la estimación.

La mayor desventaja de la estimación en grupo es el efecto que la dinámica interpersonal del grupo pueda tener en cada uno de los individuos; los miembros de un grupo pueden ser inocentes con respecto a factores de tipo político, a la presencia de alguna autoridad dentro del grupo, o al dominio de un miembro del grupo con una fuerte personalidad.

3.2.2.2. Estimación del Costo por la Técnica DELFI

La técnica DELFI puede adaptarse a la estimación de costos de la siguiente manera:

1. Un coordinador proporciona a cada experto la documentación con la definición del sistema y una papeleta para que escriba su estimación.
2. Cada experto estudia la definición y determina su estimación en forma anónima; los expertos pueden consultar con el coordinador, pero no entre ellos.
3. El coordinador prepara y distribuye un resumen de las estimaciones efectuadas, incluyendo cualquier razonamiento extraño efectuado por alguno de los expertos.
4. Los expertos realizan una segunda ronda de estimaciones, otra vez anónimamente, utilizando los resultados de la estimación anterior. En los casos que una estimación difiera mucho de las demás, se podrá solicitar que también en forma anónima el experto justifique su estimación.
5. El proceso se repite tantas veces como se juzgue necesario, impidiendo una discusión grupal durante el proceso.

Es posible que después de varias rondas de estimaciones no se llegue a un consenso; en ese caso, el coordinador deberá analizar los aspectos relacionados con cada experto para determinar las causas de tales diferencias. Puede ser que el coordinador tenga que recabar información adicional y presentársela a los expertos con el fin de resolver las diferencias en los puntos de vista.

En este proyecto se utilizó la técnica del juicio experto, formando el grupo de "expertos" con los mismos programadores, un usuario directo y los jefes de ambas oficinas; este grupo determinó varios puntos importantes, los cuales se muestran en el subindice de análisis de beneficios.

3.2.3 Análisis de Beneficios

En muchos casos es más difícil calcular los beneficios de un nuevo sistema de información que calcular los costos. A continuación hablaremos de los **beneficios tácticos y los estratégicos** de un nuevo sistema. En este contexto, un **beneficio táctico** es aquél que permite que la organización continúe realizando la misma actividad de negocios, pero a menor costo (o mayor ganancia); un **beneficio estratégico** es el que permite comenzar a realizar un tipo de negocios totalmente nuevo, o a hacerlo en un área totalmente nueva o con clientes nuevos **[YOURDON, 1993]**.

3.2.3.1. Beneficios Tácticos

Los beneficios tácticos suelen relacionarse con reducciones en el personal administrativo o de oficina. Un nuevo sistema puede permitir que se realice la misma función con la mitad o menos del número de usuarios que se ocupaban antes, en el caso de la automatización de la expedición de giros bancarios, se utilizará solo un usuario, un interventor y un supervisor, esto es importante si tomamos en cuenta que el proceso manual se efectuaba con tres usuarios y dos supervisores.

Un tipo de beneficio táctico más interesante es el ahorro que resulta de poder procesar mucho más transacciones o giros más rápidamente. El sistema también reportó ahorros en equipos, puesto que en las máquinas que se utilizaban para el contraseñado se gastaba alrededor de unos N$ 18,000.00 anuales (según estimaciones de la división de operaciones de inmuebles de la empresa), mientras que el nuevo sistema se encuentra en la red a la cual ya están conectados los usuarios de la oficina correspondiente.

En el siguiente capítulo se tiene la información de las características de la red de producción donde se instalará SISGIROS.

Los costos de mantenimiento también proporcionan un beneficio, estos costos se discuten el siguiente subtema.

3.2.3.2. Beneficios Estratégicos

Se refieren a la posibilidad de permitir a la organización o empresa hacer cosas que serían imposibles con el sistema anterior, en nuestro caso los beneficios estratégicos serán la posibilidad de que los empleados que quedarán fuera del proceso de la expedición de los giros bancarios se dedicarán a otras labores de la misma oficina, distribuir conocimientos y experiencia a los que previamente sólo tenían acceso una o dos personas dentro de la organización.

Otro beneficio estratégico será la capacidad del sistema para proporcionar información que anteriormente no se tenía, esto se reflejará en los diferentes reportes operativos y de control que ayudarán a la toma de decisiones.

Para finalizar este tema, se mencionan los puntos que se determinaron en el análisis de costos, basándose como ya se mencionó en el **Juicio Experto**:

1 El sistema no tiene una complejidad alta, pero debe de tener los suficientes controles debido, como ya se mencionó, a la importancia de la información que contendrá.

2 El tiempo aproximado tanto para el análisis, desarrollo, validaciones, pruebas y la implantación será de 2 meses, considerando que el tiempo destinado por los analistas no será de tiempo completo por las diferentes tareas que deben desempeñar.

3 Se calcula que el sistema tendrá 17,000 líneas de código, del cual un 80% será de código de servicio.

4 El nivel de confiabilidad será lo más alto posible, debido, entre otras cosas, a los niveles de seguridad (niveles de acceso que se describieron en la primera parte de este capítulo) y a las cifras de control que deberá calcular y confrontar cada uno de los procesos. El sistema garantiza tanto el control como la seguridad de los recursos (entendiendo por recursos a los formatos y firmas facsímil).

5 Con el sistema se conseguirá una reducción aproximadamente de N$ 18,000.00 anuales por concepto de mantenimiento a las dos máquinas de giros utilizadas actualmente.

6 Ahorro en las horas-nombre simultáneamente en las fases operativa y de conciliación.

7 Control durante todas las fases de la operación (captura, registro contable, conciliación y consultas).

8 Reducción de errores al sustituir la operación manual por una automática.

9 Oportunidad y exactitud en la generación de los reportes operativos para la toma de decisiones.

10 Automatización desde la expedición hasta la conciliación de los giros expedidos sobre el extranjero.

3.2.4 Estimación de los Costos del Mantenimiento del Software

El mantenimiento de software suele necesitar de 40 a 60% y en algunos casos hasta 90% del esfuerzo total durante el ciclo de vida del proyecto; estas actividades comprenden agregar mejoras al producto, adaptar el producto para nuevos ambientes de procesos y corregir los problemas de los programas.

La mayor preocupación con respecto al mantenimiento durante la fase de planeación de programación es estimar el número de programadores destinados al mantenimiento que se requerirán, así como especificar las facilidades necesarias para que se lleve a cabo.

Un estimado muy usado en la determinación del número de personas es el total de líneas de código que puede mantener cada programador en forma individual, pero basándonos en las experiencias vividas y en el esquema de la oficina, se determina que sólo una persona quedará encargada del mantenimiento, esto debido a que se tienen tanto estándares para facilitar el entendimiento de cada proceso como el respaldo de que tanto el análisis como desarrollo del mismo se llevará en forma paralela entre todos los integrantes del grupo.

4Diseño Estructurado del Sistema

4.1 Módulos y Criterios de Modularización

En esta sección definimos los diseños bottom-up y top-down así como el concepto de programación modular discutiendo las ventajas de cada uno así como las precauciones que deben ser observadas al ser empleadas [KENDALL,1988], después se definen los conceptos de acoplamiento, cohesión y algunos conceptos de modulación; y por último la forma de diseñar los módulos para SISGIROS.

4.1.1 Diseños Bottom-Up, Top-Down y Desarrollo Modular

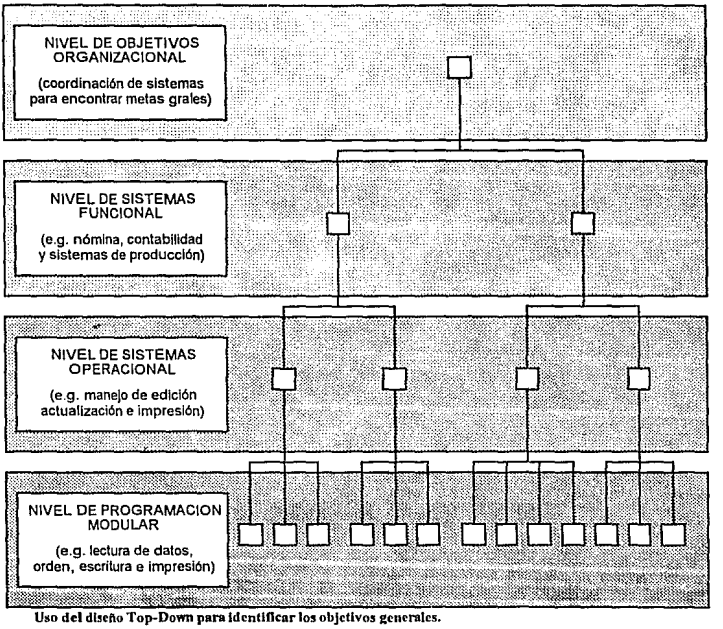
4.1.1.1 Diseño Bottom-Up

El diseño Botton-Up se refiere a tratar de identificar los procesos para resolver los problemas de manera inmediata. Generalmente estos problemas se encuentran en el nivel más bajo de la organización por lo tanto el nombre bottom-up se refiere al nivel en el cual la automatización es introducida. Un ejemplo típico se dá en una empresa que adquiere paquetes de software para contabilidad, para organización y un paquete diferente para cada necesidad.

Al utilizar este tipo de diseño es difícil relacionar los subsistemas para hacerlos actuar como un sistema resultando de gran costo la elaboración de las interfaces necesarias.

4.1.1.2 Diseño Top-Down

El diseño Top-Down significa analizar desde el punto más general dividiendo el sistema en partes menores o subsistemas como se muestra en la figura G 4.1. Este tipo de diseño hace énfasis en las interfaces que los sistemas y sus subsistemas requieren.



G 4.1 **Uso del diseño Top-Down para identificar los objetivos generales.**

Una ventaja de este diseño es su habilidad para tener equipos de análisis trabajando en paralelo en subsistemas diferentes. Una ventaja más es que al utilizarlo se evita que el analista se enfoque a detalles particulares antes de alcanzar objetivos generales.

Las desventajas es que se corre el peligro de dividir al sistema en subsistemas erróneos. Además se tienen que detallar las interfaces entre los subsistemas.

4.1.1.3 Diseño Modular

Una vez que se toma el diseño Top-Down es útil el diseño modular para la programación. Esto significa dividir la programación en porciones manejables.

El diseño modular tiene como meta producir sistemas modulares de programación bien estructurados. En esta sección se considera a cada módulo de programación como una entidad definida que tiene las siguientes características:

1.-Los módulos contienen instrucciones, lógica de procesos y estructuras de datos.

2.-Los módulos pueden ser compilados aparte y almacenados en una biblioteca.

3.-Los módulos pueden quedar incluidos dentro de un programa.

4.-Los segmentos de un módulo pueden ser utilizados por medio de invocar un nombre con algunos parámetros.

5.-Los módulos pueden usar a otros módulos.

Como ejemplos de módulos se incluyen los procedimientos, subrutinas y funciones, así como los grupos funcionales de procedimientos, subrutinas y funciones relacionados; los grupos de abstracciones de datos; los grupos de programas de apoyo y los procesos concurrentes. La modularización permite al diseñador descomponer un sistema en sus unidades funcionales con el fin de imponer un ordenamiento jerárquico en el uso de las funciones; igualmente permite la instrumentación de abstracciones de datos y el desarrollo independiente de subsistemas útiles.

Existen muchos criterios que pueden ser utilizados para definir la modularización de un sistema; dependiendo del criterio utilizado, pueden resultar diferentes estructuras para un sistema dado. Entre los criterios de modularización se incluyen al **criterio convencional** en el que cada módulo junto con sus submódulos corresponden a un paso del proceso en la secuencia de ejecución; así mismo, al **criterio de ocultamiento de información**, en el que cada módulo oculta a otros módulos una decisión difícil o modificable del diseño; al **criterio de la abstracción de los datos**, en el que cada módulo oculta los detalles de representación de una estructura de datos importante debajo de las funciones que acceden y modifican dichas estructuras; a los **niveles de abstracción**, en que los módulos y las colecciones de los mismos proporcionan una jerarquía de servicios más complejos; al **acoplamiento y cohesión**, por medio del cual un sistema se estructura para maximizar sus elementos de cohesión y minimizar el acoplamiento entre sus módulos; así

como, la **modelación de problemas**, por medio de la cual la estructura modular de un sistema se ajusta a la estructura del problema a resolver.

Para el Sistema de Expedición y Control de Giros Bancarios se modularizó utilizando un criterio sencillo de diseño, para lo cual se tomaron diversos aspectos de varios criterios de modularización basándonos en el criterio convencional y en el de acoplamiento y cohesión, del cual, a continuación describiremos sus características más importantes.

4.1.2 Acoplamiento y Cohesión

Una meta fundamental en el diseño de software es la de estructurar al producto de tal forma que el número y la complejidad de las interacciones entre los diversos módulos sea minimizada, lo cual se logra incluyendo los conceptos de acoplamiento y cohesión.

La fuerza del acoplamiento entre dos módulos está influida por la complejidad de la interfaz, por el tipo de conexión y por el tipo de comunicación; se obtienen relaciones obvias a partir de una menor complejidad que de grandes y obscuras complejidades. Así por ejemplo, las interfaces establecidas por bloques comunes de control y de datos, por regiones comunes de traslape (*overlay*) en memoria, por secciones comunes de entrada/salida, así como por nombres globales de variables son más complejas (más fuertemente atadas) que las interfaces obtenidas por el simple pasaje de listas de parámetros entre módulos.

La modificación de un bloque común de datos o de control puede requerir de modificaciones en todas las rutinas que se encuentran acopladas a ese bloque; por otro lado, si los módulos se comunican solamente por los parámetros y si las interfaces entre módulos permanecen constantes, los detalles internos de los módulos pueden ser modificados sin tener que modificar las rutinas que usan los módulos modificados.

La comunicación entre módulos incluye el pasaje de datos, de elementos de control (tales como banderas, interruptores, etiquetas y nombres de procedimientos), así como de las modificaciones de código de un módulo hacia otro. El grado de acoplamiento es menor para la comunicación de datos, mayor para la de conceptos de control y mucho mayor en el caso de módulos que modifican el código de otros módulos **[FAIRLEY, 1988]**.

El acoplamiento se define como *el grado en el cual los módulos se interconectan o se relacionan entre ellos* **[YOURDON, 1993]**. Entre más fuerte sea el acoplamiento entre módulos, más difícil es implantarlo y mantenerlo, pues entonces se necesitará un estudio cuidadoso para la modificación de algún módulo.

El acoplamiento entre módulos puede ser considerado dentro de una escala del más fuerte (el menos deseable) al más débil (el más deseable) de la siguiente forma:

1.-**Acoplamiento del contenido**. Ocurre cuando un módulo modifica los valores o las instrucciones de algún otro módulo.

2.-**Acoplamiento de zonas compartidas**. Los módulos son atados en forma conjunta por medio de zonas globales para la estructura de los datos.

3.-**Acoplamiento del control**. Incluye el pasaje de banderas de control, ya sea como parámetro o en forma global, entre los módulos de tal forma que un módulo controla la secuencia de proceso de otro.

4.-**Acoplamiento por zonas de datos**. Es similar al de zonas compartidas, excepto que los elementos globales son compartidos en forma selectiva entre las diversas rutinas que requieren los datos.

5.-**Acoplamiento de datos**. Incluye el uso de listas de parámetros para pasar a los elementos entre rutinas.

La forma más deseada de acoplamiento es ciertamente una combinación de zonas de datos y de acoplamiento de datos.

La cohesión se define como el *grado en el cual los componentes de un módulo (típicamente las instrucciones individuales que conforman un módulo) son necesarios y suficientes para llevar a cabo una sola función bien definida* [YOURDON, **1993**].Los mejores módulos son aquellos que son funcionalmente cohesivos (módulos en los cuales cada instrucción es necesaria para poder llevar a cabo una tarea) los peores módulos son aquellos que son coincidentalmente cohesivos (cuyas instrucciones no tienen una relación entre uno y otro).

La cohesión interna de un módulo se mide en términos de la fuerza de unión de los elementos dentro del módulo; esta cohesión ocurre dentro de una escala de la más débil (la menos deseada) a la más fuerte (la más deseada) en el siguiente orden:

1.-**Cohesión coincidental**. Ocurre cuando los elementos dentro de un módulo no tienen relación aparente entre cada uno de ellos.

2.-**Cohesión lógica**. Implica relaciones entre los elementos de un módulo; un módulo unido lógicamente por lo común combina varias funciones relacionadas en una forma compleja e interrelacional; lo anterior resulta en el pasaje de parámetros de control, y en un código compartido y engañoso que es difícil de entender y modificar.

3.-**Cohesión temporal**. Presenta muchas de las desventajas de los lógicamente unidos, pero aquí todos los elementos son ejecutados en un momento dado sin requerir de ningún parámetro o lógica alguna para determinar que elemento debe ejecutarse.

4.-**Cohesión en la comunicación**. Los elementos son ejecutados en un momento dado y además se refieren a los mismos datos.

5.-**Cohesión secuencial**. Ocurre cuando la salida de un elemento es la entrada para el siguiente, la estructura del módulo normalmente mantiene un parecido con la estructura del problema; sin embargo una unión secuencial puede contener diversas funciones o partes de una función, ya que los procedimientos de los procesos en un programa pueden ser distintos del funcionamiento del mismo.

6.-**Cohesión funcional**. Representa un tipo fuerte de amarre de los elementos de un módulo debido a que todos los elementos se encuentran relacionados al desempeño de una sola función.

7.-**Cohesión informacional**. Ocurre cuando el módulo contiene una estructura de datos compleja; así como varias rutinas que manejan dicha estructura; cada rutina del módulo presenta unión funcional; esta cohesión es la relación total de la abstracción de los datos; es similar a la cohesión en la comunicación en tanto que ambas se refieren a una sola entidad de datos; sin embargo, difieren en que la comunicación implica que todo el código en el módulo sea ejecutado en cada llamada al mismo; por su parte, la cohesión informacional requiere que solamente el segmento con cohesión funcional sea ejecutado al ser llamado el módulo.

La cohesión de un módulo puede ser determinada por medio de escribir el propósito del módulo y examinarlo **[FAIRLEY, 1988]**, los módulos unidos funcionalmente siempre pueden quedar descritos por una oración simple sobre su propósito; sin embargo, existe un problema potencial al decidir que tanto se puede subdividir un módulo que aparece como funcionalmente unido.

Podemos concluir que la meta de la modularización de un sistema de programación por el uso de los criterios del acoplamiento y cohesión es la de producir sistemas que tengan acoplamiento de zonas de datos y el acoplamiento de datos entre los módulos y además que cuenten con cohesión funcional e informacional en los elementos de cada módulo.

En el subtema 4.1.4, se describe la forma en que se diseñaron los módulos para el Sistema de Control de la Expedición de Giros.

4.1.3 Otros Criterios

Se tienen algunos criterios adicionales para describir que funciones poner en qué módulos de un sistema de programación, entre ellos se incluyen:

1.-El ocultamiento de las decisiones complejas o modificables de un diseño.

2.-Poner un límite al tamaño físico de cada módulo.

3.-La estructuración del sistema para mejorar la claridad y ayudar a las pruebas.

4.-El aislamiento de las rutinas dependientes de la máquina.

5.-El aligeramiento de la labor de modificación.

6.-La generación de funciones generales de apoyo.

7.-El desarrollo de una estructura aceptable de traslapes para una máquina con capacidad limitada.

8.-La minimización del número de fracaso en páginas residentes para sistemas de memoria virtual, así como la reducción de los cargos por llamadas y retornos de funciones.

Para cada producto de programación, el o los diseñadores deben sopesar estos factores y desarrollar un conjunto consistente de criterios de modularización que dirijan al proceso de diseño.

La técnica preferida para optimar la eficiencia de un sistema es primero diseñar e instrumentar el sistema en forma altamente modular; en ese momento se mide el desempeño del sistema, y los cuellos de botella se eliminan por medio de la reconfiguración y la combinación de módulos. La fuerza de esta técnica se basa en dos observaciones:

1.-Durante la mayor parte de la ejecución de los sistemas de programación se emplea una buena porción del tiempo de proceso a zonas pequeñas de código; además la región del código donde se ocupa la mayor parte del tiempo, no es predecible antes del desarrollo y las pruebas de desempeño del sistema.

2.-Resulta relativamente sencillo reconfigurar y combinar pequeños submódulos en módulos mayores si resulta necesario para obtener un mejor desempeño; sin embargo, no efectuar una descomposición inicial suficiente en un sistema puede evitar que se identifiquen funciones que pueden ser usadas en otros contextos.

4.1.4 Modularización del Sistema de Expedición de Giros

En la práctica se tienen algunas consideraciones adicionales que evitan una obtención estricta de las metas del acoplamiento y la cohesión. Además, resulta difícil decidir exactamente que nivel de acoplamiento o cohesión se presenta en los diversos segmentos de un sistema. Generalmente se ve a los niveles como ideas sugerentes más que como criterios estrictamente cuantitativos; así, deben ser utilizados como guías. Sin embargo los conceptos de acoplamiento y cohesión proporcionan un marco muy valioso para pensar en la modularización del sistema, estos módulos son:

1.-**Módulo de adquisición de datos** (giros, archivos o catálogos). Con el submódulo de validación.

Sus principales funciones son:

Captura de giros

Captura automática de giros

Captura de beneficiarios

Captura de corresponsales

Captura de cuentahabiente

Captura de divisas

2.-**Módulo para producir el documento** (giro). Con el submódulo de generación de la firma facsímil.

Las principales funciones son:

Lectura del disco llave

Configuración de la impresora para la impresión de los documentos

Generación e impresión del documento y de la firma facsímil

Actualización de cifras de control

3.-**Módulo de control y contabilización**. Con los submódulos de generación de operaciones y de conciliación.

Sus principales funciones son:

Control de usuarios

Mantenimiento a catálogos

Interfase con el sistema contable de la empresa

Interfase con el sistema de conciliaciones de la empresa

4.-**Módulo de información**.

Sus principales funciones son:

Informe de giros por generar

Informe de giros generados

Informe de captura

Reporte de los catálogos

Reporte de cifras de control

5.-**Módulo de control de los recursos del sistema**.

Sus principales funciones son:

Cifras de control (inicio y fin de día)

Modificación de cifras

Debemos destacar que estos módulos tienen otra agrupación en el menú del sistema, esto debido a la secuencia recomendada para el proceso completo de un día de trabajo (ver capítulo 3).

Esta división se realizó basándose en los conceptos de acoplamiento y cohesión, pero es importante mencionar que se consideraron aspectos como el de tener funciones generales de apoyo en módulos independientes (en programas o código compilable independientemente), que la estructura del sistema ayudará a su fácil entendimiento para la realización de pruebas y su mantenimiento, en realidad no se tiene problemas con el tamaño del sistema por lo que no fue necesario el limitar el tamaño físico de los módulos y el aislamiento de funciones para el proceso de la generación de la firma facsímil (esto obviamente por razones de seguridad).

Dentro de las funciones generales (en los módulos independientes que se mencionan en el párrafo anterior) tenemos:

Manejo de excepciones

Manejo de apertura de archivos de datos

Manejo de variables tipo numérico

Funciones de interacción con el sistema operativo (MSDOS)

4.2 Selección de la Plataforma de Desarrollo de Software y Hardware

4.2.1 Criterios de Selección

La actividad de diseño involucra el desarrollo de una serie de modelos, de forma similar a la que el analista desarrolla. Los modelos más importantes para el diseñador son el modelo de implantación de sistemas y el modelo de implantación de programas.

El modelo de implantación de sistemas, a su vez se divide en un modelo del procesador y uno de tareas. La primera tarea a la que nos enfrentamos es decidir como asignar el modelo esencial (la parte automatizada) a las piezas principales de hardware y software del sistema.

Para realizar estas asignaciones se deben de tener en cuenta varios factores, los principales son:

- **Costo**:
  Dependiendo de la naturaleza del sistema, la solución más económica puede ser un grupo de micro computadoras de bajo costo; para otros sería más práctico y económico hacer la implantación con la infraestructura existente en la organización.
- **Eficiencia**:
  El tiempo de respuesta de los sistemas en línea es de suma importancia, por lo tanto se debe escoger procesadores y dispositivos de almacenamiento de datos suficientemente rápidos y poderosos para satisfacer los requerimientos de desempeño del sistema.
- **Seguridad**:
  El usuario final puede tener requerimientos de seguridad que dicten que algunos (o todos) los datos delicados se coloquen en lugares protegidos.
- **Restricciones Políticas y Operacionales**:
  La configuración de hardware puede verse influenciada también por

restricciones políticas impuestas directamente por el usuario final, por otros niveles de administración dentro de la organización o por el departamento encargado de todos los sistemas de cómputo [YOURDON,1993].

Para la elección del software a emplear en el desarrollo del sistema, además de los criterios mencionados anteriormente, existen algunos más que deben tomarse en cuenta. Se ha estado escribiendo programas de computación desde que se desarrollaron las primeras computadoras de propósito general. Los programas se escriben con lenguajes de programación, es conveniente agrupar los distintos lenguajes de programación en cuatro generaciones distintas:

Lenguajes de Primera Generación:

Fueron los lenguajes de máquina que se usaron en los años 50. los programadores que intentaban que la computadora hiciera algo útil codificaban sus instrucciones con unos y ceros binarios.

Lenguajes de Segunda Generación:

Son los sucesores del lenguaje de máquina; generalmente se conocen como lenguajes de ensamble o ensambladores. Estos lenguajes son de bajo nivel en el sentido de que el programador tiene que escribir una declaración por cada instrucción de máquina. En lugar de pensar en términos del problema que se quiere resolver, se debe pensar en términos de la máquina.

Lenguajes de Tercera Generación:

Son la norma actual; incluyen Basic, Cobol, Fortran, Pascal, C, y muchos más. Son de alto nivel en el sentido de que una sola declaración usualmente representa cinco o diez declaraciones de lenguaje ensamblador (y a veces más).

Los lenguajes de tercera generación también se caracterizan como lenguajes guiados por procedimientos. Requieren que el programador piense con cuidado la secuencia de los cálculos o procedimientos necesarios para lograr alguna acción.

Lenguajes de Cuarta Generación:

Los lenguajes de cuarta generación, o 4GLs, son la moda actual y son considerados como el desarrollo más importante en el campo de software en los últimos 20 años.

La mayor parte tiene características de programación estructurada ausentes en los lenguajes de tercera generación. En lo particular, la mayoría de los detalles tediosos de programación relacionados con introducir datos a la computadora se ocultan al programador.

Sin tomar en cuenta el lenguaje de programación que se use, existen puntos comunes que se deben considerar, los más comunes son:

- **Productividad:**
  Probablemente, la cuestión más importante de la programación actual sea la productividad: escribir más software, más rápidamente. Exceptuando casos raros la productividad se considera más importante actualmente que la eficiencia.
- **Eficiencia:**
  En algunas aplicaciones, la eficiencia sigue siendo de importancia. Esto sucede en muchos sistemas de tiempo real, y puede darse en otros tipos de sistemas que procesan grandes volúmenes de datos. La eficiencia usualmente entra en conflicto con otras metas: si se emplea mucho tiempo en el desarrollo de un programa eficiente, es probable que sea menos mantenible y menos transportable.
- **Corrección:**
  Se puede argumentar que esto es lo más importante, si el programa no funciona correctamente, no importa que tan eficiente sea. Se prefieren lenguajes de programación como Ada y Pascal si la corrección és de importancia crítica, porque son de tipos rígidos y el lenguaje revisa todo cuidadosamente para evitar referencias ilegales a los datos.
- **Portabilidad:**
  El usuario puede desear ejecutar el mismo sistema en varios tipos distintos de computadoras. Algunos lenguajes de programación son más portátiles que otros, esto es más cierto en lenguajes de tercera generación (C, Pascal, Fortran, Cobol, etc.) que en los de cuarta. No existe un lenguaje universalmente portátil, por ello, además del lenguaje de programación es necesario tener en cuenta el estilo de programación si la portabilidad es un factor importante.
- **Mantenimiento:**
  Debemos recordar que los sistemas viven durante mucho tiempo, por lo que el software debe tener mantenimiento. **[YOURDON, 1993].**

4.2.2. Selección de Hardware

Para poder elegir el equipo a utilizar, el primer paso es realizar un inventario del equipo que se tiene actualmente para descubrir que se tiene y que se puede utilizar de esto. Además se debe trabajar conjuntamente con los usuarios ya que la determinación del hardware viene en conjunción con la determinación de los requerimientos de información, el conocimiento de la estructura organizacional puede también ser valioso para tomar alguna decisión [KENDALL, 1988] .

La elección del equipo se vio influenciada por la administración de la empresa, ya que en el área de la oficina usuaria se cuenta con el siguiente equipo ya instalado:

Una Red de Area Local Ethertwist, con sistema operativo Novell Netware 3.11

Diecisiete equipos de micro computadoras conectadas a la red.

Dieciocho impresoras de matriz de puntos, de las cuales 17 se encuentran conectadas a cada uno de los nodos de la red, y una impresora conectada a una cola de impresión de la red.

Una impresora láser, conectada a una cola de impresión en la red.

Además se cuenta con algunos equipos de micro computadoras que se encuentran independientes, y actualmente se tienen para actividades secretariales.

Teniendo en cuenta el equipo que se tiene, y considerando que la carga de trabajo de la red no se encuentra al 100% de su capacidad, la elección del hardware se vio influenciada por la administración de la empresa, ya que la oficina usuaria cuenta con un nodo de la red, desde el cual puede accesar el sistema de giros, por lo que se sugirió se desarrollara en un ambiente de red, además de cumplir con los factores mencionados en el punto 4.2.1.:

Costo: Al implementar un sistema nuevo sobre una estructura de hardware existente, se elimina el costo implícito en este ya que no es necesario adquirir nuevo equipo.

Eficiencia: Dado que el equipo se encuentra ya instalado y en operación, se ha observado que tiene un grado de eficiencia aceptable, además el sistema operativo de red utilizado ha sido probado no solo en la organización, si no es reconocido como el mejor sistema operativo para redes de área local hasta la fecha.

Seguridad: El sistema operativo permite controlar la seguridad en cuatro diferentes niveles [NOVELL CONCEPTS]:

Seguridad de Acceso:
Determina quien, cuando, y desde que estaciones pueden trabajar, así como los recursos que pueden ser utilizados.

Derechos de Asignación de recursos:
Controla los directorios, subdirectorios y archivos que un usuario puede accesar y que acciones pueden realizar.

Seguridad de atributos:
Asignación de propiedades especiales a directorios o archivos.

Seguridad en el servidor de archivos:
Deshabilita el teclado hasta el momento en que se digite la palabra de acceso asignada por el administrador de la red.

Como se muestra, la plataforma de hardware en la cual será instalado el sistema, contempla los factores para la elección y cumple con ellos satisfactoriamente. Por último es importante mencionar que el ambiente de desarrollo es igual al de producción.

4.2.3. Selección de Software

Al igual que el hardware, la decisión en la elección de software se vio influenciada por restricciones de la empresa.

En la empresa se tienen estándares los cuales deben ser seguidos, entre ellos se encuentran los siguientes:

Lenguaje de Programación: Turbo Pascal [PASCAL]

Librerías de programación : Turbo Professional [PROFESSIONAL]

Manejo de Archivos en red: Btrieve [BTRIEVE] y Xtrieve [XTRIEVE]

Estos estándares cumplen con los requisitos del punto 4.2.1. satisfactoriamente, como se muestra a continuación.

Costo: Al utilizar el software de desarrollo con el que se cuenta en la empresa, el costo implícito en la adquisición del mismo se elimina.

Productividad: Lo importante es escribir más software más rápidamente, por lo que el emplear un software de desarrollo que se tiene dentro de la empresa como estandar, incrementa la productividad ya que se conoce bien y no se requiere tiempo de aprendizaje, con esto, el tiempo de programación disminuye y se genera más código en menos tiempo.

Eficiencia: Al conocer los productos de software que se tienen como estándares, permite generar código eficiente, ya que se conocen la mayoría de las caracterísitcas del mismo, lo cual permite sean explotadas al máximo y se genere código más eficiente.

Corrección: La corrección del código generado es relativamente fácil, ya que como se mencionó es uno de los lenguajes de programación que tienen la característica de ser fácil de corregir. Aunado a esto, la corrección no necesita ser realizada por el programador que creó el código, dado que es un estándar cualquier persona del área de sistemas podría realizarla.

Portabilidad: Como se mencionó en el tema 4.2.1, los lenguajes de tercera generación son los más portátiles, al ser Pascal un lenguaje de tercera generación la portar el sistema hacia un equipo diferente al seleccionado es relativamente una tarea sencilla.

Mantenimiento: el mantenimiento del sistema con las herramientas seleccionadas resulta sencillo, y puede ser realizado por cualquier persona del área de sistemas ya que las herramientas son estándares dentro de la empresa.

Como se puede observar en los párrafos anteriores, la elección de hardware y software, aún al ser influenciadas por los administradores de la empresa ha utilizar los recursos existentes, estos cumplen con los diversos aspectos que se deben tomar en cuenta en la selección de los mismos.

5 Desarrollo del Sistema

La fase de desarrollo o instrumentación de la programación tiene que ver con la traducción de las especificaciones de diseño a código fuente. El objetivo principal es el escribir código fuente y la documentación interna de modo que la concordancia del código con sus especificaciones sea fácil de verificar, y que se faciliten la depuración, pruebas y modificaciones.

Este objetivo puede alcanzarse haciendo el código fuente tan claro y sencillo como sea posible. La claridad del código fuente se mejora mediante técnicas de codificación, buenos comentarios internos, y por las características que proporcionan los lenguajes de programación [FAIRLEY,1988].

5.1 Definición de Estándares de Programación

5.1.1 Estilo de programación

El estilo es la ruta consistente de elecciones hechas entre caminos alternos de lograr un efecto deseado. En programación el estilo de codificación se manifiesta en las rutas que usa el programador para expresar una acción o un resultado deseado.

No hay un conjunto único de reglas que se puedan aplicar en todas las situaciones; sin embargo, hay principios generales que son ampliamente aplicables y se muestran a continuación:

- **Empléense unas cuantas construcciones estándar de control:** El anidamiento de secuencias de proposiciones, la selección entre alternativas, y un mecanismo para iteración son suficientes. Esto hará más uniforme el estilo de codificación entre los programadores, con

el resultado de que los programas serán más fáciles de leer, comprender y modificar.

- **Utilícense las estructuras goto de manera disciplinada**:
  El propósito de la proposición goto en los lenguajes modernos es permitir la construcción de patrones de estilo que no son proporcionados por el lenguaje; la proposición goto no se proporciona para fomentar la violación de las prácticas de codificación estructurada. Acoplada con la proposición if, permite al programador la definición de mecanismos de flujo de control. La proposición goto debe usarse para ampliar los objetivos de sencillez, claridad y elegancia de los programas.
- **Introdúzcanse tipos de datos definidos por el programador**:
  El uso de distintos tipos de datos hace posible a los humanos y a los sistemas de computación el distinguir entre entidades del dominio del problema.
- **Cúbranse las estructuras de datos bajo las funciones de acceso**:
  Este principio es manifestación del principio de cubrimiento de la información. El cubrir las estructuras de datos bajo las funciones de acceso es el enfoque tomado en el encapsulado de datos, en donde una estructura de datos y sus rutinas de acceso quedan encapsuladas en un solo módulo.
- **Proporciónense prólogos estándar de documentación**:
  Un prólogo de documentación contiene la información acerca de un subprograma o unidad de compilación que no resulta obvia al leer el texto fuente del subprograma o unidad de compilación.

Así mismo, existen algunos principios generales de lo que no se debe hacer para un buen estilo de programación, estos se muestran a continuación:

- **No hay que ser demasiado complicado**
- **Evítense las proposiciones then nulas**
- **No se anide en forma muy profunda**
- **Evítense efectos colaterales oscuros**
- **No se suboptimice**
- **No se emplee un identificador para propósitos múltiples**

5.1.2. Estándares de Programación

Los estándares de codificación son las especificaciones para la definición de un estilo uniforme de codificación. Dada una situación en la que existen diversos caminos para lograr un efecto, se especifica un camino particular. A menudo, los estándares de programación son vistos como mecanismos para restringir y devaluar las habilidades de los programadores para resolver problemas de forma creativa.

La creatividad siempre ocurre dentro de un marco de trabajo básico de estándares. Así, es deseable que todos los programadores de un proyecto adopten un estilo de codificación similar, de modo que se produzca un código de calidad uniforme. Esto no significa que los programadores deben pensar igual, o que deben instrumentar todos los algoritmos en la misma forma.

Para la programación del sistema de expedición de giros, se establecieron los siguientes estándares, considerando el lenguaje de programación y el compilador a usar (ver capítulo 4), estos se muestran en la tabla siguiente:

Se recomienda escribir con mayúsculas: **Las palabras reservadas:** **PROCEDURE, FUNCTION, TYPE, VAR, CONST, PROGRAM, UNITS, CONSTRUCTOR** **Los nombres de procedimientos o funciones programadas. Tanto en su definición como en su referencia, por ejemplo:** **LIMPIA_VENTANA** **DESPLIEGA_ERROR** **Las palabras reservadas BEGIN y END de los procedimientos o funciones programadas**
Para indicar comentarios se usará el símbolo { y no (*. Este segundo símbolo se usará para insertar y anular código de depuración o prueba.

El código de un bloque está a la misma altura de su BEGIN y su END; por ejemplo:

```
Repeat
  Fastwrite(Pad('Esc-Terminar', 80), 25,1, BlackonLTGray);
  ReadString('DIVISA          : ', 16, 14, 3,WhiteonBlue,
              YellowonBlack, LtGrayonBlack, escape,
              string_aux);
  if not escape then
    begin
    Fastwrite(Pad(' ', 80), 25,1, LtGrayonBlack);
    move(blancos[1], llave_divisa[1], sizeof(llave1_div));
    LEE_DIVISA(wr_divisa, llave_divisa, status_io);
    if status_io = ok_io then
      divisa := wr_divisa.clave_div;
    end;
until (escape) or (status_io = ok_io);
```

Documentación interna:

En cada rutina explicando lo que hace, a continuación de su declaración.

Junto al código que por sí mismo no sea lo suficientemente claro

Junto a las variables que tengan una función poco clara

Por ejemplo:

```
PROCEDURE FIN_CAPTURA;
  { imprime cifras de control del proceso y cierra archivos }
```

Para los nombres de las rutinas propias de Turbo Pascal o Turbo Professional se usarán cómo se encuentran declaradas originalmente, ejemplo:

```
PopSublevel
MenuItem
SetStringAttr
```

Las líneas de código no deben rebasar 80 columnas de forma tal que siempre aparezcan en la pantalla. Si las rebasan, hay que dividirlas en dos o más líneas.

Los campos de un archivo siempre deben terminar con una subraya y tres letras que identifican al archivo.

Ejemplo: campo_buz que es un campo del archivo buzón

Los campos de las llaves de un archivo siempre deben terminar con una subraya, una k y tres letras que identifican al archivo

Ejemplo campo_kbuz de la llave del archivo buzón

Las constantes, tipos y variables que sólo son usados en una unidad, sólo deben estar declarados dentro de esa unidad, al iniciarse el IMPLEMENTATION área de la unidad Si las declaraciones son externas -pueden ser usadas por otras unidades-, entonces se debe crear una unidad exclusivamente con las declaraciones externas, de forma tal que pueda hacerse referencia en otras unidades sin necesidad de emplear todo el código de la unidad que las manipula principalmente
Las rutinas o funciones deben ser específicas, esto es, deben realizar una sola función o actividad (cohesión funcional). Además, se debe procurar que no excedan las 60 lineas
Siempre que sea posible, las variables deben declararse localmente en la rutina o pasarse como parámetros. Solo se usarán variables globales cuando sea necesario. Aquellas variable que hayan sido declaradas globalmente no deberán pasarse como parámetros
Indentar operadores AND y OR cuando la condición sea compleja if (llave_ant.cuentahabte_girk =llave_giros.cuentahabte_girk) and (((llave_ant.divisa_girk <> dolar_americano) and (llave_giros.divisa_girk <> dolar_americano)) or ((llave_ant.divisa_girk = dolar_americano) and (llave_giros.divisa_girk = dolar_americano))) then
Los nombres de las unidades (UNITS) serán todos con mayúsculas DOS, { funciones del DOS } TPCRT, { manejo de teclado, pantalla, colores y sonido } TPSTRING, { manejo de strings } PRINTER; { manejo de la impresora}
Procurar no usar archivos INCLUDE, usar UNITS de preferencia, y si se usan poner comentario de que programa hace referencia a el.

Tabla 5.1. Estándares de programación

5.1.3 Desarrollo del Software en Lenguaje de Programación Pascal

Como se ha mencionado anteriomente, el lenguaje para desarrollar sistema SISGIRO es PASCAL, en el ANEXO 2 se presentan los listados de los principales programas del sistema, además se presenta en cada uno de los listados una pequeña explicación con el objetivo del programa y de cada FUNCION o PROCEDIMIENTO.

Se tienen también comentarios que describen el algoritmo utilizado en cada proceso.

5.2 Verificación, Validación y Pruebas del Sistema

5.2.1 Verificación y Validación del Sistema

Los objetivos de estas actividades son valorar y mejorar la calidad de los productos del trabajo generados durante el desarrollo y modificación del software. Los atributos de la calidad deben ser la corrección, la perfección, la consistencia, la confiabilidad, la utilidad, la eficacia, el apego a los estándares y la eficacia de los costos totales.

Hay dos tipos de verificación:

- **Formal**
  Es una rigurosa demostración matemática de la concordancia del código fuente con sus especificaciones.
- **Del ciclo de vida**
  Consiste en el proceso de determinar el grado en que los productos

de trabajo de una fase dada del ciclo de desarrollo cumplen con las especificaciones establecidas durante las fases previas.

La validación es la evaluación del software al final del proceso de desarrollo del software para determinar su conformidad con los requisitos.

Generalmente se expresan estas definiciones de la siguiente manera [FAIRLEY,1988]:

- **Verificación:**
  ¿Estamos construyendo correctamente el producto?
- **Validación:**
  ¿Estamos construyendo el producto correcto?

La calidad no se puede lograr sólo mediante la prueba del código fuente; la verificación y validación implican la valoración de los productos de trabajo para determinar el apego a las especificaciones. Estas incluyen las especificaciones de requisitos, la documentación del diseño, diversos principios generales de estilo, estándares del lenguaje de instrumentación, estándares del proyecto, estándares organizacionales y expectativas del usuario. Se deben examinar los requisitos para asegurarse que concuerden con las necesidades del usuario, así como con las restricciones del ambiente y los estándares de notación.

Los errores ocurren cuando cualquier aspecto de un producto de software es incompleto, inconsistente o incorrecto. Las tres grandes clases de errores del software son:

- **Errores de requisitos**
  Se provocan por una propuesta incorrecta de las necesidades del usuario, por falta de una especificación completa de los requisitos funcionales y de desempeño, por inconsistencia entre los requisitos y por requisitos no factibles.
- **Errores de diseño.**
  Se introducen por fallas al traducir los requisitos en estructuras de solución correctas y completas, por inconsistencia tanto dentro de las especificaciones del diseño y como entre las especificaciones de diseño y los requisitos.
- **Errores de instrumentación.**
  Son los cometidos al traducir las especificaciones del diseño en código fuente. Estos errores pueden ocurrir en las declaraciones de

datos, en las referencias a los datos, en la lógica del flujo de control y en operaciones entrada/salida.

En el siguiente subtema se muestran las pruebas efectuadas para valorar y determinar los errores del sistema.

5.2.2 Pruebas del Sistema

Las pruebas del sistema implican dos clases de actividades:

- **Pruebas de integración.**

Las estrategias para integrar los componentes del software en un producto que funcione incluyen las estrategias

Ascendente.

Es la estrategia tradicional empleada para integrar los componentes de un sistema. Consiste en pruebas de unidad (que tienen como objetivo el descubrir errores en los módulos individuales), seguidas por pruebas de subsistemas (que se realizan para verificar la operación de las interfaces entre los módulos) y luego por pruebas de sistema completo (se relacionan con la lógica de decisión, el flujo de control, los procedimientos de recuperación y las características de tiempo del sistema completo).

Descendente.

Empieza con la rutina principal y una o dos rutinas inmediatamente subordinadas en la estructura del sistema. Después de que este "esqueleto" de alto nivel ha sido probado con detenimiento, se convierte en el modelo de prueba para sus subrutinas inmediatamente subordinadas.

En la integración de los módulos de SISGIROS se utilizó primordialmente la estrategia ascendente, debido a que cada integrante del equipo de desarrollo probó individualmente cada módulo desarrollado y después se fueron uniendo en los submódulos del sistema hasta lograr la integración final del sistema con sus respectivas pruebas.

- **Pruebas de aceptación.**

  Implican la planeación y ejecución de pruebas funcionales, de desempeño y de tensión para demostrar que el sistema implantado satisface sus requisitos.

  En las pruebas de aceptación para SISGIRO se efectuaron dos conjuntos de pruebas: aquellas desarrolladas por el propio equipo de desarrollo del sistema y las que realizaron los usuarios. En ambos casos se incorporaron pruebas de unidad y de aceptación así como pruebas específicas de los controles y resultados de salida del sistema. Se realizaron pruebas de cada uno de los módulos, en cada una de sus diferentes opciones.

  Con los resultados de las pruebas desarrolladas por los usuarios se pudo determinar que el sistema cumple con los requisitos que ellos mismos establecieron en la fase de análisis, y con los resultados de las pruebas desarrolladas por el equipo de trabajo se determinó la integración de todos los módulos que integran SISGIROS.

Las pruebas constaron de diferentes ejemplos de cada una de las operaciones que realiza el sistema, dentro de ellos se tenían casos críticos para evaluar el comportamiento del sistema; además se ejecutaron todos los procesos del sistema obteniendo en todas las pruebas un resultado correcto, con lo cual los usuarios mostraron gran confianza en el sistema y pidiendo su pronta implantación.

Por los resultados de las pruebas efectuadas podemos concluir que SISGIROS es un producto de software que satisface las necesidades del usuario, se apega a sus especificaciones de requisitos y diseño, y presenta una ausencia de errores. Se utilizó una combinación de las técnicas para evaluar y mejorar la calidad del producto.

Sería incorrecto ver las pruebas de código fuente como el principal vehículo para el mejoramiento de la calidad del sistema,; aunque las pruebas son una técnica importante, el evaluar y mejorar la calidad del producto es un concepto presente en todo el ciclo de vida de SISGIROS.

6 Implantación y Mantenimiento del Sistema

6.1 Instalación del Sistema

Los procesos de implantación y liberación del sistema generalmente se dividen en tres fases, las cuales son:

1. Conversión
2. Instalación
3. Capacitación

En los siguientes párrafos se describen estos procesos.

Mucha gente piensa que la labor del grupo de análisis y desarrollo acaba cuando se termina de probar el sistema. Desafortunadamente, queda aún algo qué hacer, aunque ya no es su papel de analista.

La ***conversión*** es la tarea de traducir los archivos, formas y bases de datos actuales del usuario al formato que el nuevo sistema requiere, en el caso de SISGIRO esta etapa solo consisitió en actualizar los archivos de control con los números del primer y último folio de los giros de cada una de las chequeras, así como capturar los datos a la base (catálogos). Esta actualización se realizó en forma automática desde el mismo sistema por lo que fue relativamente fácil la conversión, es obvio que no se necesitó crear un programa para realizarlo.

La ***instalación*** fué un asunto instantáneo y transparente para el usuario ya la infraestructura se tenía lista, y solo fue necesario realizar algunos modificaciones, los pasos realizados fueron:

Crear la estructura necesaria en la red de producción para el SISGIROS.

Transferir el sistema del ambiente de desarrollo hacia el de producción, archivos de datos (actualizados) y programas.

Abrir cuentas en la red de producción para los usuarios del SISGIROS en caso de no tenerla, o modificar los derechos de los usuarios.

Entregar el disco llave (para generación de firma facsimil) al usuario.

La ***capacitación*** es la tarea final del equipo de desarrollo, para poder realizar la liberación del sisema, para esto, es necesario elaborar un plan de capacitación que contemple los siguientes puntos:

¿Cómo se llevará acabo?

¿Quién llevará acabo la capacitación?

¿A quién se preparará y en que horario?

La etapa de capacitación se llevo acabo conjuntamente con la etapa de pruebas y liberación (punto 6.2.), en la cual se le explica al usuario la forma de trabajar del sistema y la forma de interactuar con el, además de proporcionarle el manual del usuario (punto 6.2.1).

La capacitación se llevo a cabo por los miembros del equipo de desarrollo, ya que son los que mejor conocen el sistema técnicamente, apoyados por personal de la oficina usuaria, que es un experto en la operativa actual, para destacar las similitudes entre los procesos manuales y automatizados.

El personal al cual se le dio capacitación, fue seleccionado por los directivos de la empresa, ya que son ellos los que asignan las labores a desarrollar para cada uno de los empleados, cubriendo los diferentes niveles de usuarios que permite SISGIROS.

6.1.1. Manual del Usuario

Como se mencionó anteriormente, uno de los puntos importantes para la implantación y liberación de un sistema es la capacitación del personal encargado de operarlo, para este fin es necesario contar con documentos de apoyo que puedan servir como referencia a diversos aspectos del sistema, uno de estos documentos es el manual del usuario, el cual sirve de base para llevar acabo la capacitación.

El manual de usuario de SISGIROS se muestra en el anexo I.

6.2 Liberación del Sistema

Para poder liberar un sistema es necesario tener una estrategia de liberación, es decir, elaborar un plan por medio del cual el sistema entre a la etapa de producción.

Para el caso de SISGIROS, este plan estuvo ligado a las etapas de pruebas e instalación, y consistió en los siguientes pasos:

Se realizó la instalación del sistema en la red de producción, cómo primer paso.

Se efectuaron las etapas de pruebas de aceptación y capacitación del personal al mismo tiempo (punto 5.3. y 6.1.), para que el personal de la oficina usuaria se familiarizara con SISGIROS, al mismo tiempo que se capacitaba en su uso.

La fase de pruebas y capacitación se estimó en un tiempo máximo de 15 días, pero dado el desempeño y por petición del personal se prolongó al doble de tiempo.

Después de la fase anterior, el sistema y los usuarios se encontraban listos para poder trabajar ya en producción, por lo cual el sistema fue liberado.

6.3 Mantenimiento del Sistema

El mantenimiento del software son las actividades que se realizan después de integrar un producto, en este caso, después de liberar SISGIROS en la red de producción.

6.3.1. Generales

La fase de mantenimiento del ciclo de vida de un producto es el periodo en el que desempeña un trabajo útil. En el desarrollo de SISGIROS se llevó un periodo de 4 meses, mientras que la fase de mantenimiento durará de 10 a 15 años.

Las actividades de mantenimiento implican mejorar el producto, adaptarlo a nuevos ambientes y corregir problemas. La mejora en los productos de software puede dar como resultado proporcionar nuevas capacidades funcionales, mejorar los despliegues al usuario y los modos de interacción, revalorar los documentos externos y la documentación interna, o revalorar las características del desempeño del sistema.

No debemos olvidar que SISGIROS es un sistema desarrollado por personal de la misma empresa y que uno de los integrantes del equipo de trabajo será el encargado de dar mantenimiento al mismo (lo cual se mencionó en el tema de costos del sistema). El encargado estará íntimamente familiarizado con el producto (el entiende la filosofía del diseño y porque funciona de ese modo)

Tomando en cuenta que uno de los principales objetivos del desarrollo de software es la producción de sistemas que faciliten su propio mantenimiento, el mejoramiento y adaptación del software reinician el desarrollo en la fase de análisis, mientras que la corrección de un problema puede reiniciar el ciclo de desarrollo en la fase de análisis, en la fase de diseño o en la implementación. por lo tanto todas las herramientas y técnicas utilizadas para desarroilar el software son potencialmente útiles para su mantenimiento.

La actividad de análisis durante el mantenimiento implica la comprensión del alcance y efecto de la modificación deseada, además de las restricciones para hacer la modificación, lo anterior es muy importante para evitar que al realizar una modificación al sistema no se provoque algún error en algún otro proceso o módulo. .El diseño durante el mantenimiento supone rediseñar el producto para incorporar los cambios deseados. Entonces estos deben implantarse, la documentación interna y externa deben ser actualizadas y realizar las pruebas necesarias.

Todo lo mencionado en el párrafo anterior no tiene mayor alcance en el mantenimiento de SISGIROS puesto que todos los integrantes del equipo de trabajo conocen el sistema y la forma en que esta diseñado cada módulo, esto se facilita gracias a los estándares de programación, a la documentación externa e interna y que se realizarán mediante un enfoque sistemático y ordenado, analizando los requisitos de las modificaciones y con cuidadoso diseño, implantación, validación y documentación.

Se puede asegurar que las actividades de mantenimiento no destruirán su facilidad de mantenimiento y que un cambio en el código fuente no requerirá modificaciones extremas en el conjunto de pruebas y en los documentos de apoyo, esto debido a que durante el desarrollo del sistema se realizaron las actividades necesarias para facilitar el mantenimiento, estas actividades son:

- Actividades de análisis
  - Se desarrollaron estándares y guías
  - Se identificaron posibles mejoras
  - Se estimó los costos de mantenimiento
- Actividades de diseño
  - Se utilizó la modularidad como criterio de diseño
  - Se diseñó para facilitar posibles mejoras
  - Se utilizaron notaciones estandarizadas para documentar flujos de datos, funciones, estructuras e interconexiones

Se utilizaron los principios de abstracción de datos y descomposición jerárquica hacia abajo

Uso de notaciones estandarizadas para especificar algoritmos, estructuras de datos y procedimientos para especificar interfaces

- Actividades de implementación

  Uso de sangrado estándar en las estructuras

  Estilo de codificación simple y claro

  Uso de constantes simbólicas para asignar parámetros a las rutinas

  Estándares de comentarios internos

- Otras actividades

  Se desarrolló un juego de pruebas

  Se proporcionó al usuario la documentación del juego de pruebas

Todas estas actividades se tienen en los capítulos anteriores del presente documento.

6.3.2. Aspectos Administrativos

El mantenimiento exitoso, como todas las actividades de la ingeniería de software, requiere una combinación de habilidades administrativas y de pericia técnica.

Uno de los aspectos más importantes del mantenimiento del software implica rastrear y controlar las actividades de mantenimiento.

Una modificación al sistema se realiza cuando el usuario del sistema pide a la oficina encargada de la parte técnica del sistema una modificación ,la filosofía o forma de proceder para una modificación de mantenimiento en la oficina encargada de desarrollar SISGIROS es la siguiente:

- Solicitud de modificación de software entregada
- Se analiza la modificación por la persona encargada del sistema en coordinación con el jefe de la oficina
- Si la solicitud no es válida, se comenta con el usuario para indicarle el por que no es válida,

- Si es válida
  - Se evalúa en base a la importancia de la modificación (prioridad), tiempo disponible de la persona que realizará la modificación (esto a que no es el único sistema a su cargo y que además esta en desarrollo de otros sistemas) y restricciones de la oficina (por ejemplo cambios de tipo contable) al determinar que se deben realizar los cambios
  - Se realizan las modificaciones (con su documentación interna)
  - Se realizan las pruebas necesarias
  - Se revisan las pruebas con la colaboración del usuario correspondiente
  - Si no están correctas, se repiten hasta conseguir que lo sean
  - Al estar correctas las pruebas se realiza la modificación a los documentos (documentación técnica y manual del usuario)
  - Se lleva una bitácora con todos los cambios realizados además de la actualización de los documentos necesarios.
  - Se realiza la implantación en la red de producción

6.4 Aplicaciones

Como se describe en todo el trabajo, SISGIROS es la respuesta a una necesidad específica dentro de una empresa, pero la estructura del mismo permite que sea aplicable a diferentes actividades en las cuales sea necesaria la expedición de un giro, entre estas podemos encontrar:

- Elaboración de cheques para pagos de nómina
- Elaboración de giros para pago de nómina a personal dentro y fuera del país.
- Elaboración de cheques para pago de prestaciones al personal
- Elaboración de cheques para pago a proveedores
- Elaboración de cheques para traspaso de fondos
- Elaboración de giros para pagos de servicios en entidades diferentes
- Elaboración de cheques para pago de obligaciones fiscales

Como se puede apreciar, en todas estas actividades, uno de los aspectos principales es la generación de un giro o cheque, pero al mismo tiempo es necesario tener un control sobre los mismos, así como la realización de asientos contables particulares a cada una de las actividades y a las empresas en las cuales se apliquen; para todas ellas se requeriría realizar modificaciones para que SISGIROS se apegue a las necesidades de la actividad específica a la cual sea aplique.

7 Conclusiones

Los Sistemas de Cómputo han sido utilizados desde hace muchos años debido a la necesidad del ser humano de perfeccionar sus técnicas y lograr mayores beneficios.

Los constantes cambios tecnológicos han hecho posible el perfeccionar las técnicas y la realización de nuevos equipos y dispositivos que permitan al hombre realizar ciertas tareas que anteriormente resultaban muy laboriosas, siendo también muy difícil obtener el nivel de calidad, precisión y rapidez requerida.

Poco a poco los sistemas se han ido perfeccionando, para que la interface entre éstos y el ser humano, resulte sencilla y amigable; además de cubrir totalmente los requerimientos existentes.

SISGIROS como sistema cumple con los requerimientos existentes en la empresa y el área para el que fue desarrollado, pero puede tener una aplicación prácticamente en cualquier empresa, para lo cual sería necesario realizar cambios poco significativos a la estructura general del mismo.

La elaboración de giros o cheques es una actividad que tiene una amplia gama de aplicaciones tanto en el ámbito comercial, industrial en todos los niveles y personal, no solo en alguna localidad específica, si no a nivel mundial.

Entre las aplicaciones del sistema de expedición de giros se tienen la elaboración de cheques para pagar:

- Nóminas, dentro o fuera del país
- Prestaciones al personal
- A proveedores de la empresa
- Servicios y obligaciones fiscales

En todas estas actividades, la fase principal es la elaboración del documento, pero se involucran las etapas de control y contabilización del mismo.

Las repercusiones que puede tener el sistema dentro de una empresa son sobresalientes (o notorias), entre ellas se encuentran:

- Reducción significativa en los costos de mantenimiento a equipo mecánico para la elaboración y protección de los giros.
- Disminución drástica en el tiempo de elaboración de los giros, lo cual significa una reducción de gastos.
- Minimización del margen de error al sustituir la operación manual por una automatizada.
- Un alto índice de control de todas las fases de la operación.
- Oportunidad y exactitud en la generación de información.
- Reducción de personal para la elaboración, control y contabilización de las operaciones.

SISGIROS facilita el manejo y control de los giros bancarios expedidos por la empresa, realiza los procesos de:

- Captura de información para la elaboración del documento
- Elaboración del documento
- Impresión de una firma facsímil como medida de seguridad
- Control de las diferentes chequeras que maneja la empresa
- Generación de informes operativos
- Interfaz con el sistema contable de la empresa
- Actualización de la cartera de operaciones
- Generación de informas operativos
- Interfaz con el sistema de conciliaciones de la empresa

Con lo anterior se logró la automatización de la expedición de giros y se implementaron controles seguros para el manejo optimo de los recursos.

Por lo mencionado en los párrafos anteriores estamos seguros que SISGIROS es un sistema que se puede explotar en cualquier empresa que necesite la expedición de cheques y se tendrá un gran control tanto operativo como contable de todas las operaciones que se realicen.

8 Bibliografía

[Btrieve] *Btrieve, Installation and Operation*. USA. Netware Inc.

[Candullo,1985] Candullo, Carl. (1985) *System development standars*.USA. McGraw-Hill Inc.

[Davis, 1983] Davis, William S., (1983).*Systems analysis and design*. USA. Addison-Wesley Publishing Company Inc.

[Fairley,1988] Fairley, Richard E., (1988). *Ingeniería de software*. Naucalpan, Edo de México. McGraw-Hill/Interamericana de México S.A. de C.V.

[Kendall, 1988] Kendall, Kennet E., (1988). *Systems analysis an design*. Englewood Cliffs, New Jersey. Prentice-Hall Inc.

[Novell, 1991] NetWare Versión 3.11. (1991) *Concepts NetWare*. USA. NetWare Inc.

[PASCAL] Turbo Pascal Versión 5.0. *Users Guide, Reference Guide*. USA. Borland Internacional

[PROFESSIONAL] Turbo Professional Versión 5.0. *User's Manual 1,2*. USA. Turbo Power Software

[Shooman,1983] Shooman, Martin L. (1983). *Software engineering*. Singapore. Mc Graw Hill Inc.

[XTRIEVE] Xtrieve. *Xtrieve Interactive Query Manual*. USA. NetWare Inc.

[Yourdon, 1993] Yourdon, Edward. (1993). *Análisis Estructurado Moderno*. Naucalpan, Edo de México. Prentice Hall Hispanoamericana, S.A.

ANEXOS

A.1 Manual del Usuario

INDICE *PAGINA*

INTRODUCCION

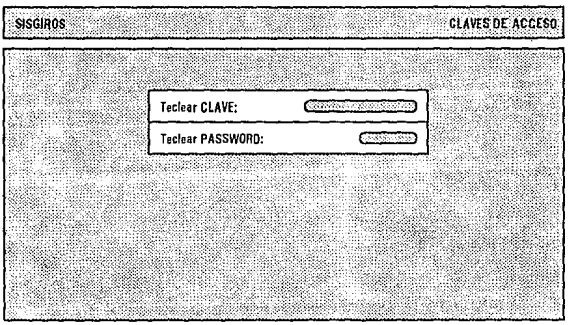
El presente manual es una descripción de la operación del SISGIRO -Sistema de Expedición de Giros-.

El SISGIROS se basa en un sistema de menús, lo cual hace que sea un sistema amigable con el usuario.

A continuación se describe cada uno de los procesos que conforman el sistema, mostrando las diferentes pantallas que SISGIROS utiliza para interactuar con el usuario, así como las acciones que se deben realizar en cada una de las etapas.

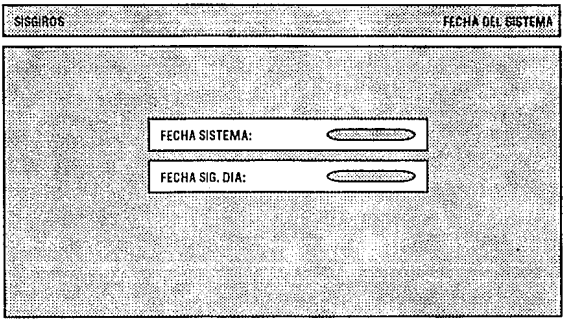
1.0.- ENTRADA AL SISGIROS

Para poder entrar al SISGIROS, el usuario debe de digitar en el directorio raíz del disco duro, (es decir desde C:\), el comando SISGIROS, el cual desplegará la siguiente pantalla:



En este momento SISGIROS solicita al usuario digitar su clave y password; verifica que ambas sean correctas, y si es así permite el acceso al sistema.

Una vez que el usuario ha entrado al sistema, SISGIROS presentará el menú principal, o bien si el usuario entra al inicio del día presentará la siguiente pantalla:



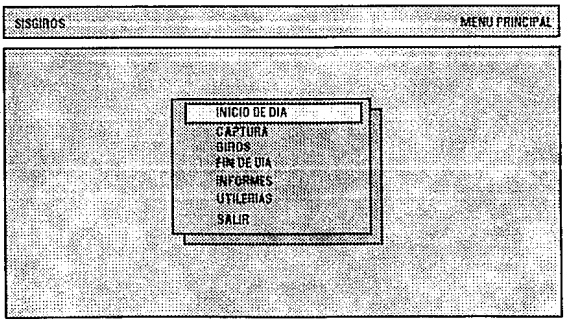
Aquí SISGIROS muestra al usuario la fecha del día anterior de operación y solicita que se digite la nueva fecha del día de operación. Esta nueva fecha debe ser siempre posterior a la fecha que SISGIROS muestra, de lo contrario marcará la fecha digitada como fecha errónea.

Si el usuario se equivoca tres veces, SISGIROS marcará el número de intentos excedidos y no permitirá al usuario proseguir.

Una vez alimentada la fecha, SISGIROS solicitará sea digitada la sig. fecha hábil de operación, siguiendo las mismas restricciones que para la fecha del dia.

Cuando se digitan las fechas correctamente, SISGIROS muestra el menú principal.

2.0.-MENU PRINCIPAL



Proceso de inicio de día

Esta pantalla corresponde al menú principal de SISGIROS, en el que se encuentran las opciones principales que llevan a diferentes submenús o permiten ejecutar algún proceso.

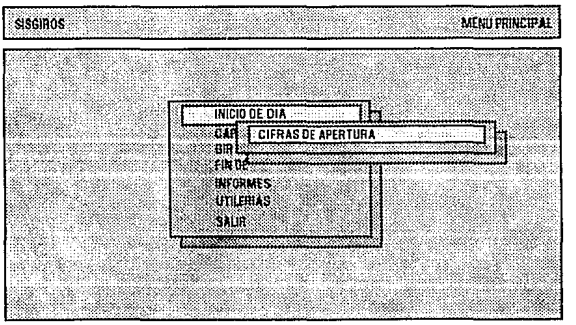
Estas opciones se encuentran relacionadas directamente con el tipo de usuario. Esto es, para el usuario con nivel de operador se encuentran habilitadas las opciones: *INICIO DE DIA*, *CAPTURA*, *FIN DE DIA*, *UTILERIAS* e *INFORMES*.

El usuario con nivel de supervisor tendrá habilitadas las opciones: *INICIO DE DIA*, *GIROS*, *FIN DE DIA*, *UTILERIAS* e *INFORMES* y el usuario con nivel de interventor contará únicamente con la opción de *FIN DE DIA*.

Para seleccionar las opciones, se puede navegar a través del menú con las flechas de dirección, o digitando la letra que aparece en color diferente.

2.1.0.-INICIO DE DIA

Cuando se selecciona esta opción en el menú principal, lleva al usuario a la siguiente pantalla:



Cifras de apertura del sistema

La opción de inicio de día presenta un submenú, en el cual se encuentra el proceso de cifras de apertura.

Este proceso debe ser lanzado siempre al inicio del día, ya que si no se realiza este proceso, SISGIROS no permite la ejecución de ningún otro proceso.

Las cifras de apertura tienen como fin controlar los recursos con los que cuenta el sistema y asegurar la integridad de los mismos; es decir, SISGIROS es un sistema que debe de tener continuidad, las cifras de cierre del día anterior deben coincidir con las cifras de apertura.

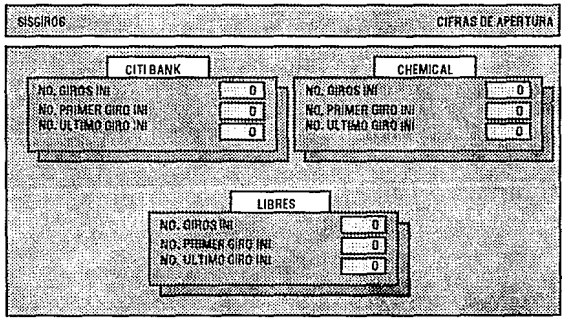
SISGIROS solicitará en el proceso de cifras de apertura tres datos por cada una de las chequeras con que cuenta, estos datos son:

1.-Número de formatos en blanco con que cuenta la chequera.

2.-El número de folio del primer formato en blanco de la chequera.

3.-El número de folio del último formato en blanco de la chequera.

Esta información es solicitada al usuario por medio de la siguiente pantalla de SISGIROS:



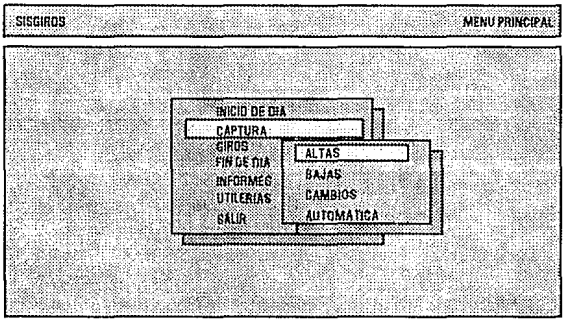
Una vez que el usuario digita la información, SISGIROS verifica que ésta sea correcta y en caso de ser errónea desplegará un mensaje de error y regresará al menú de inicio de día sin permitir que el usuario ejecute otro proceso hasta en tanto no de las cifras de apertura correctas.

2.2.0.- CAPTURA

Esta parte del sistema es donde el usuario podrá alimentar las operaciones que darán origen a los giros.

El usuario puede dar de alta operaciones, hacer modificaciones a operaciones previamente capturadas o dar de baja las operaciones que no hayan sido procesadas.

El submenú que se presenta en esta opción es el siguiente:

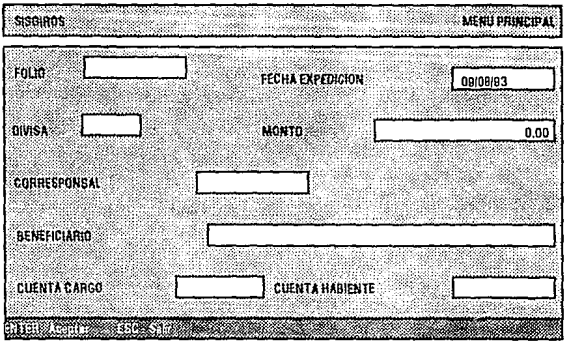


Alta de operaciones de giros

El usuario puede elegir una de las opciones que presenta esta pantalla y pasará directamente a la pantalla que presenta la información de las operaciones, esta pantalla es la misma para las tres opciones, pero en cada una de ellas los campos que la componen tienen diferentes atributos.

El campo por medio del cual se accesa información previamente capturada o se introducen nuevas operaciones, es el número de folio. En este campo se valida que el folio no exista, o si existe que no haya sido procesado, dependiendo de la operación que se desee realizar (alta, baja o modificación).

Cuando se elige la opción de bajas, SISGIROS despliega la información del folio que se desea dar de baja, pero no permite la modificación de ningún otro campo.



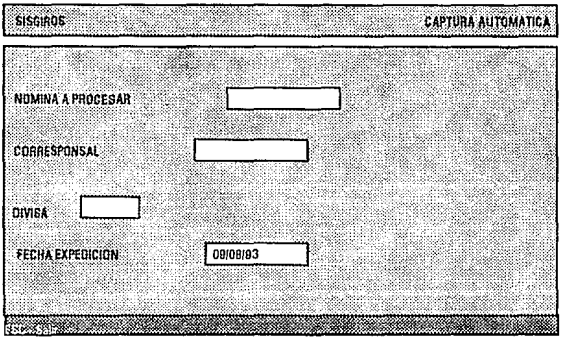
Cuando la opción elegida es la de altas, en cada uno de los campos se realiza una validación de la información capturada; la divisa, el corresponsal y el cuentahabiente deben estar dados de alta en los catálogos correspondientes.

SISGIROS asigna por omisión la fecha del siguiente día hábil de operación a la fecha del giro, ésta puede ser modificada por una fecha posterior, o al menos la fecha de operación del día, lo cual permite que las operaciones sean capturadas en la fecha en que se reciben, pero el proceso se realizará en la fecha que el usuario desee. Esta fecha no puede ser menor a la fecha del sistema.

La opción de captura automática permite al usuario introducir operaciones en lote, esto es, capturar las operaciones que son originadas por nóminas de dependencias.

Para realizar esta captura, el usuario solo debe introducir la nómina que desea procesar, el corresponsal sobre el cual desea expedir los giros, la divisa y la fecha de expedición, con lo cual el sistema genera automáticamente los registros para cada uno de los beneficiarios de la nómina para ser procesados posteriormente.

La pantalla por medio de la cual se genera la captura es la siguiente:



2.3.0.- GIROS

Este es el proceso principal del sistema, tiene como fin la elaboración y protección de los giros que se derivan por cada operación.

Este proceso solamente puede ser ejecutado por el supervisor, el cual es el responsable de los recursos del sistema y del uso de ellos.

Cuando el supervisor elige esta opción, SISGIROS solicitará se introduzca en la unidad A: el disco llave, el cual está en posesión de una persona diferente al supervisor y no tiene acceso a SISGIROS.

Al detectar SISGIROS el disco llave, procede a ejecutar el proceso de impresión de giros; posteriormente solicitará se introduzca el segundo disco llave, el cual se requiere para poder configurar la impresora.

Cuando SISGIROS ha configurado la impresora, solicitará al usuario indicar el tipo de formatos que serán alimentados (en forma manual o continua).

SISGIROS realiza la impresión de los giros en tres ciclos - uno para cada chequera -, es posible generar los giros de una sola chequera presionando la tecla ESC (Escape) cuando solicita que se introduzcan los formatos correspondiente a las chequeras de las cuales no se desea imprimir giros.

En el momento en que SISGIROS imprime cada giro, verifica que existan formatos en limpio para realizar la impresión, en caso de que se agoten los formatos en limpio de que dispone, el supervisor deberá salir de esta opción y entrar a la opción de modificación de control (la cual se describe posteriormente).

Una vez que SISGIROS terminó de imprimir todos los giros, es necesario que el segundo disco llave permanezca en la unidad A: y que la impresora se encuentre en

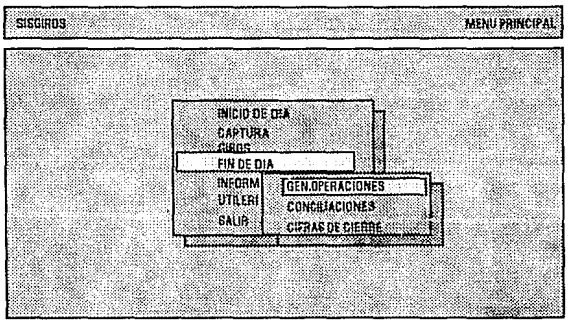
línea para que proceda a configurarla a su estado original. Si la impresora se encuentra fuera de línea no será posible terminar con este proceso.

Una vez que termina el proceso, SISGIROS regresa al menú principal, en este momento se deberá retirar el disco llave de la unidad A:.

2.4.0.- FIN DE DIA

Esta opción lleva a un submenú en el cual se encuentran los procesos que deben ser ejecutados al final de un día de operación, y una vez ejecutados SISGIROS no permite que se vuelvan a ejecutar el mismo día.

El menú al cual lleva SISGIROS es el siguiente:



Generación de Ops. Contables

2.4.1.- GENERACION DE OPERACIONES

Con esta opción se ejecuta el proceso con el cual SISGIROS genera las operaciones para realizar la contabilización a través del sistema contable de la empresa.

Este proceso genera las operaciones de los giros que son generados en el día y debido a esto, una vez realizado, SISGIROS no permite que se vuelva a lanzar el proceso de generación de giros.

2.4.2.- ACTUALIZA CONCILIACIONES

Este proceso es la interfase entre SISGIROS y el sistema de conciliaciones, con lo cual se tiene la información oportuna de la expedición de giros, sin necesidad de que la información sea capturada nuevamente.

Con esta información, se alimenta al sistema de conciliaciones para actualizar las bases de datos correspondientes.

2.4.3.- CIFRAS DE CIERRE

Este es el último proceso que debe ejecutarse diariamente y tiene como fin verificar que las cifras internas del SISGIROS -las cuales son confidenciales- coincidan con los datos que el personal de la oficina usuaria ha calculado.

La opción de cifras de cierre se encuentra habilitada sólo para el nivel de Interventor, siendo éste quien debe lanzar la ejecución de este proceso alimentando los datos que fueron calculados manual y previamente.

SISGIROS solicitará, al igual que en cifras de apertura, los datos en forma independiente para cada una de las chequeras. Los datos son los siguientes:

1.- Número de formatos utilizados.

2.- Número de folio del primer formato utilizado.

3.- Número de folio del último giro utilizado.

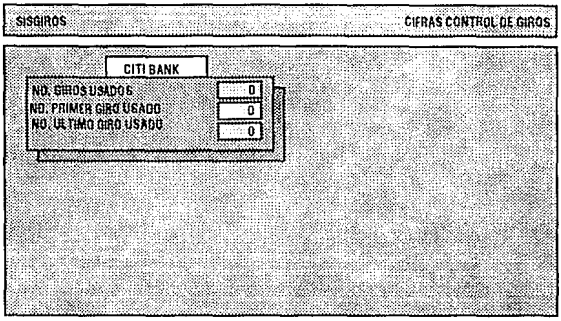
4.- Número de formatos limpios al final del día.

5.- Número de folio del primer formato en limpio.

6.- Número de folio del último formato en limpio.

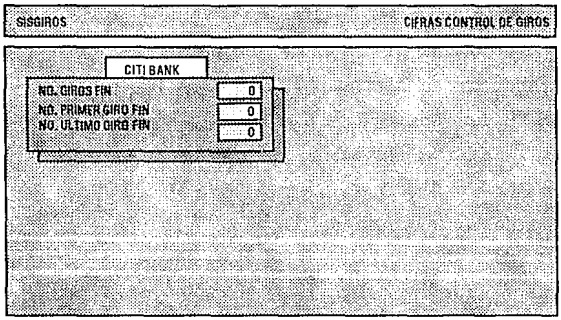
7.- Número de formatos cancelados.

SISGIROS utiliza tres pantallas para solicitar la información de cada chequera, la primera se muestra a continuación:

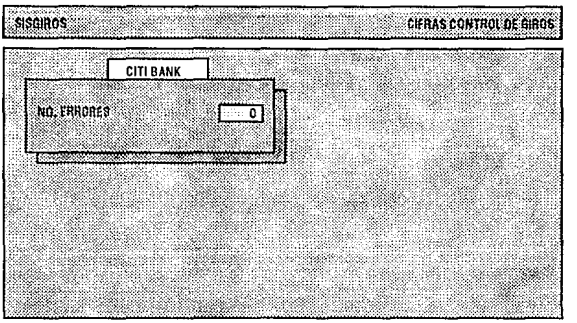


Esta pantalla es para los datos de formatos usados.

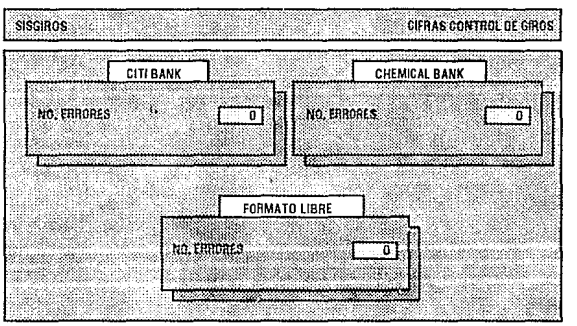
La siguiente pantalla es para los formatos sin usar al final del día.



Y la pantalla para giros cancelados es la siguiente:



Estas pantallas se presentan para cada chequera, quedando finalmente la siguiente pantalla:



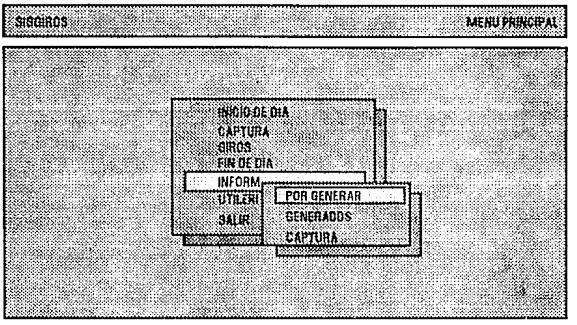
Cuando la información alimentada a SISGIROS es correcta, se genera un reporte con las cifras de control (apertura y cierre), además del reporte de giros generados y de giros cancelados, para que el interventor tenga la información necesaria.

En caso de no ser correctas estas cifras, el supervisor deberá de realizar una investigación para verificar las cifras y poder detectar en donde se encuentra la diferencia.

Una vez que se ha ejecutado el proceso y las cifras fueron correctas, SISGIROS da por terminado el día de operación (se ha cerrado el sistema), y por lo tanto al querer volver entrar a SISGIROS pedirá que se inicie otro día.

2.5.0.- INFORMES

Esta opción nos lleva a un submenú de informes cuya pantalla es la siguiente:

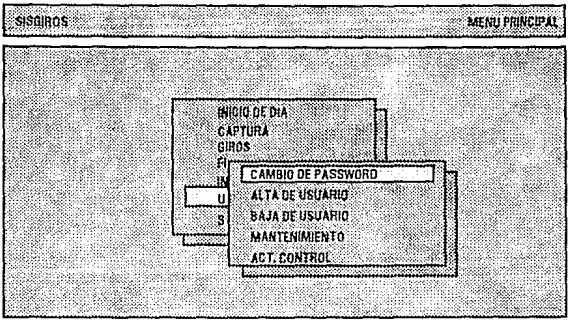


Reporte de Giros por generar del día

Este submenú contiene los informes solicitados por OTOCA, al seleccionar una de estas opciones se ejecuta el proceso para generar el informe deseado; actualmente existen cuatro reportes: giros por generarse, giros generados, giros capturados el día de operación y giros capturados anteriormente sin generar.

2.6.0- UTILERIAS

Esta opción nos lleva a un submenú que contiene las diferentes utilerías para trabajar con SISGIROS, el submenú es el siguiente:



Cambiar el password del Usuario

2.6.1.- CAMBIO DE PASSWORD

Esta opción permite al usuario modificar la palabra de acceso a SISGIROS (password), en el momento en que desee.

Para esto SISGIROS solicitará sea digitado el password anterior, y posteriormente pedirá el nuevo password dos veces para verificarlo.

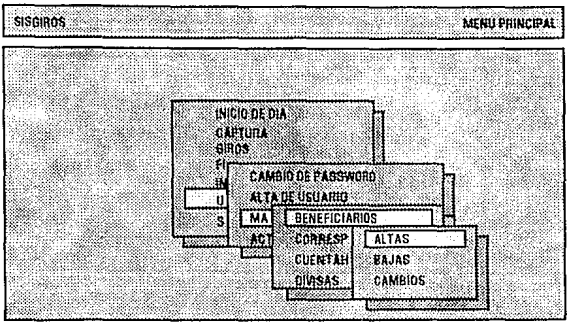
Esta opción se encuentra habilitada para todos los usuarios, y solamente se podrá cambiar el password del usuario que se encuentra en sesión con SISGIROS.

2.6.2.- ALTA DE USUARIO

Esta opción se encuentra habilitada únicamente para el supervisor, y tiene como fin que se puedan habilitar más usuarios para trabajar con SISGIROS. Los usuarios que se den de alta sólo podrán tener el nivel de operación o de supervisión.

2.6.3.- MANTENIMIENTO

Esta opción permite dar mantenimiento a los diferentes archivos utilizados por SISGIROS, al seleccionarla lleva al usuario al siguiente submenú:



Alta al archivo de Beneficiarios

Como se puede observar, en este submenú cada opción lleva a un segundo submenú que permite al usuario indicar el tipo de movimiento que desea hacer. Este submenú se presenta en cada uno de los diferentes archivos -sólo el operador puede accesar esta opción-.

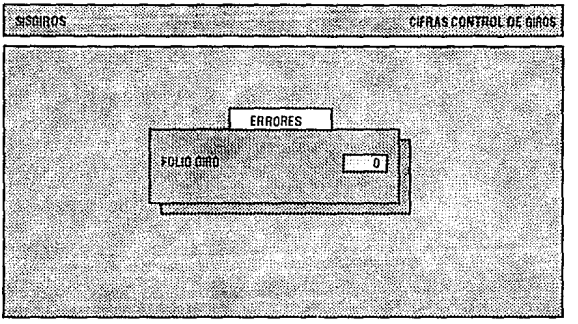
2.6.4.- ACTUALIZA CONTROL

Esta opción se encuentra habilitada únicamente para el supervisor, por medio de ésta se puede realizar la cancelación de los giros que fueron expedidos y es necesario cancelarlos; también se puede realizar la modificación de las cifras de control.

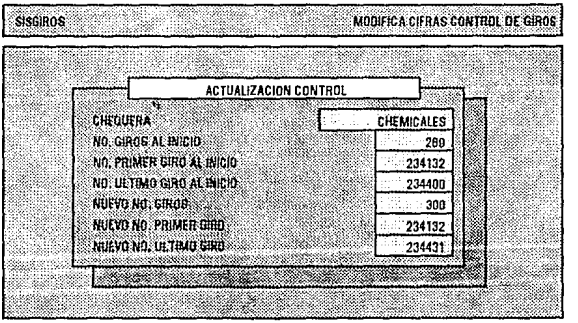
Para realizar las cancelaciones, SISGIROS solicita el folio de la operación que se desea cancelar, así como el número de folio del formato que se utilizó y el monto por el cual se expidió el giro, además del motivo por el cual se desea realizar la cancelación.

SISGIROS realiza las validaciones para la cancelación y, en caso de que los datos que se digitaron sean correctos, procede a realizar la cancelación.

La siguiente es la primer pantalla que presenta SISGIROS al usuario para realizar la cancelación:



Para realizar la modificación de las cifras de control, SISGIROS presenta la siguiente pantalla:



En esta pantalla se muestran los datos que deben ser alimentados, SISGIROS realiza las validaciones necesarias, y si coinciden las cifras se realiza la actualización.

SISGIROS emite un reporte, para indicar cuales fueron los cambios en las cifras de control.

A.2 Listados de los Programas

En esta sección se presentan los programas que conforman al SISGIROS, estan organizados en base a los procesos del menú del sistema y al final se tienen los programas de los módulos con funciones y procedimientos generales del sistema y los de manejo de archivos.

Menú principal del sistema:

```
PROGRAM MENUGIRO;
{
Ejecuta el programa que despliega el menú principal del sistema y
dependiendo de la opción indicada manda a ejecutar al programa o proceso
correspondiente
}

USES
   TPMENU,              { manejo de menus }
   UCAPTURA,            { captura de datos }
   UIMPRESO,            { rutinas de impresion }
   UMSJGIRO,            { mensajes de procesos }
   UPASSWOR,            { acceso al sistema }
   TPCRT,               { manejo de teclado, pantalla, colores y sonido }
   TPSTRING,            { manejo de strings }
   TPEDIT,              { edicion de datos }
   COLORDEF,            { definicion de colores para pantallas }
   UDCLGIRO,            { manejo de declaraciones globales }
   UGRALGIR,            { rutinas generales }
   URUTGIRO,            { rutinas generales }
   UPROGIR;             { manejo de constantes del archivo de control }

CONST
   ENTER = ^M;
   ESC   = ^[;

VAR
   clave_usuario        : tpo_operador;
   grupo_usuario        : tpo_grupo_usuarios;
   movimiento,
   opcion,
   caracter_salida      : char;
   menu_giro            : menu;
   key_giro             : menukey;
   stack_menu_giro      : menustackp;
   estado_proceso,
   op_default           : integer;
   fecha_sistema        : tpo_fecha;
   param1,
   param2               : string;
   borra_menu           : boolean;
   impresora            : tpo_impresora;

{$I f:\giros\fuentes\incluye\menu_gir.pas}

BEGIN        { BEGIN MENUGIRO   procedimiento principal }
   param1 := paramstr (1);
```

```
param2 := paramstr (2);

if Param1 = '' then
   begin
   EJECUTA('f:\giros\sistema\logo.exe','S. I. E. G. S. E.', '',
            proceso_dummy, no, no);
   PIDE_CLAVE(clave_usuario,grupo_usuario);
   if REALIZADO(cifras_cierre,'cifras_cierre(sigse)') then
      PIDE_FECHA_SISTEMA;
   end;

ENCABEZADO_PANTALLA('MENU PRINCIPAL');
HiddenCursor;
InitMenu(menu_giro);
DESHABILITA (menu_giro);

if (not Str2Int(Param1, op_default)) then
   op_default := inicio_dia
else
   SelectMenuItem(menu_giro, op_default);

repeat
   if not (Str2Int(param2, op_default)) then
      begin
      key_giro:= MenuChoice(menu_giro,caracter_salida);
      borra_menu := si;
      end
   else
      begin
      key_giro        := MenuKey (op_default);
      caracter_salida := #13;
      borra_menu      := no;
      end;

   if caracter_salida = #13 then
      begin
      {guarda estado del menu}
      EraseMenuOntoStack(menu_giro,stack_menu_giro);

      case key_giro of
         cifras_apertura:
            if NOT REALIZADO(cifras_apertura, 'CIF_APER') then
                 EJECUTA('f:\giros\sistema\cif_aper.exe','','CIFRAS DE
                          APERTURA',
                      proceso_dummy, false, false)
            else
               MENSAJE('ERROR', 'PROCESO REALIZADO', 21, error_usuario);


         altas_giros,
         cambios_giros,
         bajas_giros :
            begin
            case key_giro of
               altas_giros       : movimiento:= altas;
               cambios_giros     : movimiento:= cambios;
               bajas_giros       : movimiento:= bajas;
               end;{end case}
            if REALIZADO(cifras_apertura,'CIFRAS APERTURA') then
               CAPTURA_GIROS(movimiento)
            else
               MENSAJE('ERROR','REALIZAR CIFRAS DE INICIO',
                        21,error_usuario);
            end;

         captura_automatica_giros :
            if  REALIZADO(cifras_apertura, 'CIFRAS APERTURA') then
                EJECUTA('\giros\sistema\cap_aut.exe', '', 'CAPTURA
                        AUTOMATICA',captura_automatica_giros, no, no)
            else
               MENSAJE('ERROR', 'REALIZAR CIFRAS DE INICIO', 21,
                        error_usuario);

         altas_benef,
         cambios_benef,
```

```
bajas_benef:
   begin
   case key_giro of
      altas_benef       : movimiento:= altas;
      cambios_benef : movimiento:= cambios;
      bajas_benef       : movimiento:= bajas;
      end;
   if REALIZADO(cifras_apertura,'CIFRAS APERTURA') then
      CAPTURA_BENEFICIARIOS(movimiento)
   else
      MENSAJE('ERROR','REALIZAR CIFRAS DE INICIO', 21,
                  error_usuario);
   end;

altas_corresp,
cambios_corresp,
bajas_corresp:
   begin
   case key_giro of
      altas_corresp     : movimiento:= altas;
      cambios_corresp   : movimiento:= cambios;
      bajas_corresp     : movimiento:= bajas;
      end;{end case}
   if REALIZADO(cifras_apertura, 'CIF. APERTURA') then
      CAPTURA_CORRESPONSAL(movimiento)
   else
      MENSAJE('ERROR', 'REALIZAR CIFRAS DE APERTURA', 21,
                  error_usuario);
   end;

altas_ctahabte,
cambios_ctahabte,
bajas_ctahabte:
   begin
   case key_giro of
      altas_ctahabte          : movimiento:= altas;
      cambios_ctahabte        : movimiento:= cambios;
      bajas_ctahabte          : movimiento:= bajas;
      end;{end case}
   if REALIZADO(cifras_apertura, 'CIF. APERTURA') then
      CAPTURA_CTAHABTE(movimiento)
 else
      MENSAJE('ERROR', 'REALIZAR CIFRAS DE APERTURA', 21,
                  error_usuario);
   end;

altas_divisas,
cambios_divisas,
bajas_divisas:
   begin
   case key_giro of
      altas_divisas     : movimiento:= altas;
      cambios_divisas   : movimiento:= cambios;
      bajas_divisas     : movimiento:= bajas;
      end;{end case}
   if REALIZADO(cifras_apertura, 'CIF. APERTURA') then
      CAPTURA_DIVISA(movimiento)
   else
      MENSAJE('ERROR', 'REALIZAR CIFRAS DE APERTURA', 21,
                  error_usuario);
   end;

generacion_giros :
   if REALIZADO(cifras_apertura, 'CIF. APERTURA') then
     if not REALIZADO(formatos_con, 'FORMATOS CONT.') then
        begin
        ReadCharacter (Pad(' INSERTAR EL DISCO LLAVE. ENTER-
                           Continuar ESC-Salir', 80),25,1,
                           BlackOnLtGray, [ENTER, ESC],
                           opcion);
        if opcion = ENTER then
           EJECUTA('A:\giros\llave\giros.exe','','GENERACION DE
                       GIROS',  proceso_dummy,true,
                           false);
        end
```

```
      else
         MENSAJE('ERROR', 'SE ESTA REALIZANDO EL FIN DE DIA',
                  21, error_usuario)
   else
      MENSAJE('ERROR', 'REALIZAR CIFRAS DE APERTURA', 21,
               error_usuario);

formatos_con:
      if REALIZADO(generacion_giros, 'GEN. GIROS') then
         if not REALIZADO(formatos_con, 'FORMATOS CONT.')then
            begin
            EJECUTA('f:\giros\sistema\gen_form.exe', '2',
                     'GENERACION DE OPS.CONTABLES',
                     proceso_dummy, false,false);
            EJECUTA('f:\giros\sistema\gen_ctas.exe', '3',
                     'GENERACION DE OPS. CONTABLES',
                     proceso_dummy, false, false);
             end
         else
            MENSAJE('ERROR', 'PROCESO REALIZADO', 21,
                     error_usuario)
      else
         MENSAJE('ERROR', 'REALIZAR GIROS', 21,
                     error_usuario);

reporte_generacion :
   if REALIZADO(cifras_apertura, 'CIF. APERTURA') then
      EJECUTA('f:\giros\sistema\rep_gen.exe', 'N', 'REPORTE DE
               GIROS POR GENERAR',proceso_dummy, no, no)
   else
      MENSAJE('ERROR','REALIZAR CIFRAS DE APERTURA', 21,
               error_usuario);

reporte_generados:
   if REALIZADO(cifras_apertura, 'CIF. APERTURA') then
      EJECUTA('f:\giros\sistema\rep_gen.exe', 'G', 'REPORTE DE
               GIROS GENERADOS', proceso_dummy, no, no)
   else
      MENSAJE('ERROR','REALIZAR CIFRAS DE APERTURA', 21,
               error_usuario);

reporte_captura:
   if REALIZADO(cifras_apertura, 'CIF. APERTURA') then
      EJECUTA('f:\giros\sistema\rep_cap.exe','', 'REPORTE DE
               GIROS CAPTURADOS', proceso_dummy, no, no)
   else
      MENSAJE('ERROR','REALIZAR CIFRAS DE APERTURA', 21,
               error_usuario);

cambio_password :
   CAMBIO_PASSWORD_SIGSE;

altas_usuarios :
   ALTA_CLAVE(grupo_usuario);

bajas_usuarios :
   BAJA_CLAVE(grupo_usuario);

cancela_giros :
   if REALIZADO(cifras_apertura, 'CIF. APERTURA') then
      if REALIZADO(generacion_giros, 'GEN. GIROS') then
         EJECUTA('f:\giros\sistema\mod_ctrl.exe', '',
                  'CANCELA GIROS', proceso_dummy, false,
                  false)
      else
         MENSAJE('ERROR', 'REALIZAR GIROS', 21,
                  error_usuario)
   else
      MENSAJE('ERROR', 'REALIZAR CIFRAS APERTURA', 21,
                  error_usuario);

cifras_control:
   if REALIZADO(cifras_apertura, 'CIF. APERTURA') then
      EJECUTA('f:\giros\sistema\mod_num.exe', '', 'CIFRAS
               CONTROL', proceso_dummy, false, false)
```

```
                else
                   MENSAJE('ERROR', 'REALIZAR CIFRAS APERTURA', 21,
                           error_usuario);

             cifras_cierre :
                if NOT REALIZADO(cifras_cierre, 'CIF. CIERRE') then
                   if REALIZADO(cifras_apertura, 'CIF. APERTURA') then
                      begin
                      EJECUTA('f:\giros\sistema\cif_fin.exe','','CIFRAS DE
                              CIERRE',proceso_dummy, false,
                                      false);
                      if REALIZADO(cifras_cierre, 'CIF. CIERRE') then
                         begin
                         EJECUTA('f:\giros\sistema\rep_ctrl.exe', '',
                                 'REPORTE DE CONTROL', proceso_dummy,
                                 false, false);
                         EJECUTA('f:\giros\sistema\rep_gen.exe', 'G',
                                 'REPORTE DE GIROS GENERADOS',
                                 proceso_dummy, false, false);
                         EJECUTA('f:\giros\sistema\rep_err.exe', '',
                                 'REPORTE DE ERRORES', proceso_dummy,
                                 false, false);
                         end;
                      end.
                   else
                      MENSAJE('ERROR', 'REALIZAR GENERACION DE GIROS', 21,
                                 error_usuario)
                else
                   MENSAJE('ERROR', 'PROCESO REALIZADO', 21, error_usuario);

             reporte_benef:
                   if REALIZADO(cifras_apertura, 'CIF. APERTURA') then
                      EJECUTA('f:\giros\sistema\rep_ben.exe', '', 'REPORTE DE
                              BENEFICIARIOS', proceso_dummy, false, false)
                   else
                      MENSAJE('ERROR', 'REALIZAR CIFRAS APERTURA', 21,
                              error_usuario);

             reporte_corresp:
                   if REALIZADO(cifras_apertura, 'CIF. APERTURA') then
                      EJECUTA('f:\giros\sistema\rep_corr.exe', '', 'REPORTE
                              DE CORRESPONSALES', proceso_dummy, false,
                              false)
                   else
                      MENSAJE('ERROR', 'REALIZAR CIFRAS APERTURA', 21,
                              error_usuario);


             reporte_ctahabte:
                   if REALIZADO(cifras_apertura, 'CIF. APERTURA') then
                      EJECUTA('f:\giros\sistema\rep_ctah.exe', '', 'REPORTE
                              DE CUENTAHABIENTES', proceso_dummy, false,
                              false)
                   else
                      MENSAJE('ERROR', 'REALIZAR CIFRAS APERTURA', 21,
                              error_usuario);


             reporte_divisas:
                   if REALIZADO(cifras_apertura, 'CIF. APERTURA') then
                      EJECUTA('f:\giros\sistema\rep_div.exe', '', 'REPORTE DE
                              DIVISAS', proceso_dummy, false, false)
                   else
                      MENSAJE('ERROR', 'REALIZAR CIFRAS APERTURA', 21,
                              error_usuario);
             end;{CASE}

          { Se reestablece la pantalla }
          ENCABEZADO_PANTALLA('MENU PRINCIPAL');
          DrawMenuFromStack(menu_giro,stack_menu_giro);

          param1 := '';
          param2 := '';
          end;
       UNTIL (key_giro = salir);
```

```
    ENCABEZADO_PANTALLA('MENU PRINCIPAL');
    NormalCursor;
END.    { MENUGIRO }
```

Programa para desplegar el menú:

```
{MENUGIR.PAS}
PROCEDURE InitMenu(var M : Menu);
{ menu del sistema }
CONST
  Color1 : MenuColorArray = ($79, $79, $7F, $1F, $74, $0E, $78, $38);
  Frame1 : FrameArray = '++++-¦';

Begin
  M := NewMenu([], nil);
  SubMenu(28,9,ScreenHeight,Vertical,Frame1,Color1,'');
    MenuMode(True, True, False);
    MenuWidth(23);
    MenuItem('INICIO DE DIA',1,14,1000,'Proceso de inicio de día');
    SubMenu(39,11,ScreenHeight,Vertical,Frame1,Color1,'');
      MenuMode(True, True, False);
      MenuItem('CIFRAS DE APERTURA',1,1,1100,'Cifras de Apertura del
            Sistema');
      PopSublevel;
    MenuItem('CAPTURA',2,4,2000,'Captura de operaciones de Giros');
    SubMenu(39,11,ScreenHeight,Vertical,Frame1,Color1,'');
      MenuMode(True, True, False);
      MenuWidth(15);
      MenuItem('ALTAS',1,1,2100,'Alta de operaciones de giros');
      MenuItem('BAJAS',2,1,2200,'Baja de operaciones de Giros');
      MenuItem('CAMBIOS',3,1,2300,'Cambio de operaciones de Giros');
      MenuItem('AUTOMATICA',4,3,2400,'Captura Automática de Operaciones de
            Giros');
      PopSublevel;
    MenuItem('GIROS',3,4,3000,' Generación de los Giros');
    MenuItem('FIN DE DIA',4,4,4000,'Proceso de Fin de Día');
    SubMenu(39,13,ScreenHeight,Vertical,Frame1,Color1,'');
      MenuMode(True, True, False);
      MenuWidth(23);
      MenuItem('GEN. FORMATOS',1,1,4100,'Generación de Ops. para
            Contabilización');
      MenuItem('CONCILIACIONES',2,1,4200,'Generación de información para
            Conciliaciones');
      MenuItem('CIFRAS DE CIERRE',3,8,4300,'Cifras de Control para Fin de
            Día');
      PopSublevel;
    MenuItem(' INFORMES',5,4,5000,'Informes operativos del sistema');
    SubMenu(39,14,ScreenHeight,Vertical,Frame1,Color1,'');
      MenuMode(True, True, False);
      MenuWidth(19);
      MenuItem('POR GENERAR',1,1,5100,'Reporte de Giros por generar del
            día');
      MenuItem('GENERADOS',2,1,5200,'Reporte de Giros Generados del día');
      MenuItem('CAPTURA',3,1,5300,'Reporte de Captura');
      PopSublevel;
    MenuItem('UTILERIAS',6,4,6000,'Herramientas para el uso de SIGSE ');
    SubMenu(36,13,ScreenHeight,Vertical,Frame1,Color1,'');
      MenuMode(True, True, False);
      MenuWidth(23);
      MenuItem('CAMBIO DE PASSWORD',1,11,6100,'Cambiar el Password del
            Usuario');
      MenuItem('USUARIOS',2,1,6200,'Altas y Bajas de Usuarios');
      SubMenu(45,14,ScreenHeight,Vertical,Frame1,Color1,'');
        MenuMode(True, True, False);
        MenuItem(' ALTAS',1,2,6210,'Altas a usuarios del SISGIROS');
        MenuItem(' BAJAS ',2,2,6220,'Bajas para usuarios del SISGIROS');
        PopSublevel;
      MenuItem('MANTENIMIENTO',3,1,6300,'Mantenimiento a Catalogos');
      SubMenu(44,15,ScreenHeight,Vertical,Frame1,Color1,'');
        MenuMode(True, True, False);
        MenuItem('BENEFICIARIOS',1,1,6310,'Mantenimiento a archivo de
            Beneficiarios');
```

```
        SubMenu(53,16,ScreenHeight,Vertical,Frame1,Color1,'');
          MenuMode(True, True, False);
          MenuItem('ALTAS',1,1,6311,'Altas al archivo de Beneficiarios');
          MenuItem('BAJAS',2,1,6312,'Bajas del archivo de Beneficiarios');
          MenuItem('CAMBIOS',3,1,6313,'Cambios al archivo de
              Beneficiarios');
          MenuItem('REPORTE',4,1,6314,'Reporte del archivo de
              Beneficiarios');
          PopSublevel;
        MenuItem('CORRESPONSALES',2,1,6320,'Mantenimiento al archivo de
              Corresponsales');
        SubMenu(53,16,ScreenHeight,Vertical,Frame1,Color1,'');
          MenuMode(True, True, False);
          MenuItem('ALTAS',1,1,6321,'Altas al archivo de Corresponsales');
          MenuItem('BAJAS',2,1,6322,'Bajas del archivo de Corresponsales');
          MenuItem('CAMBIOS',3,1,6323,'Cambios al archivo de
              Corresponsales');
          MenuItem('REPORTE',4,1,6324,'Reporte del archivo de
              Corresponsales');
          PopSublevel;
        MenuItem('CUENTAHABIENTES',3,2,6330,'Mantenimiento al archivo de
              Cuentahabientes');
        SubMenu(53,16,ScreenHeight,Vertical,Frame1,Color1,'');
          MenuMode(True, True, False);
          MenuItem('ALTAS',1,1,6331,'Altas al archivo de Cuentahabientes');
          MenuItem('BAJAS',2,1,6332,'Bajas del archivo de
              Cuentahabientes');
          MenuItem('CAMBIOS',3,1,6333,'Cambios al archivo de
              Cuentahabientes');
          MenuItem('REPORTE',4,1,6334,'Reporte del archivo de
              Cuentahabientes');
          PopSublevel;
        MenuItem('DIVISAS',4,1,6340,'Mantenimiento al archivo de Divisas');
        SubMenu(53,16,ScreenHeight,Vertical,Frame1,Color1,'');
          MenuMode(True, True, False);
          MenuItem('ALTAS',1,1,6341,'Altas al archivo de Divisas');
          MenuItem('BAJAS',2,1,6342,'Bajas al archivo de Divisas');
          MenuItem('CAMBIOS',3,1,6343,'Cambios al archivo de Divisas');
          MenuItem('REPORTE',4,1,6344,'Reporte del archivo de Divisas');
          PopSublevel;
        PopSublevel;
      MenuItem('ACT. CONTROL',4,6,6400,'Actualización de Datos de
              Control');
      SubMenu(46,17,ScreenHeight,Vertical,Frame1,Color1,'');
        MenuMode(True, True, False);
        MenuItem('CANCELACIONES',1,1,6410,'Cancelación de giros
              expedidos');
        MenuItem('CIFRAS',2,3,6420,'Cifras de control de Chequeras');
        PopSublevel;
      PopSublevel;
    MenuItem('SALIR',7,4,7000,'Salida del Sistema');
    PopSublevel;
  ResetMenu(M);
  SetMenuDelay(M, 20);
end;
```

Programa para cifras de apertura del sistema:

```
{$I-,V-,R-,N+,E+}
PROGRAM CIF_APER;
{ Proceso para validar las cifras de apertura del sistema }

USES
   TPCRT,       { manejo de teclado, pantalla, colores y sonido }
   TPSTRING,    { manejo de strings }
   TPEDIT,      { edicion de datos }
   TPENTRY,     { pantallas de captura }
   TPDATE,      { manejo de fechas }
   COLORDEF,    { definicion de colores para pantallas }
   PRINTER,     { manejo de la impresora}
   UBTRIEVE,    { manejo de archivos }
   UCONTROL,    { manejo del archivo de control del sistema }
   UDCLGIRO,    { manejo de declaraciones globales }
   UGRALGIR,    { rutinas generales }
   UMSJGIRO,    { rutinas de mensajes a pantalla }
   TPWINDOW,    { manejo de ventanas }
   ULECTURA,    { captura de datos en formato estandar }
   UNUMGIR,     { manejo del archivo de cifras de control del sistema }
   UPROGIR;     { manejo de constantes del archivo de control }

CONST
  numero_cifras_contar = 5;
  max_intentos         = 3;
  color_ventana        = BlackonCyan;
  color_marco          = BlackonCyan;
  color_titulo         = CyanonBlack;
  color_sombra         = CyanonGreen;
  color_prompt         = BlackonCyan;
  color_captura        = WhiteonBlack;
  color_seleccion      = YellowonBlack;

VAR
  wt_num_gir                  : r_num_gir;
  primer_giro,
  ultimo_giro                 : tpo_folio;
  numero_giros_pregunta,
  primer_giro_pregunta,
  ultimo_giro_pregunta        : string;
  numero_giros_citi,
  p_giro,
  u_giro,
  numero_giros                : longint;
  status_io,
  cta_intentos,
  recorre,
  wtu                         : integer;
  llave_num_gir               : llavel_num_gir;
  proceso                     : longint;
  numero_str                  : string;
  ok_citibank,
  ok_chemical,
  ok_libre,
  presn_escape                : boolean;
  citiwin,
  chemicalwin,
  librewin                    : windowptr;

  PROCEDURE INICIALIZA;
  { inicializa variables y ventanas }
    begin
    explode := true;
    shadow := true;
    SoundFlagW := true;
    ShadowAttr := color_sombra;
    if not makewindow(citiwin, 4, 7, 33, 14, true, true, true,
                                color_ventana, color_marco, color_titulo,
                                'CITI BANK')
    then  DESPLIEGA_ERROR('ERROR EN INICIALIZA VENTANA', 8);
```

```
    if not makewindow(chemicalwin, 45, 7, 74, 14, true, true, true,
                      color_ventana, color_marco, color_titulo, 'CHEMICAL
                      BANK')
    then  DESPLIEGA_ERROR('ERROR EN INICIALIZA VENTANA', 8);

   if not makewindow(librewin, 26, 16, 55, 23, true, true, true,
                      color_ventana, color_marco, color_titulo, 'FORMATO
                      LIBRE')
    then  DESPLIEGA_ERROR('ERROR EN INICIALIZA VENTANA', 8);
    end;  {end inicializa}

PROCEDURE LIBERA_VENTANAS;
  { borra y libera espacio de ventanas }
    begin
    DisposeWindow(citiwin);
    DisposeWindow(Chemicalwin);
    DisposeWindow(librewin);
    end; {end libera_ventanas}


PROCEDURE LEE_DATOS(cadena : string; x, y: byte);
  { captura las cifras }
    begin
    SetFieldAttr(color_captura);
    SetPromptAttr(color_prompt);
    SetStringAttr(color_seleccion);
    numero_giros_pregunta:= copy(blancos, 1, numero_cifras_contar);
    EditString('NO. GIROS ' + cadena + ' : ',y, x, numero_cifras_contar,
           '99999', 0, press_escape, numero_giros_pregunta);
    primer_giro_pregunta:= copy(blancos, 1, sizeof(tpo_folio));
    EditString('NO. PRIMER GIRO '+ cadena + ' : ', y + 2, x,
           sizeof(tpo_folio), '99999999', 0, escape,
           primer_giro_pregunta);
    primer_giro_pregunta := Leftpadch(primer_giro_pregunta, '0',
                                      sizeof(tpo_folio));
    ultimo_giro_pregunta:= copy(blancos, 1, sizeof(tpo_folio));
    EditString('NO. ULTIMO GIRO ' + cadena + ' : ', y + 4, x,
           sizeof(tpo_folio),'99999999', 0 , ress_escape,
           ultimo_giro_pregunta);
    ultimo_giro_pregunta := Leftpadch(ultimo_giro_pregunta, '0',
                                      sizeof(tpo_folio));
    end; {end LEE_DATOS}

FUNCTION CHECA_DATOS(banco : string): boolean;
  { Valida los datos capturados con los calculados }
    VAR
      dummy : boolean;
    begin
    val(numero_giros_pregunta, numero_giros,status_io);
    if str2long(ultimo_giro, u_giro) and str2long(primer_giro, p_giro)then
       if (primer_giro <> primer_giro_pregunta) or
          (ultimo_giro <> ultimo_giro_pregunta) or
          (numero_giros <> (u_giro - p_giro + 1)) then
            dummy := false
        else
            dummy := true
    else
       DESPLIEGA_ERROR('CIFRAS CONTROL, NUMEROS GIROS '+ banco,
                       status_io);
    if not dummy then
       if (primer_giro = '00000000') and (ultimo_giro = '00000000') then
          if str2long(ultimo_giro_pregunta, u_giro) and
            str2long(primer_giro_pregunta, p_giro)
            and ((numero_giros = (u_giro - p_giro + 1)) or
                 (numero_giros = 0)                      )  then
            dummy := true;
    CHECA_DATOS := dummy;
   end; {end checa_datos}

PROCEDURE AFECTA_NUM_GIR(ident1, ident2: longint; num1, num2: tpo_folio;
                         proceso_aux, proceso_aux2: longint);
  { actualiza las cifras }
    begin
    LEE1_NUM_GIR(wr_num_gir, ident1, status_io);
    if status_io <> ok_io then
```

```
            DESPLIEGA_ERROR('CIFRAS_CONTROL: LEE1_NUM_GIR ' +
                                Form('##',ident1), status_io);
         wr_num_gir.numero_gir := num1;
         ACTUALIZA1_NUM_GIR(wr_num_gir, status_io);
         if status_io <> ok_io then
            DESPLIEGA_ERROR('ACTUALIZA_NUM_GIR', status_io);
         LEE1_NUM_GIR(wr_num_gir, ident2, status_io);
         if status_io <> ok_io then
            DESPLIEGA_ERROR('CIFRAS_CONTROL: LEE1_NUM_GIR ' +
                                Form('##',ident2),status_io);
         wr_num_gir.numero_gir := num2;
         ACTUALIZA1_NUM_GIR(wr_num_gir, status_io);
         if status_io <> ok_io then
            DESPLIEGA_ERROR('ACTUALIZA_NUM_GIR', status_io);
         LEE1_NUM_GIR(wr_num_gir, proceso_aux, status_io);
         if status_io <> ok_io then
            DESPLIEGA_ERROR('CIFRAS_CONTROL: LEE1_NUM_GIR ' +
                 Form('##',proceso_aux), status_io);
         wr_num_gir.numero_gir := num1;
         ACTUALIZA1_NUM_GIR(wr_num_gir, status_io);
         if status_io <> ok_io then
            DESPLIEGA_ERROR('ACTUALIZA_NUM_GIR', status_io);
         LEE1_NUM_GIR(wr_num_gir, proceso_aux2, status_io);
         if status_io <> ok_io then
            DESPLIEGA_ERROR('CIFRAS_CONTROL: LEE1_NUM_GIR ' +
                                Form('##',proceso_aux), status_io);
         wr_num_gir.numero_gir := num1;
         ACTUALIZA1_NUM_GIR(wr_num_gir, status_io);
         if status_io <> ok_io then
            DESPLIEGA_ERROR('ACTUALIZA_NUM_GIR', status_io);
         end; {end afecta_num_gir}

      BEGIN                        { BEGIN CIF_APER procedimiento principal }
         base := nil;
         cta_intentos:= 0;
         for wtu:=1 to 255 do
             blancos[wtu] := ' ';
         INICIALIZA;
         NormalCursor;
         ENCABEZADO_PANTALLA('CIFRAS CONTROL DE GIROS');
         SetNumeric(on);
         SetPromptAttr(BlackonLtGray);
         SetStringAttr(LtGrayOnBlack);
         HouseCursorAtEnd := off;
         status_io := ABRES(base,'NUM_GIR',abre_num_gir,modo_recuperacion,
                            cierra_num_gir);
         if status_io <> ok_io then
            DESPLIEGA_ERROR('CIFRAS_CONTROL: ABRE_NUM_GIR',status_io);

         LEE1_NUM_GIR(wr_num_gir, p_giro_fin_citi, status_io);
         if status_io <> ok_io then
            DESPLIEGA_ERROR('CIFRAS_CONTROL: LEE1_NUM_GIR ' +
                                Form('##',p_giro_fin_citi), status_io);
         primer_giro := wr_num_gir.numero_gir;
         LEE1_NUM_GIR(wr_num_gir, u_giro_fin_citi, status_io);
         if status_io <> ok_io then
            DESPLIEGA_ERROR('CIFRAS_CONTROL: LEE1_NUM_GIR ' +
                            Form('##',u_giro_fin_citi), status_io);
         ultimo_giro := wr_num_gir.numero_gir;
         if not DisplayWindow(citiwin) then
            DESPLIEGA_ERROR('ERROR AL DESPLEGAR VENTANA', 8);
         LEE_DATOS('', 5, 9);
         ok_citibank := not press_escape and CHECA_DATOS('CITI BANK');
         if ok_citibank then
            begin
            move(primer_giro_pregunta[1], primer_giro[1], sizeof(tpo_folio));
            move(ultimo_giro_pregunta[1], ultimo_giro[1], sizeof(tpo_folio));
            AFECTA_NUM_GIR(p_giro_inicio_citi, u_giro_inicio_citi,
                           primer_giro, ultimo_giro, sig_num_imprimir_citi,
                           p_giro_usado_citi);
            end;

         LEE1_NUM_GIR(wr_num_gir, p_giro_fin_chemical, status_io);
         if status_io <> ok_io then
```

```
      DESPLIEGA_ERROR('CIFRAS_CONTROL: LEE1_NUM_GIR '+
                      Form('##',p_giro_fin_chemical), status_io);
   primer_giro := wr_num_gir.numero_gir;
   LEE1_NUM_GIR(wr_num_gir, u_giro_fin_chemical, status_io);
   if status_io <> ok_io then
      DESPLIEGA_ERROR('CIFRAS_CONTROL: LEE1_NUM_GIR '+
                      Form('##',u_giro_fin_chemical), status_io);
   ultimo_giro := wr_num_gir.numero_gir;

   if not DisplayWindow(chemicalwin) then
      DESPLIEGA_ERROR('ERROR AL DESPLEGAR VENTANA', 8);
   LEE_DATOS('', 46, 9);
   ok_chemical:= not press_escape and CHECA_DATOS('CHEMICAL BANK');
   if ok_chemical and ok_citibank then
      begin
      move(primer_giro_pregunta[1], primer_giro[1], sizeof(tpo_folio));
      move(ultimo_giro_pregunta[1], ultimo_giro[1], sizeof(tpo_folio));
      AFECTA_NUM_GIR(p_giro_inicio_chemical, u_giro_inicio_chemical,
                     primer_giro, ultimo_giro, sig_num_imprimir_chemical,
                     p_giro_usado_chemical);
      end;

   LEE1_NUM_GIR(wr_num_gir, p_giro_fin_libre, status_io);
   if status_io <> ok_io then
      DESPLIEGA_ERROR('CIFRAS_CONTROL: LEE1_NUM_GIR '+
                      Form('##',p_giro_fin_libre), status_io);
   primer_giro := wr_num_gir.numero_gir;
   LEE1_NUM_GIR(wr_num_gir, u_giro_fin_libre, status_io);
   if status_io <> ok_io then
      DESPLIEGA_ERROR('CIFRAS_CONTROL: LEE1_NUM_GIR '+
                      Form('##',u_giro_fin_libre), status_io);
   ultimo_giro := wr_num_gir.numero_gir;

   if not DisplayWindow(librewin) then
      DESPLIEGA_ERROR('ERROR AL DESPLEGAR VENTANA', 8);
   LEE_DATOS('', 27, 18);
   ok_libre := not press_escape and CHECA_DATOS('LIBRES');
   if ok_libre and ok_citibank and ok_chemical then
      begin
      move(primer_giro_pregunta[1], primer_giro[1], sizeof(tpo_folio));
      move(ultimo_giro_pregunta[1], ultimo_giro[1], sizeof(tpo_folio));
      AFECTA_NUM_GIR(p_giro_inicio_libre, u_giro_inicio_libre,
                     primer_giro, ultimo_giro, sig_num_imprimir_libre,
                     p_giro_usado_libre);
      MENSAJE('','CIFRAS CORRECTAS', 21, error_usuario);
      ACTUALIZA_STATUS_PROCESO(cifras_apertura, realizado_ctr, status_io);
      if status_io <> ok_io then
         DESPLIEGA_ERROR('CIFRAS_CIERRE.ACTUALIZA_STATUS_PROCESO',
                         status_io);
      end
   else
      MENSAJE('','CIFRAS INCORRECTAS', 21, error_usuario);

   status_io := CIERRAS(base);
   if (status_io <> ok_io) then
       DESPLIEGA_ERROR('CIFRAS_CIERRE : CIERRAS',status_io);
   LIBERA_VENTANAS;
   HiddenCursor;
END.   { CIF_APER }□
```

Programa para captura automática:

```
{$I-,V-,R-,N+,E+}
PROGRAM CAPTURA_AUTOMATICA;
{ Programa que genera los registros de giros automaticamente }

USES
   DOS,              { funciones del DOS }
   TPCRT,            { manejo de teclado, pantalla, colores y sonido }
   TPSTRING,         { manejo de strings }
   TPEDIT,           { edicion de datos }
   TPENTRY,          { pantallas de captura }
   TPDATE,           { manejo de fechas }
   COLORDEF,         { definicion de colores para pantallas }
   PRINTER,          { manejo de la impresora}
   UBTRIEVE,         { manejo de archivos }
   UCONTROL,         { manejo del archivo de control del sistema }
   UDCLGIRO,         { manejo de declaraciones globales }
   UGRALGIR,         { rutinas generales }
   UIMPRIME,         { rutinas de impresion }
   UMSJGIRO,         { rutinas de mensajes a pantalla }
   URUTGIRO,         { rutinas generales }
   UBEN_GIR,         { manejo del archivo de beneficiarios }
   UCTAHAB,          { manejo del archivo de cuentahabientes }
   UDIVISA,          { manejo del archivo de divisas }
   UCORRESP,         { manejo del archivo de corresponsales }
   UGIROS;           { manejo de la cartera del sistema }

CONST
   archivo_salida = 'f:\giros\datos\cap_aut.lst';
   tipo_cambio : extended = 1.00;

VAR
   wr_benef          : r_benefgir;
   wr_giros          : r_giros;
   llave_benef       : llave1_ben;
   llave_giros       : llave1_giros;
   cuentahabiente    : tpo_corresponsal;
   corresponsal      : tpo_corresponsal;
   divisa            : tpo_divisa;
   cuenta_cargo      : tpo_cuenta;
   numero_folio      : tpo_folio;
   total_nomina      : extended;
   escape            : boolean;
   fecha_expedicion,
   fecha_captura     : tpo_fecha;
   fecha_str         : DateString;
   status_io         : integer;
   reg_inicio_gir,
   reg_fin_gir       : extended;
   reg_leidos,
   reg_procesados,
   reg_escritos_gir  : integer;


   PROCEDURE LEE_NOMINA;
   { obtiene la nomina que debe ser procesada y los datos para procesarla }
    VAR
      wr_corresp        : r_corresp;
      wr_ctahabte       : r_ctahabte;
      wr_divisa         : r_divisa;
      llave_divisa      : llave1_div;
      llave_corresp     : llave1_corr;
      llave_ctahabte    : llave1_ctahabte;
      string_aux        : string;
      pide_tipo_cambio,
      dummy             : boolean;
      opcion            : char;

    BEGIN
    string_aux := copy(blancos, 1, 10);
    ENCABEZADO_PANTALLA('CAPTURA_AUTOMATICA');
    Fastwrite(Pad('Esc-Terminar', 80), 25,1, BlackonLTGray);
```

```
HouseCursoratEnd := false;
CursortoEnd := False;
ForceUpper := true;
Repeat
   ReadString('NOMINA A PROCESAR    :  ', 10, 14, 10, WhiteonBlue,
            YellowonBlack, LtGrayonBlack, escape, string_aux);
   if not escape then
      begin
      Fastwrite(Pad(' ', 80), 25,1, LtGrayonBlack);
      move(blancos[1], llave_ctahabte[1], sizeof(llave1_ctahabte));
      move(string_aux[1], llave_ctahabte[1], length(Trim(string_aux)));
      LEE_CTAHABTE(wr_ctahabte, llave_ctahabte, status_io);
      if status_io = no_existe_llave then
         _DESPLIEGA_ERROR('EL CUENTAHABIENTE NO EXISTE', dummy);
      if status_io = ok_io then
         begin
         cuentahabiente := wr_ctahabte.clave_ctahabte;
         pide_tipo_cambio := (wr_ctahabte.divisa_monto_ctahabte =
                              moneda_nacional);
         end;
      end;
until (escape) or (status_io = ok_io);

if not escape then
   begin
   string_aux := copy(blancos, 1, 10);
   Repeat
      Fastwrite(Pad('Esc-Terminar', 80), 25,1, BlackonLTGray);
      ReadString('CORRESPONSAL         :  ',
                 13, 14,
                 10, WhiteonBlue, YellowonBlack, LtGrayonBlack,
                 escape, string_aux);
      if not escape then
         begin
         Fastwrite(Pad(' ', 80), 25,1, LtGrayonBlack);
         move(blancos[1], llave_corresp[1], sizeof(llave1_corr));
         move(string_aux[1], llave_corresp[1],
                 length(Trim(string_aux)));
         LEE_CORRESP(wr_corresp, llave_corresp, status_io);
         if status_io = no_existe_llave then
            _DESPLIEGA_ERROR('EL CORRESPONSAL NO EXISTE' +
                             llave_corresp, dummy);
         if status_io = ok_io then
            corresponsal := wr_corresp.clave_corr;
         end;
    until (escape) or (status_io = ok_io);
    end;  {end if escape}

   if not escape then
      begin
      string_aux := copy(blancos, 1, 10);
      Repeat
         Fastwrite(Pad('Esc-Terminar', 80), 25,1, BlackonLTGray);
         ReadString('DIVISA               :  ',
                    16, 14,
                    3,WhiteonBlue, YellowonBlack, LtGrayonBlack,
                    escape, string_aux);
         if not escape then
            begin
            Fastwrite(Pad(' ', 80), 25,1, LtGrayonBlack);
            move(blancos[1], llave_divisa[1], sizeof(llave1_div));
            move(string_aux[1], llave_divisa[1],
                 length(Trim(string_aux)));
            LEE_DIVISA(wr_divisa, llave_divisa, status_io);
            if status_io = no_existe_llave then
               _DESPLIEGA_ERROR('LA DIVISA NO EXISTE', dummy);
            if status_io = ok_io then
               divisa := wr_divisa.clave_div;
            end;
      until (escape) or (status_io = ok_io);
      end; {end if escape}

   if not escape then
      begin
      SetPromptAttr(WhiteonBlue);
```

```
        SetStringAttr(YellowonBlack);
        SetFieldAttr(YellowonBlack);
        string_aux := copy(blancos, 1, 10);
        Repeat
           fecha_str := '  /  /    ';
           Fastwrite(Pad('Esc-Terminar', 80), 25,1, BlackonLTGray);
           EditString('FECHA DE EXPEDICION  : ',
                      18, 14,
                      10, '99/99/9999',
                      0,
                      escape,
                      fecha_str);
           if not escape then
              begin
              Fastwrite(Pad(' ', 80), 25,1, LtGrayonBlack);
              string_aux := fecha_tpofechast(fecha_str);
              move(string_aux[1], fecha_expedicion[1],
                   sizeof(tpo_fecha));
              if fecha_expedicion < fecha_captura then
                 _DESPLIEGA_ERROR('FECHA INVALIDA ', dummy);
              end;
        until (escape) or (fecha_expedicion >= fecha_captura);
        end; {end if escape}

     if not(escape) and (pide_tipo_cambio) then
        begin
        SetPromptAttr(WhiteonBlue);
        SetStringAttr(YellowonBlack);
        string_aux := copy(blancos, 1, 10);
        tipo_cambio := 1.00;
        Repeat
           ReadReal('TIPO DE CAMBIO       :',
                    20, 14, 12,
                    WhiteonBlue, YellowonBlack,
                    6,
                    1, 99999.99,
                    escape,
                    tipo_cambio);
           ReadCharacter (Pad('¿TIPO DE CAMBIO CORRECTO [S/N]? ', 80),
                25,1, BlackOnLtGray, ['S', 's', 'n', 'N'], opcion);
        until (escape) or (opcion in ['s', 'S']);
        FastWrite(Pad(' ', 80),25,1, LtGrayOnBlack);
        end; {end if escape}
  END; {end lee_nomina}

  PROCEDURE CUENTA(var cuenta_aux : tpo_cuenta);
  { obtiene la cuenta del cuentahabiente }
    CONST
       cuenta_0  = '00000000000000000000';
    VAR
      wr_cuentahabte : r_ctahabte;
      llave          : llave1_ctahabte;

    BEGIN
      llave := cuentahabiente;
      LEE_CTAHABTE(wr_cuentahabte, llave, status_io);
      if status_io <> ok_io then
         DESPLIEGA_ERROR('CAP_AUT.CUENTA LEE_CUENTAHABTE '+
                         cuentahabiente, status_io);
      with wr_cuentahabte do
         begin
         if (trim(cuenta_ctahabte) <> '0') and
            (cuenta_ctahabte <> cuenta_0) and
            (trim(cuenta_ctahabte) <> '') then
            cuenta_aux := cuenta_ctahabte
         else
            cuenta_aux := cta_tesofe_ctahabte;
         end;
    end;  {end CUENTA}


  PROCEDURE INICIALIZA;
  { inicializa variables y abre archivos }
     var
        wtu,
```

```
      status_io : integer;
      string_aux : string;

   begin
   base := nil;
   for wtu := 1 to 255 do
       blancos[wtu] := ' ';
   status_io := ABRES(base, 'BEN_GIR', abre_benef, modo_lectura,
                      cierra_benef);
   if status_io <> ok_io then
      DESPLIEGA_ERROR('CAP_AUT.ABRE_BENEF', status_io);
   status_io := ABRES(base, 'GIROS', abre_giros, modo_recuperacion,
                      cierra_giros);
   if status_io <> ok_io then
      DESPLIEGA_ERROR('CAP_AUT.ABRE_GIROS', status_io);
   status_io := ABRES(base, 'CTAHABTE', abre_ctahabte, modo_lectura,
                      cierra_ctahabte);
   if status_io <> ok_io then
      DESPLIEGA_ERROR('CAP_AUT.ABRE_CTAHABTE', status_io);
   status_io := ABRES(base, 'CORRESP', abre_corresp, modo_lectura,
                      cierra_corresp);
   if status_io <> ok_io then
      DESPLIEGA_ERROR('CAP_AUT.ABRE_CORRESP', status_io);
   status_io := ABRES(base, 'DIVISAS', abre_divisa, modo_lectura,
                      cierra_divisa);
   if status_io <> ok_io then
      DESPLIEGA_ERROR('CAP_AUT.ABRE_DIVISA', status_io);


   FECHA_HOY(fecha_captura, status_io);
   if status_io <> ok_io then
      DESPLIEGA_ERROR('CAP_AUT.FECHA_HOY', status_io);

   LEE_NOMINA;
   if not escape then
      begin
      total_nomina := 0;
      LEE1_ULT_GIROS(wr_giros, llave_giros, status_io);
      if (status_io <> ok_io) and (status_io <> fin_archivo) then
         DESPLIEGA_ERROR('CAPT_AUT.LEE1_ULTIMO_GIROS', status_io);
      if status_io = fin_archivo then
         numero_folio := '00000001'
      else
         numero_folio := wr_giros.folio_gir;
      CUENTA(cuenta_cargo);
      REGISTROS_ARCHIVO(reg_inicio_gir, bloque_pos_gir, status_io);
      if status_io <> ok_io then
         DESPLIEGA_ERROR('CAP_AUT.REG_ARCH_1', status_io);
      reg_leidos      := 0;
      reg_procesados  := 0;
      reg_escritos_gir := 0;
      end; {end if not escape}
   INICIA_REPORTE(archivo_salida);
   CONTROL_DE_IMPRESORAS (margen_superior,margen_izquierdo, carta,
                          o_horizontal, cpp_16, lpp_8, 120);
   renglones_x_hoja := 60;
   titulo1 := '';
   titulo2 := 'CIFRAS DE CONTROL DE CAPTURA AUTOMATICA';
   titulo3 := '';
   titulo4 := '';
   end; {end inicializa}

PROCEDURE FIN_CAPTURA;
{ imprime cifras de control del proceso y cierra archivos }
   CONST
       fmto_monto = '###,###,###,###,###.##';

   VAR
     status_io : integer;

   begin
   if not escape then
      begin
      REGISTROS_ARCHIVO(reg_fin_gir, bloque_pos_gir, status_io);
      if status_io <> ok_io then
```

```
            DESPLIEGA_ERROR('CAP_AUT.REG_ARCH_2', status_io);
         ENCABEZADO;
         writeln(lst, 'Registros en Giros antes del Proceso   : ',
                      form(fmto_monto, reg_inicio_gir));
         writeln(lst, 'Registros escritos                     : ',
                      form(fmto_monto, reg_escritos_gir));
         writeln(lst, 'Registros en Giros despues del Proceso : ',
                      form(fmto_monto,reg_fin_gir));
         writeln(lst);
         writeln(lst, 'Registros Leidos de Beneficiarios      : ',
              reg_leidos);
         writeln(lst, 'Registros Procesados                   : ',
              reg_procesados);
         writeln(lst);
         writeln(lst, 'Monto Total por Nomina de ', cuentahabiente , ' : ',
                      Form(fmto_monto, total_nomina));
         end;
      status_io := CIERRAS(base);
      if status_io <> ok_io then
         DESPLIEGA_ERROR('CAP_AUT.CIERRAS', status_io);
      TERMINA_REPORTE;
      end; {end fin_reporte}


   PROCEDURE NUMERO_NUEVO_FOLIO(var numero_fol : tpo_folio;
                                var nuevo_fol : tpo_folio);
   { obtiene el numero de folio a asignar al nuevo registro }
      VAR
        numero_str : string;
        numero     : LongInt;
      begin
      if not  Str2Long(numero_fol, numero) then
         DESPLIEGA_ERROR('CAP_AUT.NUM_NVO_FOLIO ' + numero_fol , 999);
      inc(numero);
      numero_str := Long2Str(numero);
      numero_str := LeftpadCh(numero_str, '0', sizeof(tpo_folio));
      move(numero_str[1], nuevo_fol[1], sizeof(tpo_folio));
      numero_fol := nuevo_fol;
      end; {end NUMERO_NUEVO_FOLIO}


   PROCEDURE CREA_REGISTRO(wr_beneficiario    : r_benefgir;
                           var wr_giros       : r_giros);
   { arma el registro para incluirlo a la cartera de giros }
      begin
         with wr_giros do
            begin
            NUMERO_NUEVO_FOLIO(numero_folio, folio_gir);
            fecha_gir          := fecha_expedicion;
            fecha_cap_gir      := fecha_captura;
            divisa_gir         := divisa;
            corresponsal_gir   := corresponsal;
            if (tipo_cambio = 1.00) then
               monto_gir          := wr_beneficiario.monto_ben
            else
               monto_gir := (Round((wr_beneficiario.monto_ben /
                            tipo_Cambio) * 100)) / 100;
            move(blancos[1], beneficiario_gir[1],
                    sizeof(tpo_beneficiario150));
            move(wr_beneficiario.clave_ben[1], beneficiario_gir[1],
                 sizeof(tpo_corresponsal));;
            move(blancos[1], numero_cheque_gir[1],sizeof(tpo_folio));
            cuentahabte_gir    := cuentahabiente;
            cuenta_cargo_gir   := cuenta_cargo;
            estado_gir         := negativo;
            total_nomina       := total_nomina + monto_gir;
            end; {end while}
      end; {end CREA_REGISTRO}


   PROCEDURE LEE_PRIMER_BENEFICIARIO(var wr_registro : r_benefgir;
                                     var llave       : llavel_ben;
                                     var status      : integer);
   { obtiene el primer registro de beneficiarios a procesar }
   begin
```

```
      LEE_PRIMER_BENEF(wr_registro, llave, status);
      inc(reg_leidos);
      while ((Trim(wr_registro.extra_ben) <> Trim(cuentahabiente)) or
            (wr_registro.monto_ben = 0.0)) and
            (status_io = ok_io) do
            begin
            LEE_SIGUIENTE_BENEF(wr_registro, llave, status);
            inc(reg_leidos);
            end;
  end; {end LEE_PRIMER_BENEFICIARIO}

  PROCEDURE LEE_SIGUIENTE_BENEFICIARIO(var wr_registro : r_benefgir;
                                       var llave       : llave1_ben;
                                       var status      : integer);
  { obtiene el siguiente registro de beneficiarios a procesar }
  begin
      repeat
         LEE_SIGUIENTE_BENEF(wr_registro, llave, status);
         inc(reg_leidos);
      until ((Trim(wr_registro.extra_ben) = Trim(cuentahabiente)) and
            (wr_registro.monto_ben <> 0.0)) or
            (status_io <> ok_io);
  end; {end LEE_SIGUIENTE_BENEFICIARIO}

  PROCEDURE GENERA_REGISTROS;
  { genera los registros para la nomina procesada }
  begin
    LEE_PRIMER_BENEFICIARIO(wr_benef, llave_benef, status_io);
    while (status_io = ok_io) do
        begin
        CREA_REGISTRO(wr_benef, wr_giros);
        ESCRIBE_GIROS(wr_giros, status_io);
        if status_io <> ok_io then
           DESPLIEGA_ERROR('CAP_AUT.ESCRIBE_GIROS', status_io);
        inc(reg_escritos_gir);
        inc(reg_procesados);
        LEE_SIGUIENTE_BENEFICIARIO(wr_benef, llave_benef, status_io);
        end;
    if (status_io <> ok_io) and (status_io <> fin_archivo) then
       DESPLIEGA_ERROR('LEE_BENEFICIARIO', status_io);
  end;  {end GENERA REGISTROS}

BEGIN                 { BEGIN CAP_AUT, proceso principal }
  INICIALIZA;
  ENCABEZADO_PANTALLA('CAPTURA AUTOMATICA');
  MENSAJE('ESPERAR', 'REALIZANDO : CAPTURA AUTOMATICA', 21, aviso_ptr);
  if not escape then
     GENERA_REGISTROS;
  FIN_CAPTURA;
END. { CAP_AUT }
```

Programa para generar los giros:

```
{$E+,N+,V-,I-}
PROGRAM GIROS_;
{ programa para la generacion de los giros }

  USES
   UCTRLGIR,          { manejo del archivo de control de giros }
   UNUMBLET,          { rutina para cambio de numero a letras en ingles }
   UNUMLETR,          { rutina para cambio de numero a letras en ingles }
   TPCRT,             { manejo de teclado, pantalla, colores y sonido }
   TPSTRING,          { manejo de strings }
   TPEDIT,            { edicion de datos }
   COLORDEF,          { definicion de colores para pantallas }
   PRINTER,           { manejo de la impresora}
   UCONTROL,          { manejo del archivo de control del sistema }
   UDCLGIRO,          { manejo de declaraciones globales }
   UGRALGIR,          { rutinas generales }
   UMSJGIRO,          { rutinas de mensajes a pantalla }
   URUTGIRO,          { rutinas generales }
   UBEN_GIR,          { manejo del archivo de beneficiarios }
   UDIVISA,           { manejo del archivo de divisas }
   UCORRESP,          { manejo del archivo de corresponsales }
   UGIROS;            { manejo de la cartera del sistema }
   UNUMGIR,           { manejo del archivo de cifras de control del sistema }
   UPROGIR;           { manejo de constantes del archivo de control }

   CONST
     {Control de impresora}
     reset_impresora = #27#64;
     largo_hoja = #27#67#18;
     NLQuality = #27#120#1;
     cancela_NLQ = #27#120#0;
     tipo_roman = #27#107#0;
     tipo_bold = #27#69;
     cancela_bold = #27#70;
     cancela_roman = #27#107#1;
     italica = #27#52;
     cancela_italica = #27#53;
     doble_alto = #27#119#1;
     cancela_doble_alto = #27#119#0;
     condensado = #27#15;
     cancela_condensado = #18;
     FormFeed = #12;
     avanza_linea = #10;
     carry_return = #13;
     mueve_n_216 = #27#74#1;
     mueve_6_216 = #27#74#6;
     mueve_7_216 = #27#74#7;
     mueve_8_216 = #27#74#8;
     mueve_11_216 = #27#74#11;
     mueve_12_216 = #27#74#12;
     mueve_13_216 = #27#74#13;
     mueve_15_216 = #27#74#15;
     mueve_17_216 = #27#74#17;
     mueve_23_216 = #27#74#23;
     bidireccional = #27#85#0;
     ram_off = #27#37#0#0;
     ram_on = #27#37#1#0;
     unidireccional = #27#85#1;
     avanza_0  = #27#33#0;

     ENTER = ^M;
     ESC   = ^[;
     reg_procesados  : integer = 0;
     giros_generados : integer = 0;
     giros_por_generarse : integer = 0;
     titulo1 : string = '';
     titulo2 : string = '';
     alimentacion_manual : boolean = FALSE;
     citi = 'CITIES    ';
```

```
    chemical = 'CHEMICALES';
    libre = 'LIBRES    ';
    bilbao1 = 'BANCO BILBAO VIZCAYA';
    bilbao2 = 'DEPTO. CENTRAL DE EXTRANJERO';
    bilbao3 = 'APTDO. CORREOS 472.    28080';
    bilbao4 = 'MADRID.    CTA. 008-002595-4';

    {FACSIMIL}
    renglon_1  = '
'                  + '6022282222I2222222x97';
    renglon_11 = '
'                  + '6022282222T2222222vw7';
    renglon_2  = '
'                  + 'MABCDN1111$%&+1,-[\y>';
    renglon_22 = '
'                  + 'MABCDN1111UVWX1YZ34y>';
    renglon_3  = '
'                  + 'MbcdeN11111]?@EFG111>';
    renglon_33 = '
'                  + 'MbcdeN111115^_afg111>';
    renglon_4  = '
'                  + 'MrstuN11111HIJy11111>';
    renglon_44 = '
'                  + 'MrstuN11111hijy11111>';
    renglon_5  = '
'                  + 'M#()*N11RLOP11111111>';
    renglon_55 = '
'                  + 'M#()*N11RImn11111111>';
    renglon_6  = '
'                  + '.:===<=KSk=========;/';
    renglon_66 = '
'                  + '.:===<=pqk=========;/';

  TYPE
    r_impresion = record
                  fecha_imp : string;
                  folio_imp : tpo_folio;
                  monto_imp : string;
                  monto_texto1 : string[68];
                  monto_texto2 : string[68];
                  monto_texto3 : string[68];
                  beneficiario_imp1 : string[60];
                  beneficiario_imp2 : string[60];
                  beneficiario_imp3 : string[60];
                  corresponsal_imp : string[45];
                  direccion_corresponsal_imp1 : string[30];
                  direccion_corresponsal_imp2 : string[30];
                  divisa_imp       : string[3];
                  divisa_nom_imp   : string[100];
                  numero_cheque_imp: string[8];
                  end;

  VAR
    wr_corresp          : r_corresp;
    llave_corresp       : llave1_corr;
    wr_divisa           : r_divisa;
    llave_divisa        : llave1_div;
    wr_benef            : r_benefgir;
    llave_benef         : llave1_ben;
    wr_giros            : r_giros;
    llave_giros         : llave1_giros;
    wr_impresion        : r_impresion;
    corresponsal        : tpo_corresponsal;
    status_io           : integer;
    giros_erroneos      : array [1..50] of tpo_folio;
    error_giro,
    continuar           : boolean;
    fecha_expedicion,
    fecha_sistema       : tpo_fecha;
    opcion              : char;


  PROCEDURE IMPRIME_LINEA(cadena : string);
  { manda la cadena a impresion }
     VAR
```

```
      status_io : integer;
      dummy     : boolean;

   begin
   repeat
      write(lst, cadena);
      status_io := IOResult;
      if status_io <> ok_io then
        _DESPLIEGA_ERROR('PROBLEMAS CON LA IMPRESORA, FAVOR DE
                          REVISARLA', dummy);
   until status_io = ok_io;
   end; {end imprime_linea}


PROCEDURE ERROR_DESC_IMPRESORA(error_st: string; error_num : byte);
{ para configuracion de la impresora }
   begin
   IMPRIME_LINEA(reset_impresora);
   EJECUTA('A:\GIROS2\LLAVE2\DESFACS.EXE', '', 'CONFIGURAR IMPRESORA',
           proceso_dummy, no, no);
   DESPLIEGA_ERROR(error_st, error_num);
   end;

PROCEDURE INICIALIZA_IMPRESORA;
{ para configuracion de la impresora }
   VAR
     status_io : integer;
     dummy     : boolean;

   begin
   repeat
     write(lst,  reset_impresora);
     status_io := IOResult;
     if status_io <> ok_io then
        _DESPLIEGA_ERROR('PROBLEMAS CON LA IMPRESORA FAVOR DE
                          REVISARLA', dummy);
   until status_io = ok_io;
   repeat
     write(lst, largo_hoja + NLQuality + tipo_roman);
     status_io := IOResult;
     if status_io <> ok_io then
       _DESPLIEGA_ERROR('PROBLEMAS CON LA IMPRESORA FAVOR DE REVISARLA',
                        dummy);
   until status_io = 0;
   end;{end inicializa_impresora}

PROCEDURE PROCESA_ERROR_GIRO;
{ para interrumpir la generacion de giros }
   VAR
     car : char;
   BEGIN
   inc (giros_por_generarse);
   giros_erroneos[giros_por_generarse] := wr_giros.folio_gir;
   {-preguntar al usuario si desea interrumpir la generación}
   ReadCharacter (Pad(' ENTER-Continuar  ESC-Interrumpir: ', 80),
                  25,1, BlackOnLtGray, [ENTER, ESC], car   );
   continuar := (car = ENTER);
   END; { PROCESA_ERROR_GIRO }

FUNCTION NUMERO_NUEVO_GIRO(corresponsal : tpo_corresponsal) : string;
{ obtiene el numero de folio del siguiente formato a imprimir }
   VAR
      wr_num_gir : r_num_gir;
      proceso1,
      proceso2,
      numero_giro,
      numero_giro2 : longint;
      numero_aux : tpo_folio;
      str_aux    : string;
      dummy : boolean;
   begin
   proceso1 := 0;
   proceso2 := 0;
   if corresponsal = citi then
      begin
```

```
        proceso1 := sig_num_imprimir_citi;
        proceso2 := u_giro_inicio_citi;
        end
      else
        if corresponsal = chemical then
          begin
          proceso1 := sig_num_imprimir_chemical;
          proceso2 := u_giro_inicio_chemical;
          end
        else
          begin
          proceso1 := sig_num_imprimir_libre;
          proceso2 := u_giro_inicio_libre;
          end;
      LEE1_NUM_GIR(wr_num_gir, proceso2, status_io);
      if status_io <> ok_io then
        ERROR_DESC_IMPRESORA('NUMERO_NUEVO_GIRO.LEEQ_NUM_GIR2',
                                    status_io);
      if not Str2Long(wr_num_gir.numero_gir, numero_giro2 ) then
        begin
        _DESPLIEGA_ERROR('ERROR EN NUMERO DE GIRO 2', dummy);
        status_io := CIERRAS(base);
        IMPRIME_LINEA(reset_impresora);
        EJECUTA('A:\GIROS2\LLAVE2\DESFACS.EXE', '', 'CONFIGURAR
                    IMPRESORA',
                proceso_dummy, no, no);
        halt(0);
        end;
      LEE1_NUM_GIR(wr_num_gir, proceso1, status_io);
      if status_io <> ok_io then
        ERROR_DESC_IMPRESORA('NUMERO_NUEVO_GIRO.LEE1_NUM_GIR1',
                             status_io);
      numero_aux := wr_num_gir.numero_gir;
      str_aux := copy(blancos, 1, sizeof(tpo_folio));
      move(numero_aux[1], str_aux[1], sizeof(tpo_folio));
      NUMERO_NUEVO_GIRO := str_aux;
      if not Str2Long(numero_aux, numero_giro) then
        begin
        _DESPLIEGA_ERROR('ERROR EN NUMERO DE GIRO 1', dummy);
        status_io := CIERRAS(base);
        IMPRIME_LINEA(reset_impresora);
        EJECUTA('A:\GIROS2\LLAVE2\DESFACS.EXE', '', 'CONFIGURAR
                    IMPRESORA',
                proceso_dummy, no, no);
        halt(0);
        end;
      if numero_giro <= numero_giro2 then
        begin
        inc(numero_giro);
        str_aux := leftpadch(Long2Str(numero_giro), '0',
                             sizeof(tpo_folio));
        move(str_aux[1], wr_num_gir.numero_gir[1], sizeof(tpo_folio));
        ACTUALIZA1_NUM_GIR(wr_num_gir, status_io);
        if status_io <> ok_io then
          ERROR_DESC_IMPRESORA('ACTUALIZA1_NUM_GIR', status_io);
        end
      else
        begin
        _DESPLIEGA_ERROR('EL SIG. NUMERO DE GIRO ES MAYOR AL ULTIMO GIRO
                          DEL DIA', dummy);
        status_io := CIERRAS(base);
        IMPRIME_LINEA(reset_impresora);
        EJECUTA('A:\GIROS2\LLAVE2\DESFACS.EXE', '', 'CONFIGURAR
             IMPRESORA',
                proceso_dummy, no, no);
        halt(0);
        end;
      end; {end numero_nuevo_giro}

    FUNCTION DIVISA_CON_CENTAVOS(divisa_pregunta: tpo_divisa) : boolean;
    { detecta que divisas se procesan con centavos y cuales no }
      begin
        if ((divisa_pregunta = yenes) or
            (divisa_pregunta = pesetas) or
            (divisa_pregunta = liras_italianas) or
```

```
        (divisa_pregunta = francos_belgas_financieros)) then
        DIVISA_CON_CENTAVOS := false
      else
        DIVISA_CON_CENTAVOS := true;
    end; {end divisa con centavos}

PROCEDURE IMPRIME_GIRO(wr_imprime : r_impresion);
{ manda a imprimir el giro correspondiente }
  CONST
    pos_benef  = 97;
    pos_titulo = 39;
    pos_fecha  = 62;
    pos_numgir = 5;
    pos_divisa = 50;
    pos_mtexto = 31;

  VAR
    dummy       : char;
    pos_punto   : integer;
    num_aux     : byte;
    string_aux  : string;
    divisa_aux  : tpo_divisa;

  begin
  INICIALIZA_IMPRESORA;
  with wr_imprime do
     begin
     string_aux := '';
     pos_punto := Pos('.', monto_imp);
     IMPRIME_LINEA(mueve_n_216 + carry_return);
     IMPRIME_LINEA(leftpad(numero_cheque_imp, pos_numgir +
                           length(numero_cheque_imp)));
     IMPRIME_LINEA(carry_return +
                   doble_alto +
                   tipo_bold +
                   leftpad(fecha_imp, pos_fecha + length(fecha_imp)));
     IMPRIME_LINEA(cancela_doble_alto +
                   cancela_bold +
                   avanza_linea +
                   avanza_linea +
                   avanza_linea);
     move(divisa_imp[1], divisa_aux[1], sizeof(tpo_divisa));
     if divisa_con_centavos(divisa_aux) then
        string_aux :=copy(monto_imp, 1,pos_punto - 1)  + divisa_imp +
              copy(monto_imp, pos_punto + 1,length(monto_imp) -
                 pos_punto) + 'CENTS'
     else
        begin
        monto_imp := copy(monto_imp, 1,pos_punto - 1);
        string_aux :=copy(monto_imp, 1,pos_punto - 1)  + divisa_imp;
        end;
     if (divisa_imp = dolar_americano) and
        (Trim(corresponsal_imp) = '') then
        divisa_imp := copy(blancos, 1, sizeof(tpo_divisa));
     IMPRIME_LINEA(leftpad('', pos_numgir + length(numero_cheque_imp))
                        + leftpad(divisa_imp, pos_divisa +
                        length(divisa_imp)) + '*' + monto_imp + '*' +
                        avanza_linea);
     IMPRIME_LINEA(italica + tipo_bold);
     IMPRIME_LINEA(leftpad(titulo1, pos_titulo + length(titulo1)));
     IMPRIME_LINEA(mueve_17_216 + doble_alto);
     pos_punto := pos('.',monto_imp);
     IMPRIME_LINEA(string_aux);
     IMPRIME_LINEA(cancela_doble_alto + carry_return + mueve_6_216);
     IMPRIME_LINEA(leftpad(titulo2, pos_titulo + length(titulo2)) +
                   avanza_linea);
     IMPRIME_LINEA(cancela_italica + cancela_bold + cancela_doble_alto
                   + mueve_7_216 + condensado + carry_return);
     IMPRIME_LINEA(leftpad(beneficiario_imp1, pos_benef) +
                   avanza_linea);
     IMPRIME_LINEA(leftpad(beneficiario_imp2, pos_benef) +
                   avanza_linea);
     IMPRIME_LINEA(leftpad(beneficiario_imp3, pos_benef) + mueve_12_216
                    + cancela_condensado + cancela_NLQ + cancela_roman +
                    carry_return);
```

```
        IMPRIME_LINEA(ram_on + unidireccional + avanza_0 + carry_return);
        IMPRIME_LINEA(renglon_1 + carry_return);
        IMPRIME_LINEA(renglon_1 + carry_return);
        IMPRIME_LINEA(renglon_11 + carry_return);
        IMPRIME_LINEA(renglon_11 + carry_return);
        IMPRIME_LINEA(ram_off + bidireccional + mueve_15_216 +
                carry_return);
        IMPRIME_LINEA(NLQuality + tipo_roman + condensado);
        IMPRIME_LINEA(' ' + correspondal_imp + cancela_condensado +
                cancela_NLQ + cancela_roman + mueve_8_216 +
                carry_return);
        IMPRIME_LINEA(ram_on + unidireccional + avanza_0 + carry_return);
        IMPRIME_LINEA(renglon_2 + carry_return);
        IMPRIME_LINEA(renglon_2 + carry_return);
        IMPRIME_LINEA(renglon_22 + carry_return);
        IMPRIME_LINEA(renglon_22 + carry_return);
        IMPRIME_LINEA(mueve_23_216 + carry_return + avanza_0);
        IMPRIME_LINEA(renglon_3 + carry_return);
        IMPRIME_LINEA(renglon_3 + carry_return);
        IMPRIME_LINEA(renglon_33 + carry_return);
        IMPRIME_LINEA(renglon_33 + carry_return);
        IMPRIME_LINEA(ram_off + bidireccional + carry_return);
        IMPRIME_LINEA(NLQuality + tipo_roman + condensado);
        IMPRIME_LINEA(' ' + direccion_corresponsal_impl + carry_return);
        IMPRIME_LINEA(leftpad(monto_texto1, pos_mtexto +
                length(monto_texto1)) + cancela_NLQ + cancela_roman);
        IMPRIME_LINEA(mueve_23_216 + carry_return);
        IMPRIME_LINEA(ram_on + unidireccional + avanza_0 + carry_return);
        IMPRIME_LINEA(renglon_4 + carry_return);
        IMPRIME_LINEA(renglon_4 + carry_return);
        IMPRIME_LINEA(renglon_44 + carry_return);
        IMPRIME_LINEA(renglon_44 + carry_return);
        IMPRIME_LINEA(mueve_11_216 + carry_return + avanza_0);
        IMPRIME_LINEA(ram_off + bidireccional + carry_return);
        IMPRIME_LINEA(NLQuality + tipo_roman + condensado + ' ' +
                direccion_corresponsal_imp2 + carry_return +
                leftpad(monto_texto2, pos_mtexto +
                length(monto_texto2)) + cancela_NLQ + cancela_roman +
                cancela_condensado + mueve_13_216 + carry_return);
        IMPRIME_LINEA(ram_on + unidireccional + avanza_0 + carry_return);
        IMPRIME_LINEA(renglon_5 + carry_return);
        IMPRIME_LINEA(renglon_5 + carry_return);
        IMPRIME_LINEA(renglon_55 + carry_return);
        IMPRIME_LINEA(renglon_55 + carry_return);
        IMPRIME_LINEA(mueve_23_216 + carry_return + avanza_0);
        IMPRIME_LINEA(renglon_6 + carry_return);
        IMPRIME_LINEA(renglon_6 + carry_return);
        IMPRIME_LINEA(renglon_66 + carry_return);
        IMPRIME_LINEA(renglon_66 + carry_return);
        IMPRIME_LINEA(ram_off + bidireccional + carry_return);
        if divisa_imp = 'ESB' THEN
           IMPRIME_LINEA(NLQuality + tipo_roman + condensado + ' ' +
                        bilbao4 + carry_return + leftpad(monto_texto3,
                        pos_mtexto + length(monto_texto3)))
        else
           IMPRIME_LINEA(NLQuality + tipo_roman + condensado +
                        leftpad(monto_texto3, pos_mtexto +
                        length(monto_texto3)));
        IMPRIME_LINEA(FormFeed + reset_impresora);
        end; {end with}
    end;{end IMPRIME_GIRO}

PROCEDURE BLANQUEA_REGISTRO(wr_registro : r_giros);
{ limpia el registro }
    begin
    move(blancos[1], wr_registro, sizeof(r_giros));
    end;

PROCEDURE OBTEN_NOMBRE_DIVISA (var divisa_st : string;
                               llave : llavel_div);
{ obtiene del catalogo de divisas el nombre correspondiente }
    CONST
      rutina_actual = 'OBTEN_NOMBRE_DIVISA';

    VAR
```

```
    wr_divisa : r_divisa;

  begin
  LEE_DIVISA(wr_divisa, llave, status_io);
  if status_io <> ok_io then
     begin
     MENSAJE('ERROR',rutina_actual + LLAVE + form('##',status_io), 21,
             error_usuario);
     error_giro := true;
     exit;
     end;
  divisa_st := copy(blancos, 1, sizeof(wr_divisa.nombre_ing_div));
  move(wr_divisa.nombre_ing_div[1], divisa_st[1],
       sizeof(tpo_beneficiario));
  end; {end obten_nombre_divisa}


PROCEDURE OBTEN_CORRESPONSAL(var wr_corr : r_corresp;
                             llave : llavel_corr);
{ obtiene del catalogo de corresponsales el nombre correspondiente }
  CONST
     rutina_actual = 'OBTEN_CORRESPONSAL';

  begin
  LEE_CORRESP(wr_corresp, llave, status_io);
  if status_io <> ok_io then
     begin
     MENSAJE('ERROR',rutina_actual + LLAVE + form('##',status_io), 21,
             error_usuario);
     error_giro := true;
     exit;
     end;
  end; {end obten_corresponsal}

PROCEDURE OBTEN_BENEFICIARIO(var benef_str : string;
                             llave : llavel_ben);
{ obtiene del catalogo de beneficiarios el nombre correspondiente }
  CONST
     rutina_actual = 'OBTEN_BENEFICIARIO';

  begin
  LEE_BENEF(wr_benef, llave, status_io);
  if status_io <> ok_io then
     begin
     MENSAJE('ERROR',rutina_actual + LLAVE + form('##',status_io), 21,
             error_usuario);
     error_giro := true;
     exit;
     end;
  move(wr_benef.nombre_ben[1], benef_str[1],
       sizeof(tpo_beneficiario150));
  end; {end obten_beneficiario}


PROCEDURE PREPARA_IMPRESION(var wr_reg_giro  : r_giros;
                            var wr_impresion : r_impresion);
{ obtiene los datos necesarios para la impresion de los giros }
  CONST
    formato_monto = '###,###,###,###,###,###.##';
    num_centavos : integer = 2;
  VAR
    monto_texto,
    str_aux : string;

  begin
  move(blancos[1], wr_impresion, sizeof(r_impresion));
  with wr_reg_giro, wr_impresion do
     begin
     fecha_imp := FECHA_TEXTO(fecha_gir);
     folio_imp := folio_gir;
     llave_divisa := divisa_gir;
     str_aux := '';
     OBTEN_NOMBRE_DIVISA(str_aux, llave_divisa);
     divisa_imp := divisa_gir;
     divisa_nom_imp := Trim(str_aux);
```

```
corresponsal_imp := copy(blancos, 1,
                         sizeof(wr_corresp.nombre_corr));
str_aux:= copy(blancos, 1, sizeof(wr_corresp.direccion_corr));
if (corresponsal_gir <> citi) and
   (corresponsal_gir <> chemical) and
   (Trim(corresponsal_gir) <> 'BILBAOMAD') then
   begin
   llave_corresp := corresponsal_gir;
   OBTEN_CORRESPONSAL(wr_corresp, llave_corresp);
   move(wr_corresp.nombre_corr[1], corresponsal_imp[1],
        sizeof(wr_corresp.nombre_corr));
   move(wr_corresp.direccion_corr[1], str_aux[1],
       sizeof(wr_corresp.direccion_corr));
   end;
if (Trim(corresponsal_gir) = 'BILBAOMAD')  then
   begin
   corresponsal_imp := copy(bilbao1, 1, length(bilbao1));
   direccion_corresponsal_imp1 := copy(bilbao2, 1,
                                       length(bilbao2));
   direccion_corresponsal_imp2 := copy(bilbao3, 1,
                                       length(bilbao3));
   end
else
   begin
   WordWrap(Trim(str_aux), direccion_corresponsal_imp1, str_aux,
            sizeof(direccion_corresponsal_imp1) -1, true);
   WordWrap(Trim(str_aux), direccion_corresponsal_imp2, str_aux,
            sizeof(direccion_corresponsal_imp2) -1, true);
   corresponsal_imp := Trim(corresponsal_imp);
   end;
str_aux := copy(blancos, 1, sizeof(tpo_beneficiario150));
move(beneficiario_gir[1], str_aux[1],
     sizeof(tpo_beneficiario150));
if length(Trim(str_aux)) <= sizeof(tpo_corresponsal) then
   begin
   move(blancos[1], llave_benef[1], sizeof(llave1_ben));
   move(str_aux[1], llave_benef[1], sizeof(llave1_ben));
   OBTEN_BENEFICIARIO(str_aux, llave_benef);
   end;
str_aux := Trim(str_aux);
beneficiario_imp1 := copy(blancos, 1, sizeof(beneficiario_imp1) -
                          1);
beneficiario_imp2 := beneficiario_imp1;
beneficiario_imp3 := copy(blancos, 1, sizeof(beneficiario_imp3) -
                          1);
WordWrap(Trim(str_aux), beneficiario_imp1, str_aux,
         sizeof(beneficiario_imp1) - 1, true);
WordWrap(Trim(str_aux), beneficiario_imp2, str_aux,
         sizeof(beneficiario_imp1) - 1, true);
if length(str_aux) > 0 then
   begin
   str_aux := Trim(str_aux);
   move(str_aux[1], beneficiario_imp3[1],
        length(str_aux));
   end;
numero_cheque_imp:=
            NUMERO_NUEVO_GIRO(wr_reg_giro.corresponsal_gir);
move(numero_cheque_imp[1], numero_cheque_gir[1],
     sizeof(tpo_folio));
monto_imp := Trim(Form(formato_monto, monto_gir));
str_aux := '';
if divisa_gir <> pesetas then
   NUMBER_TO_LETERS(monto_gir,num_centavos,divisa_nom_imp,'',
                    false, monto_texto, status_io)
else
   CANTIDAD_A_LETRAS(monto_gir,num_centavos,divisa_nom_imp,'',
                     false, monto_texto, status_io);
if divisa_con_centavos(divisa_gir) then
   str_aux := ' ' + copy(monto_imp, pos('.', monto_imp) + 1,
              num_centavos) + '/100';
insert(str_aux, monto_texto, pos(')', monto_texto));
str_aux := Trim(monto_texto);
monto_texto1 := copy(blancos, 1, sizeof(monto_texto1) - 1);
monto_texto2 := monto_texto1;
monto_texto3 := monto_texto1;
```

```
      WordWrap(str_aux, monto_texto1, str_aux, sizeof(monto_texto1) - 2,
               false);
      if length(str_aux) > 0 then
         begin
         monto_texto1 := Padch(Trim(monto_texto1), '-',
                               sizeof(monto_texto1) - 1);
         WordWrap(Trim(str_aux), monto_texto2, str_aux,
                  sizeof(monto_texto2) - 3, false);
         end;
      if length(str_aux) > 0 then
         begin
         monto_texto2 := Padch('-' + Trim(monto_texto2), '-' ,
                               sizeof(monto_texto2) - 1);
         WordWrap(Trim(str_aux), monto_texto3, str_aux,
                  sizeof(monto_texto3) - 2, false);
         monto_texto3 := '-' + Trim(monto_texto3);
         end
      else
         if Trim(monto_texto2) <> '' then
            begin
            monto_texto2 := '--' +Trim(monto_texto2);
            end;
      if status_io <> ok_io then
         error_giro := true;
      end; {end with }
   end; {end prepara_impresion}


PROCEDURE AFECTA_CONTROL_GIROS( wr_impresion : r_impresion;
                                wr_giros: r_giros );
{ actualiza el archivo de control }
   CONST
      fmt_monto = '################';
      rutina_actual = 'AFECTA_CONTROL_GIROS';
   VAR
      wr_ctrl_gir: r_ctrl_gir;
      nr_ctrl_gir: r_ctrl_gir;  {-nuevo registro}
      monto             : string[15];
      dummy_f           : tpo_fecha;
      str_aux           : string;
      string_aux        : string[8];
      status_io         : integer;
   BEGIN
   move(blancos[1], nr_ctrl_gir, sizeof(r_giros));
   with nr_ctrl_gir do
      begin
      move (wr_giros.folio_gir[1], folio_ctrgir[1], sizeof(tpo_folio));
      move (wr_impresion.numero_cheque_imp[1], num_giro_ctrgir[1],
          sizeof(tpo_folio));
      estado_giro_ctrgir:= giro_generado;
      FECHA_HOY(fecha_gen_ctrgir, status_io);
      if status_io <> ok_io then
        ERROR_DESC_IMPRESORA('AFECTA_CTRL_GIR.FECHA_HOY', status_io);
      DA_FECHA_Y_HORA_MSDOS (dummy_f, hora_gen_ctrgir);
      move(wr_impresion.divisa_imp[1], divisa_ctrgir[1],
          sizeof(tpo_divisa));
      monto := form(fmt_monto, wr_giros.monto_gir);
      move(monto[1], monto_ctrgir[1], sizeof(tpo_monto_gir));
      corresponsal_ctrgir := wr_giros.corresponsal_gir;
      beneficiario_ctrgir := wr_giros.beneficiario_gir;
      cuentahabte_ctrgir := wr_giros.cuentahabte_gir;
      end;{end with}
   LEE2_CTRL_GIR(wr_ctrl_gir, nr_ctrl_gir.folio_ctrgir, status_io);
   case status_io of
      ok_io:
         {-se trata de un giro regenerado; actualizar}
         begin
         ACTUALIZA2_CTRL_GIR(nr_ctrl_gir, status_io);
         if status_io <> ok_io then
            ERROR_DESC_IMPRESORA('GIROS.ACTUALIZA1_CTRL_GIR',
                                 status_io);
         end;
      no_existe_llave:
         {-se trata de un mensaje nuevo}
```

```
        begin
        ESCRIBE2_CTRL_GIR(nr_ctrl_gir, status_io);
        if status_io <> ok_io then
          ERROR_DESC_IMPRESORA('GIROS.ESCRIBE2_CTRL_GIR', status_io);
        end
      else
        ERROR_DESC_IMPRESORA(rutina_actual + ': LEE1_CTRL_GIR',
                             status_io);
    end; {case}
  END; { AFECTA_CONTROL_GIROS}


PROCEDURE PROCESA_REGISTRO(var wr_registro : r_giros);
{ procesa los registro que generan un giro }
  begin
  continuar := true;
  error_giro := false;
  with wr_registro do
    begin
    fastwrite('No. Folio     : ' + folio_gir, 7, 4, YellowOnBlue);
    if (estado_gir = negativo) and
       (fecha_gir <= fecha_expedicion)  then
       begin
       if alimentacion_manual then
          _DESPLIEGA_ERROR('FAVOR DE ALIMENTAR EL GIRO', continuar);
       if continuar then
          PREPARA_IMPRESION(wr_registro, wr_impresion);
       if error_giro then
          PROCESA_ERROR_GIRO;
       if (not error_giro) and (continuar) then
          begin
          IMPRIME_GIRO (wr_impresion);
          delay(50);
          inc(giros_generados);
          estado_gir := generado;
          AFECTA_CONTROL_GIROS(wr_impresion, wr_registro);
          move(wr_impresion.numero_cheque_imp[1],
               wr_registro.numero_cheque_gir[1], sizeof(tpo_folio));
          ACTUALIZAL_GIROS(wr_registro, status_io);
          if status_io <> ok_io then
             ERROR_DESC_IMPRESORA('PROCESA_REGISTRO.ACTUALIZAL_GIROS',
                                  status_io);
          end;
       end;
    end;{end with}
  end; {end procesa_registro}


PROCEDURE PROCESA_REGISTROS_CORRESPONSAL(corresponsal : string);
{ procesa los registro del corresponsal correspondiente }
  CONST
    leyenda1 = 'ALIMENTAR FORMATOS DE ';
    leyenda2 = 'ALIMENTAR FORMATOS ';

  VAR
    string_aux : string;
  begin
  continuar := true;
  if corresponsal = libre then
     string_aux := leyenda2
  else
     string_aux := leyenda1;
  string_aux := string_aux + corresponsal + ' (Enter-Continuar  ESC-
                Suspender)';
  ReadCharacter (Pad(string_aux,80),25,1, BlackOnLtGray,[ENTER, ESC],
                 opcion);
  if opcion = enter then
     begin
     fastwrite(Pad('ESC - Interrumpir', 80), 25, 1, BlackonLtGray);
     LEE_PRIMER_GIROS(wr_giros, llave_giros, status_io);
     if status_io <> ok_io then
        ERROR_DESC_IMPRESORA('GIROS.LEE1_PRIMER_GIROS', status_io);
     while (status_io = ok_io) and (continuar) do
        begin
        inc(reg_procesados);
```

```
            if (wr_giros.corresponsal_gir = corresponsal) or
               ((corresponsal = libre ) and
                (wr_giros.corresponsal_gir <> citi) and
                (wr_giros.corresponsal_gir <> chemical)) then
               PROCESA_REGISTRO(wr_giros);
            BLANQUEA_REGISTRO(wr_giros);
            LEE_SIG_GIROS(wr_giros, llave_giros, status_io);
            if keypressed then
               begin
               opcion := readkey;
               continuar := (opcion = enter);
               end;
            end; {end while}
         if (status_io <> fin_archivo) and (status_io <> ok_io) then
            ERROR_DESC_IMPRESORA('GIROS.LEE1_SIG_GIROS', status_io);
         end;
      end; {end procedure procesa_registros_corresponsal}

   PROCEDURE DESPLIEGA_DATOS_FIN;
   { presenta las cifras de control del proceso }
      begin
      IMPRIME_LINEA(reset_impresora);
      LIMPIA_VENTANA;
      FastWrite ('GIROS ', 7,4, YellowOnBlue);
      FastWrite ('   Giros generados      : ' +
                  Form('####', giros_generados), 8, 3, YellowOnBlue);
      FastWrite ('   Giros por generarse  : ' + Form('####',
                  giros_por_generarse), 9, 3,   YellowOnBlue);
      FastWrite ('   Registros procesados : ' + Form('####', reg_procesados
                  / 3), 10, 3, YellowOnBlue );
      end;{end despliega_datos_fin}


BEGIN              { BEGIN  GIROS  procedimiento principal }
HiddenCursor;
ENCABEZADO_PANTALLA('GENERACION DE GIROS');
ABRE_ARCHIVOS(Giros, altas);
status_io := ABRES(base, 'NUM_GIRO', abre_num_gir, modo_recuperacion,
                   cierra_num_gir);
if status_io <> ok_io then
   ERROR_DESC_IMPRESORA('GIROS.ABRES NUM_GIRO',status_io);
status_io := ABRES(base, 'CTRL_GIR', abre_ctrl_gir, modo_recuperacion,
                   cierra_ctrl_gir);
if status_io <> ok_io then
   ERROR_DESC_IMPRESORA('GIROS.ABRES NUM_GIRO',status_io);
FECHA_HOY(fecha_sistema, status_io);
if status_io <> ok_io then
   ERROR_DESC_IMPRESORA('GIROS.FECHA_HOY', status_io);
FECHA_SIGUIENTE(fecha_expedicion, status_io);
if status_io <> ok_io then
   ERROR_DESC_IMPRESORA('GIROS.FECHA_HOY', status_io);
ReadCharacter (Pad(' INSERTAR EL DISCO LLAVE 2. ENTER-Continuar ESC-Salir',
            80),
               25,1, BlackOnLtGray, [ENTER, ESC], opcion);
if opcion = ENTER then
   begin
   IMPRIME_LINEA(Reset_impresora);
   EJECUTA('A:\GIROS2\LLAVE2\CARFACS.EXE', '', 'CONFIGURAR IMPRESORA',
            proceso_dummy, no, no);

   ReadCharacter (Pad(' ALIMENTACION DE FORMAS CONTINUA O MANUAL (C/M)?',
                  80),
                  25,1, BlackOnLtGray, ['C', 'c', 'M', 'm'], opcion);
   alimentacion_manual := (Upcase(opcion) = 'M');
   IMPRIME_LINEA(reset_impresora);
   ENCABEZADO_PANTALLA('GENERACION DE GIROS');
   BLANQUEA_REGISTRO(wr_giros);
   PROCESA_REGISTROS_CORRESPONSAL(citi);
   PROCESA_REGISTROS_CORRESPONSAL(chemical);
   PROCESA_REGISTROS_CORRESPONSAL(libre);
   DESPLIEGA_DATOS_FIN;
   LINEA_GUIA('ENTER-Continuar');
   while (ENTER <> readkey) do ;
   IMPRIME_LINEA(reset_impresora);
   EJECUTA('A:\GIROS2\LLAVE2\DESFACS.EXE', '', 'CONFIGURAR IMPRESORA',
```

```
            proceso_dummy, no, no);
   end;
ENCABEZADO_PANTALLA('GENERACION DE GIROS');
status_io := CIERRAS(base);
if (status_io <> ok_io) then
   DESPLIEGA_ERROR('GIROS.CIERRAS', status_io);
NormalCursor;
ACTUALIZA_STATUS_PROCESO(generacion_giros, realizado_ctr, status_io);
if status_io <> ok_io then
   DESPLIEGA_ERROR('GIROS.ACTUALIZA_STATUS(generacion_giros)', status_io);
EJECUTA('f:\giros\sistema\menugiro.exe',form('####', generacion_giros),
        'MENU PRINCIPAL', generacion_giros, si, no);
END.  {END GIROS}
```

Listado del programa par acifrasa de cierre:

```
{$I-,V-,R-,N+,E+}
PROGRAM CIF_FIN;
{ proceso para validar las cifras de cierre del sistema }
USES
   TPCRT,              { manejo de teclado, pantalla, colores y sonido }
   TPSTRING,           { manejo de strings }
   TPEDIT,             { edicion de datos }
   TPENTRY,            { pantallas de captura }
   COLORDEF,           { definicion de colores para pantallas }
   PRINTER,            { manejo de la impresora}
   UBTRIEVE,           { manejo de archivos }
   UCONTROL,           { manejo del archivo de control del sistema }
   UDCLGIRO,           { manejo de declaraciones globales }
   UGRALGIR,           { rutinas generales }
   UMSJGIRO,           { rutinas de mensajes a pantalla }
   TPWINDOW,           { manejo de ventanas }
   ULECTURA,           { captura de datos en formato estandar }
   UNUMGIR,            { manejo del archivo de cifras de control del sistema }
   UPROGIR;            { manejo de constantes del archivo de control }

CONST
  numero_cifras_contar = 5;
  max_intentos         = 3;

VAR
  wr_num_gir                    : r_num_gir;
  primer_giro_inicio,
  ultimo_giro_inicio,
  primer_giro_usado,
  ultimo_giro_usado,
  primer_giro_fin,
  ultimo_giro_fin,
  sig_numero_imprimir           : tpo_folio;
  num_giros_usados_pregunta,
  num_giros_fin_pregunta,
  primer_giro_usado_pregunta,
  ultimo_giro_usado_pregunta,
  primer_giro_fin_pregunta,
  ultimo_giro_fin_pregunta,
  errores_pregunta              : string;
  p_giro_inicio,
  u_giro_inicio,
  p_giro_usado,
  u_giro_usado,
  p_giro_fin,
  u_giro_fin,
  errores_giro,
  num_giros_usados,
  num_giros_fin,
  errores                       : longint;
  status_io,
  cta_intentos,
  recorre,
  wtu                           : integer;
  llave_num_gir                 : llavel_num_gir;
  proceso                       : longint;
  numero_str                    : string;
  ok_citibank,
  ok_chemical,
  ok_libre,
  press_escape                  : boolean;
  citiwin,
  chemicalwin,
  librewin                      : windowptr;

  PROCEDURE INICIALIZA;
  { inicializa variables, ventanas y abre archivos }
    begin
    explode := true;
```

```
  shadow := true;
  SoundFlagW := false;
  ShadowAttr := DkGrayonCyan;
  if not makewindow(citiwin, 4, 7, 40, 14, true, true, true,
                    BlackonLtGray, BlackonLtGray, LtGrayonBlack, 'CITI
                      BANK')
  then  DESPLIEGA_ERROR('ERROR EN INICIALIZA VENTANA', 8);
  if not makewindow(chemicalwin, 42, 7, 76, 14, true, true, true,
                    BlackonLtGray, BlackonLtGray, LtGrayonBlack,
                      'CHEMICAL BANK')
  then  DESPLIEGA_ERROR('ERROR EN INICIALIZA VENTANA', 8);
  if not makewindow(librewin, 24, 16, 58, 23, true, true, true,
                    BlackonLtGray, BlackonLtGray, LtGrayonBlack, 'FORMATO
                      LIBRE')
  then  DESPLIEGA_ERROR('ERROR EN INICIALIZA VENTANA', 8);
  NormalCursor;
  ENCABEZADO_PANTALLA('CIFRAS CONTROL DE GIROS');
  SetNumeric(on);
  SetPromptAttr(BlackonLtGray);
  SetStringAttr(LtGrayOnBlack);
  HouseCursorAtEnd := off;
  status_io := ABRES(base,'NUM_GIR',abre_num_gir,modo_recuperacion,
                     cierra_num_gir);
  if status_io <> ok_io then
     DESPLIEGA_ERROR('CIFRAS_CONTROL: ABRE_NUM_GIR',status_io);
  end;  {end inicializa}

PROCEDURE LIBERA_VENTANAS;
{ borra y libera espacio de las ventanas }
  begin
  DisposeWindow(citiwin);
  DisposeWindow(Chemicalwin);
  DisposeWindow(librewin);
  end; {end libera_ventanas}


PROCEDURE LEE_DATOS(cadena : string; x, y: byte);
{ captura los cifras }
   begin
   press_escape := true;
   num_giros_usados_pregunta:= copy(blancos, 1, numero_cifras_contar);
   EditString('NO. GIROS USADOS' + cadena + ' : ',
              y, x, numero_cifras_contar, '99999',
              0, press_escape, num_giros_usados_pregunta);
   primer_giro_usado_pregunta:= copy(blancos, 1, sizeof(tpo_folio));
   EditString('NO. PRIMER GIRO USADO'+ cadena + ' : ',
              y + 2, x, sizeof(tpo_folio), '99999999',
              0, press_escape, primer_giro_usado_pregunta);
   primer_giro_usado_pregunta := Leftpadch(primer_giro_usado_pregunta,
                                           '0', sizeof(tpo_folio));
   ultimo_giro_usado_pregunta:= copy(blancos, 1, sizeof(tpo_folio));
   EditString('NO. ULTIMO GIRO USADO' + cadena + ' : ',
              y + 4, x, sizeof(tpo_folio),'99999999',
              0 , press_escape, ultimo_giro_usado_pregunta);
   ultimo_giro_usado_pregunta := Leftpadch(ultimo_giro_usado_pregunta,
                                           '0', sizeof(tpo_folio));
   clrscr;
   num_giros_fin_pregunta:= copy(blancos, 1, numero_cifras_contar);
   EditString('NO. GIROS FIN' + cadena + ' : ',
              y, x, numero_cifras_contar, '99999',
              0, press_escape, num_giros_fin_pregunta);
   primer_giro_fin_pregunta:= copy(blancos, 1, sizeof(tpo_folio));
   EditString('NO. PRIMER GIRO FIN'+ cadena + ' : ',
              y + 2, x, sizeof(tpo_folio),'99999999',
              0, press_escape, primer_giro_fin_pregunta);
   primer_giro_fin_pregunta:= Leftpadch(primer_giro_fin_pregunta, '0',
                                        sizeof(tpo_folio));
   ultimo_giro_fin_pregunta:= copy(blancos, 1, sizeof(tpo_folio));
   EditString('NO. ULTIMO GIRO FIN' + cadena + ' : ',
              y + 4, x, sizeof(tpo_folio),
              '99999999', 0 , press_escape, ultimo_giro_fin_pregunta);
   ultimo_giro_fin_pregunta := Leftpadch(ultimo_giro_fin_pregunta, '0',
                                         sizeof(tpo_folio));
   clrscr;
   errores_pregunta := copy(blancos, 1, numero_cifras_contar);
```

```
      EditString('NO. ERRORES' + cadena + ' : ',
                 y + 2, x, numero_cifras_contar,
                 '99999', 0 , press_escape, errores_pregunta);
    end; {end LEE_DATOS}

  FUNCTION CHECA_DATOS(banco : string): boolean;
  { checa los datos capturados con los calculados }
    VAR
      dummy,
      dummy1,
      dummy2,
      dummy3,
      dummy4,
      dummy5,
      dummy6,
      dummy7  : boolean;
      sig_num : longint;

    begin
    dummy := false;
    val(num_giros_usados_pregunta, num_giros_usados, status_io);
    val(num_giros_fin_pregunta, num_giros_fin, status_io);
    val(errores_pregunta, errores_giro, status_io);
    if str2long(primer_giro_inicio, p_giro_inicio) and
       str2long(primer_giro_usado, p_giro_usado) and
       str2long(primer_giro_fin_pregunta, p_giro_fin) and
       str2long(ultimo_giro_usado_pregunta, u_giro_usado) and
       str2long(ultimo_giro_fin_pregunta, u_giro_fin) and
       str2long(ultimo_giro_inicio, u_giro_inicio) and
       str2long(sig_numero_imprimir, sig_num)
     then
       begin
       dummy1 := ((num_giros_usados = u_giro_usado - p_giro_usado + 1) or
                  ((num_giros_usados = 0) and (p_giro_usado = sig_num)));
       dummy2 := ((primer_giro_usado = primer_giro_usado_pregunta) or
                  ((sig_num = p_giro_usado ) and
                     (primer_giro_usado_pregunta = '00000000')));
       dummy3 := ((u_giro_usado = sig_num - 1) or
                  ((u_giro_usado = 0)  and (sig_num = p_giro_usado)));
       dummy4 := ((p_giro_fin = u_giro_usado + 1) or
                  ((p_giro_fin = p_giro_inicio) and (sig_num =
                     p_giro_usado)) or
                  ((p_giro_fin = 0) and (sig_num > u_giro_inicio)));
       dummy5 := ((p_giro_fin = p_giro_usado + num_giros_usados) or
                  ((p_giro_fin = p_giro_inicio ) and (num_giros_usados =
                     0)) or
                  ((p_giro_fin = 0) and (sig_num > u_giro_inicio)));
       dummy6 := ((u_giro_fin = p_giro_fin + num_giros_fin - 1) and
                  (u_giro_fin = p_giro_inicio + num_giros_usados +
                     num_giros_fin  - 1) or
                  ((u_giro_fin = u_giro_inicio) and (num_giros_usados =
                     0)) or
                  ((p_giro_fin = 0) and (sig_num > u_giro_inicio)));
       dummy7 := (errores_giro = errores);
       dummy := dummy1 and dummy2 and dummy3 and
                dummy4 and dummy5 and dummy6 and dummy7;
       end
     else
       DESPLIEGA_ERROR('CIFRAS CONTROL, NUMEROS GIROS '+ banco,
  status_io);
     CHECA_DATOS := dummy;
    end; {end checa_datos}

  PROCEDURE AFECTA_NUM_GIR(ident1, ident2, ident3 : longint;
                           num1, num2, num3 : tipo_folio);
  { actualiza las cifras }
    begin
    LEE1_NUM_GIR(wi_num_gir, ident1, status_io);
    if status_io <> ok_io then
       DESPLIEGA_ERROR('CIFRAS CONTROL: LEE1_NUM_GIR ' +
                       Form('##',ident1), status_io);
    wi_num_gir.numero_gir := num1;
    ACTUALIZA1_NUM_GIR(wi_num_gir, status_io);
    if status_io <> ok_io then
       DESPLIEGA_ERROR('ACTUALIZA_NUM_GIR', status_io);
```

```
      LEE1_NUM_GIR(wr_num_gir, ident2, status_io);
      if status_io <> ok_io then
         DESPLIEGA_ERROR('CIFRAS_CONTROL: LEE1_NUM_GIR ' +
                         Form('##',ident2), status_io);
      wr_num_gir.numero_gir := num2;
      ACTUALIZA1_NUM_GIR(wr_num_gir, status_io);
      if status_io <> ok_io then
         DESPLIEGA_ERROR('ACTUALIZA_NUM_GIR', status_io);
      LEE1_NUM_GIR(wr_num_gir, ident3, status_io);
      if status_io <> ok_io then
         DESPLIEGA_ERROR('CIFRAS_CONTROL: LEE1_NUM_GIR ' +
                         Form('##',ident3), status_io);
      wr_num_gir.numero_gir := num3;
      ACTUALIZA1_NUM_GIR(wr_num_gir, status_io);
      if status_io <> ok_io then
         DESPLIEGA_ERROR('ACTUALIZA_NUM_GIR', status_io);
      end; {end afecta_num_gir}

BEGIN                          { BEGIN CIF_FIN procedimiento principal }
   base := nil;
   cta_intentos:= 0;
   for wtu:=1 to 255 do
      blancos[wtu] := ' ';
   INICIALIZA;
   LEE1_NUM_GIR(wr_num_gir, p_giro_inicio_citi, status_io);
   if status_io <> ok_io then
      DESPLIEGA_ERROR('CIFRAS_CONTROL: LEE1_NUM_GIR ' +
                      Form('##',p_giro_fin_citi), status_io);
   primer_giro_inicio := wr_num_gir.numero_gir;

   LEE1_NUM_GIR(wr_num_gir, p_giro_usado_citi, status_io);
   if status_io <> ok_io then
      DESPLIEGA_ERROR('CIFRAS_CONTROL: LEE1_NUM_GIR ' +
                      Form('##',p_giro_fin_citi), status_io);
   primer_giro_usado := wr_num_gir.numero_gir;

   LEE1_NUM_GIR(wr_num_gir, errores_citi, status_io);
   if status_io <> ok_io then
      DESPLIEGA_ERROR('CIFRAS_CONTROL: LEE1_NUM_GIR ' +
                      Form('##',u_giro_fin_citi), status_io);
   errores := wr_num_gir.num_aux_num_gir;

   LEE1_NUM_GIR(wr_num_gir, sig_num_imprimir_citi, status_io);
   if status_io <> ok_io then
      DESPLIEGA_ERROR('CIFRAS_CONTROL: LEE1_NUM_GIR ' +
                      Form('##',sig_num_imprimir_citi), status_io);
   sig_numero_imprimir := wr_num_gir.numero_gir;

   LEE1_NUM_GIR(wr_num_gir, u_giro_inicio_citi, status_io);
   if status_io <> ok_io then
      DESPLIEGA_ERROR('CIFRAS_CONTROL: LEE1_NUM_GIR ' +
                      Form('##',u_giro_inicio_citi), status_io);
   ultimo_giro_inicio := wr_num_gir.numero_gir;

   if not DisplayWindow(citiwin) then
      DESPLIEGA_ERROR('ERROR AL DESPLEGAR VENTANA', 8);
   LEE_DATOS('', 5, 9);
   ok_citibank := not pres_escape and CHECA_DATOS('CITI BANK');
   if ok_citibank then
      begin
      move(ultimo_giro_usado_pregunta[1], ultimo_giro_usado[1],
           sizeof(tpo_folio));
      move(primer_giro_fin_pregunta[1], primer_giro_fin[1],
           sizeof(tpo_folio));
      move(ultimo_giro_fin_pregunta[1], ultimo_giro_fin[1],
           sizeof(tpo_folio));
      AFECTA_NUM_GIR(u_giro_usado_citi, p_giro_fin_citi, u_giro_fin_citi,
                     ultimo_giro_usado, primer_giro_fin, ultimo_giro_fin);
      end;

   LEE1_NUM_GIR(wr_num_gir, p_giro_inicio_chemical, status_io);
   if status_io <> ok_io then
      DESPLIEGA_ERROR('CIFRAS_CONTROL: LEE1_NUM_GIR ' +
                      Form('##',p_giro_fin_chemical), status_io);
   primer_giro_inicio := wr_num_gir.numero_gir;
```

```
LEE1_NUM_GIR(wr_num_gir, p_giro_usado_chemical, status_io);
if status_io <> ok_io then
   DESPLIEGA_ERROR('CIFRAS_CONTROL: LEE1_NUM_GIR ' +
                   Form('##',p_giro_fin_chemical), status_io);
primer_giro_usado := wr_num_gir.numero_gir;

LEE1_NUM_GIR(wr_num_gir, errores_chemical, status_io);
if status_io <> ok_io then
   DESPLIEGA_ERROR('CIFRAS_CONTROL: LEE1_NUM_GIR ' +
                   Form('##',u_giro_fin_chemical), status_io);
errores := wr_num_gir.num_aux_num_gir;

LEE1_NUM_GIR(wr_num_gir, sig_num_imprimir_chemical, status_io);
if status_io <> ok_io then
   DESPLIEGA_ERROR('CIFRAS_CONTROL: LEE1_NUM_GIR ' +
                   Form('##',sig_num_imprimir_chemical), status_io);
sig_numero_imprimir := wr_num_gir.numero_gir;

LEE1_NUM_GIR(wr_num_gir, u_giro_inicio_chemical, status_io);
if status_io <> ok_io then
   DESPLIEGA_ERROR('CIFRAS_CONTROL: LEE1_NUM_GIR ' +
                   Form('##',u_giro_inicio_chemical), status_io);
ultimo_giro_inicio := wr_num_gir.numero_gir;

if not DisplayWindow(chemicalwin) then
   DESPLIEGA_ERROR('ERROR AL DESPLEGAR VENTANA', 8);
LEE_DATOS('', 43, 9);
ok_chemical:= not press_escape and CHECA_DATOS('CHEMICAL BANK');
if ok_citibank and ok_chemical then
   begin
   move(ultimo_giro_usado_pregunta[1], ultimo_giro_usado[1],
        sizeof(tpo_folio));
   move(primer_giro_fin_pregunta[1], primer_giro_fin[1],
        sizeof(tpo_folio));
   move(ultimo_giro_fin_pregunta[1], ultimo_giro_fin[1],
        sizeof(tpo_folio));
   AFECTA_NUM_GIR(u_giro_usado_chemical, p_giro_fin_chemical,
                  u_giro_fin_chemical, ultimo_giro_usado,
                  primer_giro_fin, ultimo_giro_fin);

   end;

LEE1_NUM_GIR(wr_num_gir, p_giro_inicio_libre, status_io);
if status_io <> ok_io then
   DESPLIEGA_ERROR('CIFRAS_CONTROL: LEE1_NUM_GIR ' +
                   Form('##',p_giro_fin_libre), status_io);
primer_giro_inicio := wr_num_gir.numero_gir;

LEE1_NUM_GIR(wr_num_gir, p_giro_usado_libre, status_io);
if status_io <> ok_io then
   DESPLIEGA_ERROR('CIFRAS_CONTROL: LEE1_NUM_GIR ' +
                   Form('##',p_giro_fin_libre), status_io);
primer_giro_usado := wr_num_gir.numero_gir;

LEE1_NUM_GIR(wr_num_gir, errores_libre, status_io);
if status_io <> ok_io then
   DESPLIEGA_ERROR('CIFRAS_CONTROL: LEE1_NUM_GIR ' +
                   Form('##',u_giro_fin_libre), status_io);
errores := wr_num_gir.num_aux_num_gir;

LEE1_NUM_GIR(wr_num_gir, sig_num_imprimir_libre, status_io);
if status_io <> ok_io then
   DESPLIEGA_ERROR('CIFRAS_CONTROL: LEE1_NUM_GIR ' +
                   Form('##',sig_num_imprimir_libre), status_io);
sig_numero_imprimir := wr_num_gir.numero_gir;

LEE1_NUM_GIR(wr_num_gir, u_giro_inicio_libre, status_io);
if status_io <> ok_io then
   DESPLIEGA_ERROR('CIFRAS_CONTROL: LEE1_NUM_GIR ' +
                   Form('##',u_giro_inicio_libre), status_io);
ultimo_giro_inicio := wr_num_gir.numero_gir;
if not DisplayWindow(librewin) then
   DESPLIEGA_ERROR('ERROR AL DESPLEGAR VENTANA', 8);
LEE_DATOS('', 25,10);
```

```
   ok_libre:= not press_escape and CHECA_DATOS('FORMATOS LIBRE');
   if ok_citibank and ok_chemical and ok_libre then
      begin
      move(ultimo_giro_usado_pregunta[1], ultimo_giro_usado[1],
           sizeof(tpo_folio));
      move(primer_giro_fin_pregunta[1], primer_giro_fin[1],
           sizeof(tpo_folio));
      move(ultimo_giro_fin_pregunta[1], ultimo_giro_fin[1],
           sizeof(tpo_folio));
      AFECTA_NUM_GIR(u_giro_usado_libre, p_giro_fin_libre,
                     u_giro_fin_libre, ultimo_giro_usado, primer_giro_fin,
                     ultimo_giro_fin);
      MENSAJE('','CIFRAS CORRECTAS', 21, error_usuario);
      ACTUALIZA_STATUS_PROCESO(cifras_cierre, realizado_ctr, status_io);
      if status_io <> ok_io then
         DESPLIEGA_ERROR('CIFRAS_FIN.ACTUALIZA_STATUS_PROCESO', status_io);
      end
   else
      MENSAJE('','CIFRAS INCORRECTAS', 21, error_usuario);
   status_io := CIERRAS(base);
   if (status_io <> ok_io) then
       DESPLIEGA_ERROR('CIFRAS_CIERRE : CIERRAS',status_io);
   LIBERA_VENTANAS;
(*   ENCABEZADO_PANTALLA('MENU PRINCIPAL');*)
   HiddenCursor;
END.   { CIF_FIN }
```

Programa que genera reportes de giros por generar o giros generados (segun parámetro).

```
{$I-,V-,R-,N+,E+}
PROGRAM REP_GEN;
{  Produce el reporte de Giros por generarse }

USES
   DOS,                { funciones del DOS }
   TPCRT,              { manejo de teclado, pantalla, colores y sonido }
   TPSTRING,           { manejo de strings }
   PRINTER,            { manejo de la impresora}
   UCONTROL,           { manejo del archivo de control del sistema }
   UDCLGIRO,           { manejo de declaraciones globales }
   UGRALGIR,           { rutinas generales }
   UIMPRIME,           { rutinas de impresion }
   UMSJGIRO,           { rutinas de mensajes a pantalla }
   URUTGIRO,           { rutinas generales }
   UGIROS;             { manejo de la cartera del sistema }

CONST
  doble_linea = '=';
  linea_sencilla = '-';
  archivo_salida = 'f:\giros\datos\rep_gen.lst';
  fmto_monto    = '###,###,###,###.##';

VAR
   total_folios_giros : integer;
   total_monto_giros  : extended;
   fecha_expedicion,
   fecha_sistema      : tpo_fecha;
   fecha_rep          : tpo_fecha_reporte;
   wr_giros           : r_giros;
   llave_giros        : llave2_giros;
   llave_ant          : llave2_giros;
   status_io          : integer;
   estado_reporte     : char;

   PROCEDURE INICIALIZA;
   { inicializa valores y abre archivos }
      VAR
         wtu,
         status_io : integer;
         string_aux : string;

      begin
      base := nil;
      for wtu := 1 to 255 do
          blancos[wtu] := ' ';
      status_io := ABRES(base, 'GIROS', abre_giros, modo_lectura,
                         cierra_giros);
      if status_io <> ok_io then
         DESPLIEGA_ERROR('REP_GEN.ABRE_GIROS', status_io);
      INICIA_REPORTE(archivo_salida);
      CONTROL_DE_IMPRESORAS (margen_superior,margen_izquierdo, carta,
                             o_horizontal, cpp_16, lpp_8, 120);
      renglones_x_hoja := 60;
      FECHA_HOY(fecha_sistema, status_io);
      if status_io <> ok_io then
         DESPLIEGA_ERROR('REP_GEN.FECHA_HOY', status_io);
      FECHA_SIGUIENTE(fecha_expedicion, status_io);
      if status_io <> ok_io then
         DESPLIEGA_ERROR('REP_GEN.FECHA_HOY', status_io);
      string_aux := ParamStr(1);
      estado_reporte := string_aux[1];
      if estado_reporte = negativo then
         titulo1 := 'REPORTE DE GIROS POR GENERAR'
      else
         titulo1 := 'REPORTE DE GIROS GENERADOS';
      titulo2 := 'CON FECHA VALOR ' + FECHA_TEXTO(fecha_sistema);
      titulo3 := 'Y ' + FECHA_TEXTO(fecha_expedicion);
      titulo4 := 'REP_GEN';
      end; {end inicializa}
```

```
PROCEDURE FIN_REPORTE;
{ termina reporte y cierra archivos }
   var
     status_io : integer;

   begin
   status_io := CIERRAS(base);
   if status_io <> ok_io then
      DESPLIEGA_ERROR('REP_GEN.CIERRAS', status_io);
   TERMINA_REPORTE;
   end; {end fin_reporte}

PROCEDURE INICIALIZA_R_GENERACION;
{ inicializa valores }
BEGIN
   total_monto_giros := 0.0;
   total_folios_giros:= 0;
   pagina:=0;
END; { PROCEDURE INICIALIZA_R_GENERACION}


PROCEDURE TITULOS_R_GENERACION;
{ imprime titulo }
BEGIN
   ENCABEZADO;
   if estado_reporte = negativo then
      begin
      IMPRIME_LINEA(0,'+' + PadCh('',doble_linea,10) +
                      '-' + PadCh('', doble_linea, 16) +
                      '-' + PadCh('', doble_linea, 12) +
                      '-' + PadCh('', doble_linea, 12) +
                      '-' + PadCh('', doble_linea, 6) +
                      '-' + PadCh('', doble_linea, 152) +
                      '-' + Padch('', doble_linea, 21) + '+');

      IMPRIME_LINEA(0, '¦' + Pad('FOLIO', 10) +
                       '¦' + 'FECHA EXPEDICION' +
                       '¦' + ' CTA. ABONO ' +
                       '¦' + ' CTA. CARGO ' +
                       '¦' + 'DIVISA' +
                       '¦' + Pad(' BENEFICIARIO', 152) +
                       '¦' + Pad(' MONTO',21) + '¦');

      IMPRIME_LINEA(0,'¦' + PadCh('', doble_linea, 10) +
                      '+' + PadCh('', doble_linea, 16) +
                      '+' + PadCh('', doble_linea, 12) +
                      '+' + PadCh('', doble_linea, 12) +
                      '+' + PadCh('', doble_linea, 6) +
                      '+' + PadCh('', doble_linea, 152) +
                      '+' + PadCh('', doble_linea, 21) + '¦');
      end
   else
      begin
      IMPRIME_LINEA(0,'+' + PadCh('',doble_linea,10) +
                      '-' + PadCh('', doble_linea, 16) +
                      '-' + PadCh('', doble_linea, 14) +
                      '-' + PadCh('', doble_linea, 12) +
                      '-' + PadCh('', doble_linea, 12) +
                      '-' + PadCh('', doble_linea, 6) +
                      '-' + PadCh('', doble_linea, 152) +
                      '-' + Padch('', doble_linea, 21) + '+');

      IMPRIME_LINEA(0,'¦' + Pad('FOLIO', 10) +
                      '¦' + 'FECHA EXPEDICION' +
                      '¦' + 'NUMERO DE GIRO' +
                      '¦' + ' CTA. ABONO ' +
                      '¦' + ' CTA. CARGO ' +
                      '¦' + 'DIVISA' +
                      '¦' + Pad(' BENEFICIARIO', 152) +
                      '¦' + Pad(' MONTO', 21) + '¦');

      IMPRIME_LINEA(0,'¦' + PadCh('', doble_linea, 10) +
                      '+' + PadCh('', doble_linea, 16) +
                      '+' + PadCh('', doble_linea, 14) +
```

```
                        '+' + PadCh('', doble_linea, 12) +
                        '+' + PadCh('', doble_linea, 12) +
                        '+' + PadCh('', doble_linea, 6) +
                        '+' + PadCh('', doble_linea, 152) +
                        '+' + PadCh('', doble_linea, 21) + '¦');
      end;
END; { PROCEDURE TITULOS_R_GENERACION}

PROCEDURE IMPRIME_LINEA_R_GENERACION;
{ imprime linea }
BEGIN
   if estado_reporte = negativo then
      IMPRIME_LINEA(0,'¦' + PadCh('', linea_sencilla, 10) +
                        '+' + PadCh('', linea_sencilla, 16) +
                        '+' + PadCh('', linea_sencilla, 12) +
                        '+' + PadCh('', linea_sencilla, 12) +
                        '+' + PadCh('', linea_sencilla, 6) +
                        '+' + PadCh('', linea_sencilla, 152) +
                        '+' + PadCh('', linea_sencilla, 21) + '¦')
   else
      IMPRIME_LINEA(0,'¦' + PadCh('', linea_sencilla, 10) +
                        '+' + PadCh('', linea_sencilla, 16) +
                        '+' + PadCh('', linea_sencilla, 14) +
                        '+' + PadCh('', linea_sencilla, 12) +
                        '+' + PadCh('', linea_sencilla, 12) +
                        '+' + PadCh('', linea_sencilla, 6) +
                        '+' + PadCh('', linea_sencilla, 152) +
                        '+' + PadCh('', linea_sencilla, 21) + '¦');
END;

PROCEDURE CIERRA_CUADRO_R_GENERACION;
{ imprime lineas para cerrar cuedro }
BEGIN
   if estado_reporte = negativo then
      IMPRIME_LINEA(0,'+' + PadCh('', doble_linea, 10) +
                        '-' + PadCh('', doble_linea, 16) +
                        '-' + PadCh('', doble_linea, 12) +
                        '-' + PadCh('', doble_linea, 12) +
                        '-' + PadCh('', doble_linea, 6) +
                        '-' + PadCh('', doble_linea, 152) +
                        '-' + PadCh('', doble_linea, 21) + '+')
   else
      IMPRIME_LINEA(0,'+' + PadCh('', doble_linea, 10) +
                        '-' + PadCh('', doble_linea, 16) +
                        '-' + PadCh('', doble_linea, 14) +
                        '-' + PadCh('', doble_linea, 12) +
                        '-' + PadCh('', doble_linea, 12) +
                        '-' + PadCh('', doble_linea, 6) +
                        '-' + PadCh('', doble_linea, 152) +
                        '-' + PadCh('', doble_linea, 21) + '+');
END;

PROCEDURE IMPRIME_CUADRO_FIRMAS;
{ imprime cuadro para cifras totales }
   begin
   IMPRIME_LINEA(2,'+' + PadCh('', doble_linea, 20) +
                     '-' + PadCh('', doble_linea, 41) + '+');
   IMPRIME_LINEA(0,'¦' + Center('CONTROL', 20) +
                     '¦' + Center('TRAMITE DE CAMBIOS', 41) + '¦');
   IMPRIME_LINEA(0,'¦' + PadCh('', doble_linea, 20) +
                    '+' + PadCh('', doble_linea, 20) + '-' +
                        + PadCh('', doble_linea, 20) + '¦');

   IMPRIME_LINEA(0,'¦' + Pad(' PROTEGIO', 20) +
                     '¦' + Pad(' REQUISITO', 20) +
                     '¦' + Pad(' AUTORIZO', 20) + '¦');
   IMPRIME_LINEA(0,'¦' + Pad('', 20) +
                     '¦' + Pad('', 20) +
                     '¦' + Pad('', 20) + '¦');
   IMPRIME_LINEA(0,'¦' + Pad('', 20) +
                     '¦' + Pad('', 20) +
                     '¦' + Pad('', 20) + '¦');
   IMPRIME_LINEA(0,'¦' + Pad('', 20) +
                     '¦' + Pad('', 20) +
                     '¦' + Pad('', 20) + '¦');
```

```
    IMPRIME_LINEA(0,'¦' + Pad('', 20) +
                    '¦' + Pad('', 20) +
                    '¦' + Pad('', 20) + '¦');
    IMPRIME_LINEA(0,'+' + PadCh('', doble_linea, 20) +
                    '-' + PadCh('', doble_linea, 20) +
                    '-' + PadCh('', doble_linea, 20) + '+');
    end; {end cuadro firmas}


PROCEDURE IMPRIME_TOTALES_R_GENERACION;
{ imprime cifras totales }
BEGIN
   if estado_reporte = negativo then
      begin
      IMPRIME_LINEA(0,'¦' + PadCh('', doble_linea, 10) +
                      '-' + PadCh('', doble_linea, 16) +
                      '-' + PadCh('', doble_linea, 7) + '-' + Padch('',
                                    doble_linea, 4) +
                      '-' + PadCh('', doble_linea, 12) +
                      '-' + PadCh('', doble_linea, 6) +
                      '-' + PadCh('', doble_linea, 151) + '-' +
                      '-' + PadCh('', doble_linea, 21) + '¦');
      IMPRIME_LINEA(0,'¦ Número de Registros =     ' + form('########',
                       total_folios_giros)+ ' ¦' + Pad('', 150) +
                       'CUENTAHABIENTE: ' +
                       llave_ant.cuentahabte_girk +
                      '¦  '+ form(fmto_monto,total_monto_giros) + '
                       ¦');
      IMPRIME_LINEA(0,'+-----------------------------------+' +
                      Pad('', 176) +
                      '+----------------------+');
      end
   else
      begin
      IMPRIME_LINEA(0,'¦' + PadCh('', doble_linea, 10) +
                      '-' + PadCh('', doble_linea, 16) +
                      '-' + PadCh('', doble_linea, 7) + '-' + Padch('',
                                    doble_linea, 6) +
                      '-' + PadCh('', doble_linea, 12) +
                      '-' + PadCh('', doble_linea, 12) +
                      '-' + PadCh('', doble_linea, 6) +
                      '-' + PadCh('', doble_linea, 151) + '-' +
                      '-' + PadCh('', doble_linea, 21) + '¦');

      IMPRIME_LINEA(0,'¦ Número de Registros =     ' +
                      form('########', total_folios_giros) + ' ¦'+
                      Pad( '', 165) +
                      'CUENTAHABIENTE: '+
                       llave_ant.cuentahabte_girk +
                      '¦  '+ form(fmto_monto,total_monto_giros) + '
                       ¦');
      IMPRIME_LINEA(0,'+-----------------------------------+'+
                      Pad('', 191) +
                      '+----------------------+');
      end;
      if estado_reporte = generado then
         begin
         if contrenglon + 11 > renglones_x_hoja then
            ENCABEZADO;
         IMPRIME_CUADRO_FIRMAS
         end;
END; { PROCEDURE IMPRIME_TOTALES_R_GENERACION}


PROCEDURE IMPRIME_FIJA_R_GENERACION;
{ imprime lineas para los registros }
VAR
  string_aux : string;

BEGIN
   with wr_giros do
      begin
      if estado_reporte = negativo then
         begin
         string_aux := '¦' + Center(folio_gir, 10) +
```

```
                        '¦' + Center(TPOFECHA_FECHA(fecha_gir),16)  +
                        '¦' + Center(corresponsal_gir, 12) +
                        '¦' + Center(copy(cuenta_cargo_gir, 1, 10),12) +
                        '¦' + Center(divisa_gir,6) +
                        '¦' + Pad(' ' + Trim(beneficiario_gir), 152) +
                        '¦' + ' ' + form(fmto_monto,monto_gir) + '  ¦';
          end
       else
          begin
          string_aux := '¦' + Center(folio_gir, 10) +
                        '¦' + Center(TPOFECHA_FECHA(fecha_gir), 16) +
                        '¦' + Center(numero_cheque_gir, 14)  +
                        '¦' + Center(corresponsal_gir, 12) +
                        '¦' + Center(copy(cuenta_cargo_gir, 1, 10),12) +
                        '¦' + Center(divisa_gir, 6) +
                        '¦' + Pad(' ' + Trim(beneficiario_gir), 152) +
                        '¦' + ' ' + form(fmto_monto,monto_gir) + '  ¦';
          if estado_gir = cancelado then
             string_aux := string_aux + '***';
          end;
       IMPRIME_LINEA(0,string_aux);
       end;
    inc(total_folios_giros);
    total_monto_giros := total_monto_giros + wr_giros.monto_gir;
END; { PROCEDURE IMPRIME_FIJA_R_GENERACION}

FUNCTION IMPRIME_REGISTRO(wr_registro : r_giros): boolean;
{ imprime registro }
    VAR
      si_imprime : boolean;
    begin
    with wr_giros do
       begin
       si_imprime := ((fecha_gir <= fecha_expedicion) or
                      ((fecha_gir = fecha_sistema) and
                       (fecha_cap_gir = fecha_sistema)));
       si_imprime := si_imprime and
                     ((estado_gir = estado_reporte) or
                      ((estado_gir = cancelado) and
                       (estado_reporte = generado) and
                       (fecha_gir = fecha_expedicion)));
       end;
    IMPRIME_REGISTRO := si_imprime;
    end; {end imprime_registro}

PROCEDURE LEE_PRIMERO(var wr_giros : r_giros;
                      var llave    : llave2_giros;
                      var status   : integer);
{ lee el primer registro del archivo de giros }
   begin
   LEE2_PRIMER_GIROS(wr_giros, llave, status);
   if (status <> ok_io) and (status <> fin_archivo) then
      DESPLIEGA_ERROR('REP_GEN.LEE2_PRIMER_GIROS', status);
   while not IMPRIME_REGISTRO(wr_giros) and
             (status = ok_io) do
      begin
      LEE2_SIG_GIROS(wr_giros, llave, status);
      if (status <> ok_io) and (status <> fin_archivo) then
         DESPLIEGA_ERROR('REP_GEN.LEE2_SIG_GIROS-1', status);
      end; {end while}
   end; {end lee_primero}

PROCEDURE LEE_SIGUIENTE(var wr_giros : r_giros;
                        var llave    : llave2_giros;
                        var status   : integer);
{ lee el siguiente registro del archivo de giros }
   begin
   repeat
      LEE2_SIG_GIROS(wr_giros, llave, status);
      if (status <> ok_io) and (status <> fin_archivo) then
         DESPLIEGA_ERROR('REP_GEN.LEE2_SIG_GIROS-2', status);
   until IMPRIME_REGISTRO(wr_giros) or
         (status <> ok_io);
   end; {end lee_siguiente}
```

```
PROCEDURE GENERACION_R_GIROS;
{ imprime el reporte }
BEGIN
   INICIALIZA_R_GENERACION;
   TITULOS_R_GENERACION;
   while status_io = ok_io do
      begin
      with wr_giros do
        begin
        IMPRIME_FIJA_R_GENERACION;
        LEE_SIGUIENTE(wr_giros, llave_giros, status_io);
        if status_io = fin_archivo then
           IMPRIME_TOTALES_R_GENERACION
        else
           if (status_io = ok_io) then
              if (llave_ant.cuentahabte_girk =
                    llave_giros.cuentahabte_girk) and
                 (((llave_ant.divisa_girk <> dolar_americano) and
                   (llave_giros.divisa_girk <> dolar_americano)   ) or
                  ((llave_ant.divisa_girk = dolar_americano) and
                   (llave_giros.divisa_girk = dolar_americano)  )      )
              then
                begin
                if contrenglon + 3 < renglones_x_hoja then
                   IMPRIME_LINEA_R_GENERACION
                else
                   begin
                   CIERRA_CUADRO_R_GENERACION;
                   TITULOS_R_GENERACION;
                   end;
                end
              else
                 begin
                 IMPRIME_TOTALES_R_GENERACION;
                 llave_ant := llave_giros;
                 TITULOS_R_GENERACION;
                 total_folios_giros := 0;
                 total_monto_giros := 0;
                 end;
        end;{end with}
        if (status_io <> ok_io) and (status_io <> fin_archivo) then
           DESPLIEGA_ERROR('REP_GEN.LEE_SIG_GIROS',status_io);
        end; {end while}
END; { PROCEDURE GENERACION_R_GIROS}

PROCEDURE GENERA_REPORTE;
{ genera el reporte }
BEGIN
   MENSAJE('AVISO',' IMPRIMIENDO REPORTE DE GENERACION', 21, aviso);
   LEE_PRIMERO(wr_giros, llave_giros, status_io);
   if (status_io <> fin_archivo) then
      begin
      llave_ant := llave_giros;
      GENERACION_R_GIROS;
      OTRA_HOJA_CON_ESPERA;
      end
   else
      begin
      if estado_reporte = negativo then
         IMPRIME_LINEA(0,'NO EXISTEN GIROS POR GENERAR CON FECHA ' +
fecha_rep)
      else
         IMPRIME_LINEA(0,'NO SE GENERARON GIROS CON FECHA ' +
fecha_rep);
      OTRA_HOJA_CON_ESPERA;
      end;
END; { PROCEDURE GENERA_REPORTE }


BEGIN      { BEGIN   REP_GEN   procedimiento principal }
   INICIALIZA;
   FECHA_REPORTES(fecha_sistema,fecha_rep);
   GENERA_REPORTE;
   FIN_REPORTE;
END. { REP_GEN }
```

Programa que genera el reporte de captura:

```
{$I-,V-,R-,N+,E+}
PROGRAM REP_CAP;
{  Produce  el reporte de Giros Capturados }

USES
   UGRALGIR,          { funciones generales }
   DOS,               { funciones del DOS }
   TPCRT,             { manejo de teclado, pantalla, colores y sonido }
   TPSTRING,          { manejo de strings }
   COLORDEF,          { definicion de colores para pantallas }
   PRINTER,           { manejo de la impresora}
   UDCLGIRO,          { manejo de declaraciones globales }
   UGRALGIR,          { rutinas generales }
   UIMPRIME,          { rutinas de impresion }
   UMSJGIRO,          { rutinas de mensajes a pantalla }
   URUTGIRO,          { rutinas generales }
   UGIROS;            { manejo de la cartera del sistema }

CONST
  doble_linea     = '=';
  linea_sencilla  = '-';
  archivo_salida  = 'c:\giros\datos\rep_cap.lst';
  fmto_monto      = '###,###,###,###.##';

VAR
   total_folios_giros : integer;
   total_monto_giros  : extended;
   fecha_sistema      : tpo_fecha;
   fecha_rep          : tpo_fecha_reporte;
   wr_giros           : r_giros;
   llave_giros        : llave2_giros;
   llave_ant          : llave2_giros;
   status_io          : integer;
   estado_reporte     : char;

   PROCEDURE INICIALIZA;
   { inicializa variables y abre archivos }
      VAR
         wtu,
         status_io : integer;
         string_aux : string;

      begin
      base := nil;
      for wtu := 1 to 255 do
          blancos[wtu] := ' ';
      status_io := ABRES(base, 'GIROS', abre_giros, modo_lectura,
                         cierra_giros);
      if status_io <> ok_io then
         DESPLIEGA_ERROR('REP_CAP.ABRE_GIROS', status_io);
      INICIA_REPORTE(archivo_salida);
      CONTROL_DE_IMPRESORAS (margen_superior,margen_izquierdo, carta,
                             o_horizontal, cpp_16, lpp_8, 230);
      renglones_x_hoja := 60;
      FECHA_HOY(fecha_sistema, status_io);
      if status_io <> ok_io then
         DESPLIEGA_ERROR('REP_GEN.FECHA_HOY', status_io);
      titulo1 := '';
      titulo2 := 'REPORTE DE GIROS CAPTURADOS';
      titulo3 := 'EL DIA ' + FECHA_TEXTO(fecha_sistema);
      titulo4 := '';
      end; {end inicializa}

   PROCEDURE FIN_REPORTE;
   { termina reporte y cierra archivos }
      VAR
        status_io : integer;

      begin
      status_io := CIERRAS(base);
      if status_io <> ok_io then
```

```
      DESPLIEGA_ERROR('REP_CAP.CIERRAS', status_io);
   TERMINA_REPORTE;
   end; {end fin_reporte}

PROCEDURE INICIALIZA_R_CAPTURA;
{inicializa variables }
BEGIN
   total_monto_giros := 0.0;
   total_folios_giros:= 0;
   pagina:=0;
END; { PROCEDURE INICIALIZA_R_CAPTURA}


PROCEDURE TITULOS_R_CAPTURA;
{ imprime titulo }
BEGIN
   ENCABEZADO;
   IMPRIME_LINEA(0,'+' + PadCh('', doble_linea, 10) +
                   '-' + PadCh('', doble_linea, 16) +
                   '-' + PadCh('', doble_linea, 12) +
                   '-' + PadCh('', doble_linea, 12) +
                   '-' + PadCh('', doble_linea, 6) +
                   '-' + PadCh('', doble_linea, 152) +
                   '-' + PadCh('', doble_linea, 21) + '+');

   IMPRIME_LINEA(0,'¦' + Center('FOLIO',10) +
                   '¦' + 'FECHA EXPEDICION' +
                   '¦' + 'CORRESPONSAL' +
                   '¦' + ' CTA. CARGO ' +
                   '¦' + 'DIVISA' +
                   '¦' + Pad(' BENEFICIARIO', 152) +
                   '¦' + Pad(' MONTO', 21) + '¦');
   IMPRIME_LINEA(0,'¦' + PadCh('', doble_linea, 10) +
                   '+' + PadCh('', doble_linea, 16) +
                   '-' + PadCh('', doble_linea, 12) +
                   '+' + PadCh('', doble_linea, 12) +
                   '+' + PadCh('', doble_linea, 6) +
                   '+' + PadCh('', doble_linea, 152) +
                   '+' + PadCh('', doble_linea, 21) + '¦');
END; { PROCEDURE TITULOS_R_CAPTURA}

PROCEDURE IMPRIME_LINEA_R_CAPTURA;
{ imprime linea }
BEGIN
   IMPRIME_LINEA(0,'¦' + PadCh('', linea_sencilla, 10) +
                   '+' + PadCh('', linea_sencilla, 16) +
                   '+' + PadCh('', linea_sencilla, 12) +
                   '+' + PadCh('', linea_sencilla, 12) +
                   '+' + PadCh('', linea_sencilla, 6) +
                   '+' + PadCh('', linea_sencilla, 152) +
                   '+' + PadCh('', linea_sencilla, 21) + '¦');
END;

PROCEDURE CIERRA_CUADRO_R_CAPTURA;
{ cierra cuadro de impresion }
BEGIN
   IMPRIME_LINEA(0,'+' + PadCh('', doble_linea, 10) +
                   '-' + PadCh('', doble_linea, 16) +
                   '-' + PadCh('', doble_linea, 12) +
                   '-' + PadCh('', doble_linea, 12) +
                   '-' + PadCh('', doble_linea, 6) +
                   '-' + PadCh('', doble_linea, 152) +
                   '-' + PadCh('', doble_linea, 21) + '+');
END;

PROCEDURE IMPRIME_TOTALES_R_CAPTURA;
{ imprime totales del reporte }
BEGIN
   IMPRIME_LINEA(0,'¦' + PadCh('', doble_linea, 10) +
                   '-' + PadCh('', doble_linea, 16) +
                   '-' + PadCh('', doble_linea, 5) + '-' + PadCh('',
                     doble_linea, 6) +
                   '-' + PadCh('', doble_linea, 12) +
                   '-' + PadCh('', doble_linea, 6) +
                   '-' + PadCh('', doble_linea, 151) +  '-' +
```

```
                    '-' + PadCh('', doble_linea, 21) + '|');
IMPRIME_LINEA(0,'| Número de Registros =  ' + form('########',
                total_folios_giros)+
                ' |' + Pad('', 151) +
                'CUENTAHABIENTE : '+ llave_ant.cuentahabte_girk +
               '|  '+ form(fmto_monto,total_monto_giros) + '  |');
IMPRIME_LINEA(0,'+-----------------------------------+' +
                Pad('', 178) +
                '+----------------------+');
END; { PROCEDURE IMPRIME_TOTALES_R_CAPTURA}

PROCEDURE IMPRIME_FIJA_R_CAPTURA;
{ imprime registro }
VAR
  string_aux : string;

BEGIN
   with wr_giros do
      begin
      string_aux := '|' + Center(folio_gir,10) +
                    '|' + Center(TPOFECHA_FECHA(fecha_gir), 16) +
                    '|' + ' ' + corresponsal_gir + ' ' +
                    '|' + Center(copy(cuenta_cargo_gir, 1, 10), 12) +
                    '|' + Center(divisa_gir, 6) +
                    '|' + Pad(' ' + Trim(beneficiario_gir), 152) +
                    '|' + ' ' + form(fmto_monto,monto_gir) + '  ' + '|';
      IMPRIME_LINEA(0,string_aux);
      end;
   inc(total_folios_giros);
   total_monto_giros := total_monto_giros + wr_giros.monto_gir;
(*   inc(num_reg); *)
END; { PROCEDURE IMPRIME_FIJA_R_CAPTURA}

PROCEDURE LEE_PRIMERO(var wr_giros : r_giros;
                      var llave    : llave2_giros;
                      var status   : integer);
{ lee el primer registro del archivo de giros }
   begin
   LEE2_PRIMER_GIROS(wr_giros, llave, status);
   if (status_io <> ok_io) and (status_io <> fin_archivo) then
      DESPLIEGA_ERROR('REP_GEN.LEE2_PRIMER_GIROS', status_io);
   while (wr_giros.fecha_cap_gir <> fecha_sistema) and
         (status_io <> fin_archivo)  do
      begin
      LEE2_SIG_GIROS(wr_giros, llave, status);
      if (status_io <> ok_io) and (status_io <> fin_archivo) then
         DESPLIEGA_ERROR('REP_GEN.LEE2_SIG_GIROS-1', status_io);
      end; {end while}
   end; {end lee_primero}

PROCEDURE LEE_SIGUIENTE(var wr_giros : r_giros;
                        var llave    : llave2_giros;
                        var status   : integer);
{ lee el siguiente registro del archivo de giros }
   begin
   repeat
      LEE2_SIG_GIROS(wr_giros, llave, status);
      if (status_io <> ok_io) and (status_io <> fin_archivo) then
         DESPLIEGA_ERROR('REP_GEN.LEE2_SIG_GIROS-2', status_io);
   until (wr_giros.fecha_cap_gir = fecha_sistema) or
         (status_io = fin_archivo);
   end; {end lee_siguiente}

PROCEDURE GENERACION_R_CAPTURA;
{ imprime el reporte de captura }
BEGIN
   INICIALIZA_R_CAPTURA;
   TITULOS_R_CAPTURA;
   while status_io = ok_io do
      begin
      IMPRIME_FIJA_R_CAPTURA;
      LEE_SIGUIENTE(wr_giros, llave_giros, status_io);
      if status_io = fin_archivo then
         IMPRIME_TOTALES_R_CAPTURA
      else
```

```
            if (status_io = ok_io) then
               if (llave_ant.cuentahabte_girk =
                     llave_giros.cuentahabte_girk) and
                  (llave_ant.folio_girk <> llave_giros.folio_girk)
                  then
                  begin
                  if contrenglon + 3 < renglones_x_hoja then
                     IMPRIME_LINEA_R_CAPTURA
                  else
                    begin
                    CIERRA_CUADRO_R_CAPTURA;
                    TITULOS_R_CAPTURA;
                    end;
                  end
               else
                  begin
                  IMPRIME_TOTALES_R_CAPTURA;
                  llave_ant := llave_giros;
                  TITULOS_R_CAPTURA;
                  total_folios_giros := 0;
                  total_monto_giros := 0;
                  end;
         end; {end while}
      if (status_io <> ok_io) and (status_io <> fin_archivo) then
         DESPLIEGA_ERROR('REP_GEN.LEE_SIG_GIROS',status_io);
   END; { PROCEDURE GENERACION_CAPTURA}

   PROCEDURE GENERA_REPORTE;
   { genera el reporte de captura }
   BEGIN
      MENSAJE('AVISO',' IMPRIMIENDO REPORTE DE CAPTURA', 21, aviso);
      LEE_PRIMERO(wr_giros, llave_giros, status_io);
      if (status_io <> fin_archivo) then
         begin
         llave_ant := llave_giros;
         GENERACION_R_CAPTURA;
         OTRA_HOJA_CON_ESPERA;
         end
      else
         begin
         IMPRIME_LINEA(0,'NO EXISTEN GIROS CAPTURADOS CON FECHA ' +
                  fecha_rep);
         OTRA_HOJA_CON_ESPERA;
         end;
   END; { PROCEDURE GENERA_REPORTE }

BEGIN     { BEGIN   REP_CAP   procedimiento principal }
   INICIALIZA;
   FECHA_REPORTES(fecha_sistema,fecha_rep);
   GENERA_REPORTE;
   FIN_REPORTE;
END. { REP_CAP}
```

Listado del programa que genera el reporte de giros cancelados:

```
{$I-,V-,R-,N+,E+}
PROGRAM REP_ERR;
{  Produce  el reporte de Giros erroneos }

USES
   DOS,                 { funciones del DOS }
   TPCRT,               { manejo de teclado, pantalla, colores y sonido }
   TPSTRING,            { manejo de strings }
   PRINTER,             { manejo de la impresora}
   UBTRIEVE,            { manejo de archivos }
   UCONTROL,            { manejo del archivo de control del sistema }
   UDCLGIRO,            { manejo de declaraciones globales }
   UGRALGIR,            { rutinas generales }
   UIMPRIME,            { rutinas de impresion }
   UMSJGIRO,            { rutinas de mensajes a pantalla }
   UERRORES;            { manejo del archivo de giros cancelados }

CONST
  archivo_salida = 'f:\giros\datos\rep_err.lst';
  reg_por_hoja = 60;
  fmto_monto   = '###,###,###,###.##';

VAR
   fecha_sistema       : tpo_fecha;
   fecha_rep           : tpo_fecha_reporte;
   wr_errores          : r_errores;
   llave_anterior,
   llave_errores       : llavel_errores;
   status_io,
   num_reg             : integer;


   PROCEDURE INICIALIZA;
   { inicializa valores y abre archivos }
      VAR
         wtu,
         status_io : integer;

      begin
      base := nil;
      for wtu := 1 to 255 do
         blancos[wtu] := ' ';
      status_io := ABRES(base, 'ERRORES', abre_errores, modo_lectura,
                         cierra_errores);
      if status_io <> ok_io then
         DESPLIEGA_ERROR('REP_ERR.ABRE_ERRORES', status_io);
      INICIA_REPORTE(archivo_salida);
      CONTROL_DE_IMPRESORAS (margen_superior,margen_izquierdo, carta,
                             o_horizontal, cpp_16, lpp_8, 120);
      renglones_x_hoja := 66;
      FECHA_HOY(fecha_sistema, status_io);
      if status_io <> ok_io then
         DESPLIEGA_ERROR('REP_ERR.FECHA_HOY', status_io);
      titulo1 := 'REPORTE DE GIROS CON ERROR';
      titulo2 := 'PARA: CAJA ADMINISTRATIVA Y VALORES';
      titulo3 := '';
      titulo4 := '';
      end; {end inicializa}

   PROCEDURE FIN_REPORTE;
   { termina reporte y cierra archivos }
      VAR
        status_io : integer;

      begin
      status_io := CIERRAS(base);
      if status_io <> ok_io then
         DESPLIEGA_ERROR('REP_ERR.CIERRAS', status_io);
      TERMINA_REPORTE;
      end; {end fin_reporte}
```

```
   PROCEDURE INICIALIZA_R_ERRORES;
   { inicializa valores }
   BEGIN
     pagina:=0;
     num_reg:=0;
   END; { PROCEDURE INICIALIZA_R_ERRORES}


   PROCEDURE TITULOS_R_ERRORES;
   { imprime titulo }
   BEGIN
     ENCABEZADO;
     IMPRIME_LINEA(0,'+----------------------------------------------------
--' +
     '-------------------------------------------------------------------
--' +
     '--------------------------+');

     IMPRIME_LINEA(0,'| FOLIO    | CORRESPONSAL | DIVISA | NUMERO GIRO |
CAUSA '+
   '
|');
     IMPRIME_LINEA(0,'|----------+--------------+--------+-------------+---
----'+
     '-------------------------------------------------------------------
----'+
     '-----------------------|');
     num_reg := 5;
   END; { PROCEDURE TITULOS_R_ERRORES}

   PROCEDURE IMPRIME_LINEA_R_ERRORES;
   { imprime linea }
   BEGIN
     IMPRIME_LINEA(0,'|----------+--------------+--------+-------------+---
---' +
     '-------------------------------------------------------------------
---' +
     '-----------------------|');
     inc(num_reg);
   END;

   PROCEDURE CIERRA_CUADRO_R_ERRORES;
   { imprime lineas para cerrar cuadro }
   BEGIN
     IMPRIME_LINEA(0,'+----------------------------------------------------
---' +
     '-------------------------------------------------------------------
---' +
     '-------------------------+');
   END;


   PROCEDURE IMPRIME_FIJA_R_ERRORES;
   { imprime registro }
   VAR
    string_aux : string;

   BEGIN
     with wr_errores do
        begin
        string_aux := '| ' + folio_err + ' | ' + corresponsal_err + '   |
                      ' + divisa_err + '   |   ' + num_giro_err+ '  | '+
                    causa_err + ' |';
        IMPRIME_LINEA(0,string_aux);
        end;
     num_reg:=num_reg+1;
   END; { PROCEDURE IMPRIME_FIJA_R_ERRORES}

   PROCEDURE IMPRIME_AUTORIZACION;
   { imprime area para controles }
     begin
     IMPRIME_LINEA(0, '');
     IMPRIME_LINEA(0, '+----------------------------------------------------
                      ---------+');
```

```
  IMPRIME_LINEA(0, '| TRAMITE DE CAMBIOS                    |  INTERVENTOR
                   |');
  IMPRIME_LINEA(0, '|-------------------------------+-------------------
                   ---------|');
  IMPRIME_LINEA(0, '|                               |
                   |');
  IMPRIME_LINEA(0, '|                               |
                   |');
  IMPRIME_LINEA(0, '+-----------------------------------------------------
                   ---------+');
  IMPRIME_LINEA(4, 'c.c.p. Control de Efectivos Metales y Valores');
  IMPRIME_LINEA(0, 'Oficina');
  end; {END IMPRIME_AUTORIZACION}


PROCEDURE LEE_PRIMERO(var wr_errores: r_errores;
                      var llave    : llave1_errores;
                      var status   : integer);
{ lee el primer registro del archivo errores }
  begin
  LEE_PRIMER_ERRORES(wr_errores, llave, status);
  if (status_io <> ok_io) and (status_io <> fin_archivo) then
     DESPLIEGA_ERROR('REP_ERR.LEE_PRIMER_ERRORES', status_io);
  while (wr_errores.fecha_err <> fecha_sistema) and
        (status_io <> fin_archivo)  do
     begin
     lee_sig_errores(wr_errores, llave, status);
     if (status_io <> ok_io) and (status_io <> fin_archivo) then
        DESPLIEGA_ERROR('REP_ERR.LEE_SIG_ERRORES-1', status_io);
     end; {end while}
  end; {end lee_primero}

PROCEDURE LEE_SIGUIENTE(var wr_errores: r_errores;
                        var llave    : llave1_errores;
                        var status   : integer);
{ lee el siguiente registro del archivo errores }
  begin
  repeat
     LEE_SIG_ERRORES(wr_errores, llave, status);
     if (status_io <> ok_io) and (status_io <> fin_archivo) then
        DESPLIEGA_ERROR('REP_ERR.LEE_SIG_ERRORES-2', status_io);
  until (wr_errores.fecha_err = fecha_sistema) or
        (status_io = fin_archivo);
  end; {end lee_siguiente}

PROCEDURE GENERACION_R_ERRORES;
{ genera el reporte }
BEGIN
  INICIALIZA_R_ERRORES;
  TITULOS_R_ERRORES;
  while status_io = ok_io do
     begin
     if (wr_errores.fecha_err = fecha_sistema) and
        (llave_anterior.corresponsal_kerr =
          llave_errores.corresponsal_kerr) and
        (llave_anterior.divisa_kerr = llave_errores.divisa_kerr) then
        IMPRIME_FIJA_R_ERRORES;
     if num_reg >= reg_por_hoja then
        begin
        CIERRA_CUADRO_R_ERRORES;
        TITULOS_R_ERRORES;
        num_reg:=0;
        end;
     LEE_SIGUIENTE(wr_errores, llave_errores, status_io);
     if status_io = fin_archivo then
        CIERRA_CUADRO_R_ERRORES;
     if status_io = ok_io then
       if (llave_anterior.corresponsal_kerr =
            llave_errores.corresponsal_kerr) and
          (llave_anterior.divisa_kerr = llave_errores.divisa_kerr) then
          IMPRIME_LINEA_R_ERRORES
       else
          begin
          CIERRA_CUADRO_R_ERRORES;
          IMPRIME_AUTORIZACION;
```

```
            TITULOS_R_ERRORES;
            llave_anterior := llave_errores;
            end;
        if (status_io <> ok_io) and (status_io <> fin_archivo) then
          DESPLIEGA_ERROR('REP_ERR.LEE_SIG_ERRORES',status_io);
      end; {end while}
  END; { PROCEDURE GENERACION_ERRORES}


BEGIN     { BEGIN   REP_ERR   procedimiento principal }
  INICIALIZA;
  FECHA_REPORTES(fecha_sistema,fecha_rep);
  MENSAJE('AVISO',' IMPRIMIENDO REPORTE DE ERRORES', 21, aviso);
  LEE_PRIMERO(wr_errores, llave_errores, status_io);
  if (status_io <> fin_archivo) then
     begin
     llave_anterior := llave_errores;
     GENERACION_R_ERRORES;
     OTRA_HOJA_CON_ESPERA;
     end
  else
     begin
     IMPRIME_LINEA(0,'NO EXISTEN GIROS ERRONEOS CON FECHA ' + fecha_rep);
     OTRA_HOJA_CON_ESPERA;
     end;
  FIN_REPORTE;
END. { REP_ERR }
```

Programa para generar el reporte del catalogo de corresponsales:

```
{$I-,V-,R-,N+,E+}
PROGRAM REP_CORR;
{  Produce  el reporte del archivo de CORRESPONSALES }

USES
   DOS,              { funciones del DOS }
   TPCRT,            { manejo de teclado, pantalla, colores y sonido }
   TPSTRING,         { manejo de strings }
   PRINTER,          { manejo de la impresora}
   UCONTROL,         { manejo del archivo de control del sistema }
   UDCLGIRO,         { manejo de declaraciones globales }
   UGRALGIR,         { rutinas generales }
   UIMPRIME,         { rutinas de impresion }
   UMSJGIRO,         { rutinas de mensajes a pantalla }
   UCORRESP;         { manejo del archivo de corresponsales }

CONST
  archivo_salida = 'f:\giros\datos\reporte.lst';
  reg_por_hoja = 60;

VAR
   wr_corresp          : r_corresp;
   llave_corresp       : llavel_corr;
   status_io,
   num_reg             : integer;

   PROCEDURE INICIALIZA;
   { inicializa variables y abre archivos }
      VAR
         wtu,
         status_io : integer;
         string_aux : string;

      begin
      base := nil;
      for wtu := 1 to 255 do
          blancos[wtu] := ' ';
      status_io := ABRES(base, 'CORRESP', abre_corresp, modo_lectura,
                         cierra_corresp);
      if status_io <> ok_io then
         DESPLIEGA_ERROR('REP_CORR.ABRE_CORRESP', status_io);
      INICIA_REPORTE(archivo_salida);
      CONTROL_DE_IMPRESORAS (margen_superior,margen_izquierdo, carta,
                             o_horizontal, cpp_16, lpp_8, 120);
      renglones_x_hoja := 66;
      titulo1 := '';
      titulo2 := 'REPORTE DEL ARCHIVO DE CORRESPONSALES';
      titulo3 := '';
      titulo4 := '';
      end; {end inicializa}

   PROCEDURE FIN_REPORTE;
   { termina reporte y cierra archivos }
      VAR
        status_io : integer;

      begin
      status_io := CIERRAS(base);
      if status_io <> ok_io then
         DESPLIEGA_ERROR('REP_CORR.CIERRAS', status_io);
      TERMINA_REPORTE;
      end; {end fin_reporte}

   PROCEDURE TITULOS_R_CORRESPONSALES;
   { imprime titulo del reporte }
   BEGIN
      ENCABEZADO;
      writeln(lst, 'Clave       Nombre Corresponsal
'+
      '   Dirección Corresponsal                                Cuenta');
      num_reg := 6;
```

```
  END; { PROCEDURE TITULOS_R_CORRESPONSALES}

  PROCEDURE INICIALIZA_R_CORRESPONSALES;
  { inicializa variables }
  BEGIN
     pagina:=0;
     num_reg:=0;
  END; { PROCEDURE INICIALIZA_R_CORRESPONSALES}


  PROCEDURE IMPRIME_FIJA_R_CORRESPONSALES;
  { imprime registro }
  VAR
    string_aux : string;

  BEGIN
     with wr_corresp do
        writeln(lst, clave_corr, '  ', nombre_corr, '   ', direccion_corr,
                '   ',cuenta_corr);
     num_reg:=num_reg+1;
  END; { PROCEDURE IMPRIME_FIJA_R_CORRESPONSALES}


  PROCEDURE GENERACION_R_CORRESPONSALES;
  { imprime el reporte }
  BEGIN
     INICIALIZA_R_CORRESPONSALES;
     TITULOS_R_CORRESPONSALES;
     while status_io = ok_io do
        begin
        if num_reg >= reg_por_hoja then
           begin
           TITULOS_R_CORRESPONSALES;
           num_reg:=0;
           end
        else
           IMPRIME_FIJA_R_CORRESPONSALES;
        LEE_SIGUIENTE_CORRESP(wr_corresp, llave_corresp, status_io);
        end; {end while}
     if status_io <> fin_archivo then
        DESPLIEGA_ERROR('REP_CORR.LEE_SIG', status_io);
  END; { PROCEDURE GENERACION_CORRESPONSALES}

  PROCEDURE GENERA_REPORTE;
  { genera el reporte }
  BEGIN
     MENSAJE('AVISO',' IMPRIMIENDO REPORTE DE CORRESPONSALES', 21, aviso);
     LEE_PRIMER_CORRESP(wr_corresp, llave_corresp, status_io);
     if (status_io = ok_io) then
        begin
        GENERACION_R_CORRESPONSALES;
        OTRA_HOJA_CON_ESPERA;
        end
     else
        begin
        ENCABEZADO;
        writeln(lst,'NO EXISTEN CORRESPONSALES EN EL ARCHIVO');
        OTRA_HOJA_CON_ESPERA;
        end;
  END; { PROCEDURE GENERA_REPORTE }

BEGIN    { BEGIN REP_CORR  procedimiento principal }
  INICIALIZA;
  GENERA_REPORTE;
  FIN_REPORTE;
END. { REP_CORR}
```

Programa para generar reporte del catalogo de cuentahabientes:

```
{$I-,V-,R-,N+,E+}
PROGRAM REP_CTAH;
{  Produce el reporte del archivo de CUENTAHABIENTES }

USES
   DOS,              { funciones del DOS }
   TECRT,            { manejo de teclado, pantalla, colores y sonido }
   TPSTRING,         { manejo de strings }
   PRINTER,          { manejo de la impresora}
   UCONTROL,         { manejo del archivo de control del sistema }
   UDCLGIRO,         { manejo de declaraciones globales }
   UGRALGIR,         { rutinas generales }
   UIMPRIME,         { rutinas de impresion }
   UMSJGIRO,         { rutinas de mensajes a pantalla }
   UCTAHAB;          { manejo del archivo de cuentahabientes }

CONST
  archivo_salida = 'f:\giros\datos\reporte.lst';
  reg_por_hoja = 60;

VAR
   wr_ctahabte       : r_ctahabte;
   llave_ctahabte    : llave1_ctahabte;
   status_io,
   num_reg           : integer;

   PROCEDURE INICIALIZA;
   { inicializa y abre archivos }
      VAR
         wtu,
         status_io : integer;
         string_aux : string;

      begin
      base := nil;
      for wtu := 1 to 255 do
          blancos[wtu] := ' ';
      status_io := ABRES(base, 'CTAHABTE', abre_ctahabte, modo_lectura,
                         cierra_ctahabte);
      if status_io <> ok_io then
         DESPLIEGA_ERROR('REP_CTAH.ABRE_CTAHBTE', status_io);
      INICIA_REPORTE(archivo_salida);
      CONTROL_DE_IMPRESORAS (margen_superior,margen_izquierdo, carta,
                             o_horizontal, cpp_16, lpp_8, 120);
      renglones_x_hoja := 66;
      titulo1 := '';
      titulo2 := 'REPORTE DEL ARCHIVO DE CUENTAHABIENTES';
      titulo3 := '';
      titulo4 := '';
      end; {end inicializa}

   PROCEDURE FIN_REPORTE;
   { termina reporte y cierra archivos }
      VAR
        status_io : integer;

      begin
      status_io := CIERRAS(base);
      if status_io <> ok_io then
         DESPLIEGA_ERROR('REP_CTAH.CIERRAS', status_io);
      TERMINA_REPORTE;
      end; {end fin_reporte}

   PROCEDURE TITULOS_R_CTAHABTE;
   { imprime titulo del reporte }
   BEGIN
      ENCABEZADO;
      writeln(lst, 'Cuentahabiente  Nombre Cuentahabiente
                    ' + '
                    '+'   Cuenta Tesofe          Cuenta
                    Divisa');
```

```
    num_reg := 6;
  END; { PROCEDURE TITULOS_R_CTAHBTE}

  PROCEDURE INICIALIZA_R_CTAHABTE;
  { inicializa variables }
  BEGIN
     pagina :=0;
     num_reg:=0;
  END; { PROCEDURE INICIALIZA_R_CTAHABTE}


  PROCEDURE IMPRIME_FIJA_R_CTAHABTE;
  { imprime registro }
  VAR
    string_aux : string;

  BEGIN
     with wr_ctahabte do
        writeln(lst, clave_ctahabte, '      ', nombre_ctahabte, '   ',
                     cta_tesofe_ctahabte,                  '   ',
                  cuenta_ctahabte, '   ', divisa_monto_ctahabte);
     num_reg:=num_reg+1;
  END; { PROCEDURE IMPRIME_FIJA_R_CTAHABTE}


  PROCEDURE GENERACION_R_CTAHABTE;
  { imprime reporte }
  BEGIN
     INICIALIZA_R_CTAHABTE;
     TITULOS_R_CTAHABTE;
     while status_io = ok_io do
        begin
        if num_reg >= reg_por_hoja then
           begin
           TITULOS_R_CTAHABTE;
           num_reg:=0;
           end
        else
           IMPRIME_FIJA_R_CTAHABTE;
        LEE_SIGUIENTE_CTAHABTE(wr_ctahabte, llave_ctahabte, status_io);
        end; {end while}
     if status_io <> fin_archivo then
        DESPLIEGA_ERROR('REP_CTAH.LEE_SIG', status_io);
  END; { PROCEDURE GENERACION_CTAHABTE}

  PROCEDURE GENERA_REPORTE;
  { genera reporte }
  BEGIN
     MENSAJE('AVISO',' IMPRIMIENDO REPORTE DE CUENTAHABIENTES', 21,
              aviso);
     LEE_PRIMER_CTAHABTE(wr_ctahabte, llave_ctahabte, status_io);
     if (status_io = ok_io) then
        begin
        GENERACION_R_CTAHABTE;
        OTRA_HOJA_CON_ESPERA;
        end
     else
        begin
        ENCABEZADO;
        writeln(lst,'NO EXISTEN CUENTAHABIENTES EN EL ARCHIVO');
        OTRA_HOJA_CON_ESPERA;
        end;
  END; { PROCEDURE GENERA_REPORTE }

BEGIN    { BEGIN  REP_CTAH   procedimiento principal }
  INICIALIZA;
  GENERA_REPORTE;
  FIN_REPORTE;
END. { REP_CTAH}
```

Programa para generar reporte de cifras de control:

```
{$I-,V-,R-,N+,E+}
PROGRAM REP_CTRL;
{  Produce  el reporte de cifras de control de Giros }

USES
   DOS,                 { funciones del DOS }
   TPCRT,               { manejo de teclado, pantalla, colores y sonido }
   TPSTRING,            { manejo de strings }
   PRINTER,             { manejo de la impresora}
   UCONTROL,            { manejo del archivo de control del sistema }
   UDCLGIRO,            { manejo de declaraciones globales }
   UGRALGIR,            { rutinas generales }
   UIMPRIME,            { rutinas de impresion }
   UMSJGIRO,            { rutinas de mensajes a pantalla }
   URUTGIRO,            { rutinas generales }
   UNUMGIR,             { manejo del archivo de cifras de control del sistema }
   UPROGIR;             { manejo de constantes del archivo de control }

CONST
  texto1 = 'INICIO DIA  ';
  texto2 = 'USADOS      ';
  texto3 = 'FIN DIA     ';
  archivo_salida = 'f:\giros\datos\rep_ctrl.lst';

VAR
   fecha_sistema       : tpo_fecha;
   fecha_expedicion    : tpo_fecha;
   fecha_rep           : tpo_fecha_reporte;
   wr_numgir           : r_num_gir;
   llave_numgir        : llave1_num_gir;
   status_io,
   num_reg             : integer;
   chequera            : string;
   nombre_copia        : array [0..3] of string;
   copia_act,
   copias              : integer;

   PROCEDURE INICIALIZA;
   { inicializa valores y abre archivos }
      VAR
         wtu,
         status_io : integer;

      begin
      base := nil;
      for wtu := 1 to 255 do
          blancos[wtu] := ' ';
      status_io := ABRES(base, 'NUMGIRO', abre_num_gir, modo_lectura,
                         cierra_num_gir);
      if status_io <> ok_io then
         DESPLIEGA_ERROR('REP_CTRL.ABRE_NUMGIR', status_io);
      clrscr;
      ACTUALIZA_STATUS_IMPRESORA(laser_red);
      ENCABEZADO_PANTALLA('REPORTE DE CONTROL');
      INICIA_REPORTE(archivo_salida);
      CONTROL_DE_IMPRESORAS (margen_superior,margen_izquierdo, oficio,
                             o_vertical, cpp_16, lpp_8, 120);
      renglones_x_hoja := 66;
      FECHA_HOY(fecha_sistema, status_io);
      if status_io <> ok_io then
         DESPLIEGA_ERROR('REP_CTRL.FECHA_HOY', status_io);
      chequera := ParamStr(1);
      if chequera = '' then
         begin
         FECHA_SIGUIENTE(fecha_expedicion, status_io);
         if status_io <> ok_io then
            DESPLIEGA_ERROR('REP_CTRL.FECHA_HOY', status_io);
         titulo1 := 'REPORTE DE CIFRAS DE CONTROL DE GIROS';
         titulo2 := 'GIROS CON FECHA VALOR ' + FECHA_TEXTO(fecha_sistema);
         titulo3 := 'Y ' + FECHA_TEXTO(fecha_expedicion);
         titulo4 := 'REP_CTRL';
```

```
      nombre_copia[0] := 'ACUSE';
      nombre_copia[1] := 'CONTROL DE EFECTIVO, METALES Y VALORES';
      nombre_copia[2] := 'CONCILIACION DE OPS. INTERNACIONALES';
      nombre_copia[3] := 'SUPERVISOR S.I.E.G.S.E';
      end
   else
      begin
      titulo1 := 'REPORTE DE MODIFICACION DE CIFRAS ';
      titulo2 := '';
      titulo3 := '';
      titulo4 := '';
      end;
   end; {end inicializa}


PROCEDURE FIN_REPORTE;
{ termina reporte y cierra archivos }
   VAR
     status_io : integer;

   begin
   status_io := CIERRAS(base);
   if status_io <> ok_io then
      DESPLIEGA_ERROR('REP_CTRL.CIERRAS', status_io);
   TERMINA_REPORTE;
   end; {end fin_reporte}


PROCEDURE INICIALIZA_R_GENERACION;
{ inicializa variables }
BEGIN
   pagina:=0;
   num_reg:=0;
END; { PROCEDURE INICIALIZA_R_GENERACION}


PROCEDURE TITULOS_R_CONTROL(corresponsal : string);
{ imprime titulo }
BEGIN
   IMPRIME_LINEA(1,'                                    ' + corresponsal);
   IMPRIME_LINEA(0,center('+-------------------------------------------------
                          ----------------+', 120));
   IMPRIME_LINEA(0,center('| GIROS          | TOTAL | NUM. PRIMER GIRO |
                           NUM. ULTIMO GIRO |', 120));
   IMPRIME_LINEA(0,center('|----------------+-------+------------------+--
                          ----------------|', 120));
   num_reg := 4;
END; { PROCEDURE TITULOS_R_CONTROL}


PROCEDURE IMPRIME_LINEA_R_CONTROL(numero : byte);
{ imprime linea }
BEGIN
   case numero of
     1: IMPRIME_LINEA(0,center('|----------------+-------+--------------
                                --+------------------|', 120));
     2: IMPRIME_LINEA(0,center('|-------------------------------------
                                --+------------------|', 120));
   end;
END;


PROCEDURE CIERRA_CUADRO_R_CONTROL(cual : byte);
{ imprime lineas para cerrar cuadro }
BEGIN
   case cual of
     1:
       IMPRIME_LINEA(0,center('+-----------------------------------------
                               ------------------+', 120));
     2:
       IMPRIME_LINEA(0,center('+-----------------------------------------
                              ------------------+', 120));
   end;
END;
```

```
PROCEDURE IMPRIME_FIJA_R_CONTROL(texto : string; numero : longint;
                                 numero1, numero2 : tpo_folio);
{ imprime registro }
VAR
  string_aux : string;
  num_aux    : string;

BEGIN
   num_aux := Long2Str(numero);
   num_aux := Leftpad(num_aux,5);
   string_aux := '| ' + texto + ' | ' + num_aux + ' |      ' + numero1 +
                 '     |      ' + numero2 + '      |';
   IMPRIME_LINEA(0,center(string_aux, 120));
   num_reg:=num_reg+1;
END; { PROCEDURE IMPRIME_FIJA_R_CONTROL}


PROCEDURE LEE (llave_k1, llave_k2 : llave1_num_gir;
               var num_aux1 : longint;
               var folio_aux1, folio_aux2 : tpo_folio);
{ lee un registro del archivo de control de giros }
   VAR
     num1, num2 : longint;

   begin
    LEE1_NUM_GIR(wr_numgir, llave_k1, status_io);
    if status_io <> ok_io then
       DESPLIEGA_ERROR('REP_CTRL.LEE1_NUMGIR1', status_io);
    folio_aux1 := wr_numgir.numero_gir;
    LEE1_NUM_GIR(wr_numgir, llave_k2, status_io);
    if status_io <> ok_io then
       DESPLIEGA_ERROR('REP_CTRL.LEE1_NUMGIR2', status_io);
    folio_aux2 := wr_numgir.numero_gir;
    num_aux1 := 0;
    if Str2Long(folio_aux1, num1) and Str2Long(folio_aux2, num2) then
       if folio_aux2 = '00000000' then
          folio_aux1 := '00000000'
       else
          num_aux1:= num2 - num1 +1;
    end; {end lee}


PROCEDURE IMPRIME_DATOS(texto: string;
                        llave_1, llave_2, llave_3, llave_4, llave_5,
                        llave_6, llave_7: llave1_num_gir);
{ imprime reporte }
    VAR
      folio1,
      folio2   : tpo_folio;
      numero   : longint;
      string_aux: string;

    BEGIN
    TITULOS_R_CONTROL(texto);
    LEE(llave_1, llave_2, numero, folio1, folio2);
    IMPRIME_FIJA_R_CONTROL(texto1, numero, folio1, folio2);
    if cheqera = '' then
       begin
       IMPRIME_LINEA_R_CONTROL(1);
       LEE(llave_3, llave_4, numero, folio1, folio2);
       IMPRIME_FIJA_R_CONTROL(texto2, numero, folio1, folio2);
       IMPRIME_LINEA_R_CONTROL(1);
       LEE(llave_5, llave_6, numero, folio1, folio2);
       IMPRIME_FIJA_R_CONTROL(texto3, numero, folio1, folio2);
       IMPRIME_LINEA_R_CONTROL(2);
       LEE1_NUM_GIR(wr_numgir, llave_7, status_io);
       if status_io <> ok_io then
          DESPLIEGA_ERROR('LEE1_NUMGIR(errores) ' + texto, status_io);
       string_aux := Leftpad(Long2Str(wr_numgir.num_aux_num_gir), 6);
       string_aux:='|        NUMERO DE GIROS CON ERROR        |      ' +
                   string_aux + '      |';
       IMPRIME_LINEA(0,center(string_aux, 120));
       CIERRA_CUADRO_R_CONTROL(1);
       end
```

```
        else
           CIERRA_CUADRO_R_CONTROL(2);
        IMPRIME_LINEA(0, '');
        IMPRIME_LINEA(0, '');
        IMPRIME_LINEA(0, '');
        IMPRIME_LINEA(0, '');
        end; {Imprime datos}


    PROCEDURE GENERA_REPORTE;
    { genera reporte }
    BEGIN
        MENSAJE('AVISO',' IMPRIMIENDO REPORTE DE CONTROL', 21, aviso);
        INICIALIZA_R_GENERACION;
        ENCABEZADO;
        if (chequera = citi) or
           (chequera = '')  then
           IMPRIME_DATOS('CITI BANK', p_giro_inicio_citi,
                         u_giro_inicio_citi,
                      p_giro_usado_citi, u_giro_usado_citi,
                      p_giro_fin_citi, u_giro_fin_citi,
                      errores_citi);

        if (chequera = chemical) or
           (chequera = '')  then
           IMPRIME_DATOS('CHEMICAL BANK', p_giro_inicio_chemical,
                         u_giro_inicio_chemical,
                      p_giro_usado_chemical, u_giro_usado_chemical,
                      p_giro_fin_chemical, u_giro_fin_chemical,
                      errores_chemical);

        if (chequera = libre) or
           (chequera = '')  then
           IMPRIME_DATOS('FORMATOS LIBRES', p_giro_inicio_libre,
                         u_giro_inicio_libre,
                      p_giro_usado_libre, u_giro_usado_libre,
                      p_giro_fin_libre, u_giro_fin_libre,
                      errores_libre);
        IMPRIME_LINEA(0,center('+-----------------------------------------
-----------------+', 120));
        IMPRIME_LINEA(0,center('| TRAMITE DE CAMBIOS             |
                                INTERVENTOR                 |', 120));
        IMPRIME_LINEA(0,center('|-------------------------------+----------
                      -----------------|', 120));
        IMPRIME_LINEA(0,center('|                                |
                      |', 120));
        IMPRIME_LINEA(0,center('|                                |
                      |', 120));
        IMPRIME_LINEA(0,center('+-----------------------------------------
                                -----------------+', 120));
        if (chequera = '') and (copia_act <> copias) then
           IMPRIME_LINEA(3, Pad('', 29) + 'COPIA PARA: ' +
                            nombre_copia[copia_act]);
        OTRA_HOJA_CON_ESPERA;
    END; { PROCEDURE GENERA_REPORTE }

BEGIN     { BEGIN   REP_CTRL   procedimiento principal }
    INICIALIZA;
    FECHA_REPORTES(fecha_sistema,fecha_rep);
    if chequera = '' then
       copias := 4
    else
       copias := 0;
    for copia_act := 0 to copias do
       GENERA_REPORTE;
    FIN_REPORTE;
    ACTUALIZA_STATUS_IMPRESORA(ati_local);
END. { REP_CTRL}
```

Programa que genera el reporte del catalogo de divisas:

```
{$I-,V-,R-,N+,E+}
PROGRAM REP_DIV;
{  Produce  el reporte del archivo de DIVISAS }

USES
   DOS,              { funciones del DOS }
   TPCRT,            { manejo de teclado, pantalla, colores y sonido }
   TPSTRING,         { manejo de strings }
   PRINTER,          { manejo de la impresora}
   UCONTROL,         { manejo del archivo de control del sistema }
   UDCLGIRO,         { manejo de declaraciones globales }
   UGRALGIR,         { rutinas generales }
   UIMPRIME,         { rutinas de impresion }
   UMSJGIRO,         { rutinas de mensajes a pantalla }
   UDIVISA;          { manejo del archivo de divisas }

CONST
  archivo_salida = 'f:\giros\datos\reporte.lst';
  reg_por_hoja = 60;

VAR
   fecha_sistema       : tpo_fecha;
   fecha_rep           : tpo_fecha_reporte;
   wr_divisa           : r_divisa;
   llave_divisa        : llavel_div;
   status_io,
   num_reg             :integer;

   PROCEDURE INICIALIZA;
   { inicializa valores y abre archivos }
      VAR
         wtu,
         status_io : integer;
         string_aux : string;

      begin
      base := nil;
      for wtu := 1 to 255 do
          blancos[wtu] := ' ';
      status_io := ABRES(base, 'DIVISAS', abre_divisa, modo_lectura,
                         cierra_divisa);
      if status_io <> ok_io then
         DESPLIEGA_ERROR('REP_DIV.ABRE_DIVISA', status_io);
      INICIA_REPORTE(archivo_salida);
      CONTROL_DE_IMPRESORAS (margen_superior,margen_izquierdo, carta,
                             o_horizontal, cpp_16, lpp_8, 120);
      renglones_x_hoja := 66;
      FECHA_HOY(fecha_sistema, status_io);
      if status_io <> ok_io then
         DESPLIEGA_ERROR('REP_DIV.FECHA_HOY', status_io);
      titulo1 := '';
      titulo2 := 'REPORTE DEL ARCHIVO DE DIVISAS';
      titulo3 := '';
      titulo4 := '';
      end; {end inicializa}

   PROCEDURE FIN_REPORTE;
   { termina reporte y cierra archivos }
      VAR
        status_io : integer;

      begin
      status_io := CIERRAS(base);
      if status_io <> ok_io then
         DESPLIEGA_ERROR('REP_DIV.CIERRAS', status_io);
      TERMINA_REPORTE;
      end; {end fin_reporte}

   PROCEDURE TITULOS_R_DIVISAS;
   { imprime titulo }
   BEGIN
```

```
      ENCABEZADO;
      writeln(lst, 'Divisa   Nombre Divisa en Español
                    '+ '
                   Nombre Divisa Ingles');
      num_reg := 6;
   END; { PROCEDURE TITULOS_R_DIVISAS}

   PROCEDURE INICIALIZA_R_DIVISAS;
   { inicializa valores }
   BEGIN
      pagina:=0;
      num_reg:=0;
   END; { PROCEDURE INICIALIZA_R_DIVISAS}


   PROCEDURE IMPRIME_FIJA_R_DIVISA;
   { imprime registro }
   VAR
     string_aux : string;

   BEGIN
      with wr_divisa do
         writeln(lst, clave_div, '      ', nombre_esp_div, '    ',
                      nombre_ing_div);
      num_reg:=num_reg+1;
   END; { PROCEDURE IMPRIME_FIJA_R_DIVISA}


   PROCEDURE GENERACION_R_DIVISA;
   { imprime reporte }
   BEGIN
      INICIALIZA_R_DIVISAS;
      TITULOS_R_DIVISAS;
      while status_io = ok_io do
         begin
         if num_reg >= reg_por_hoja then
            begin
            TITULOS_R_DIVISAS;
            num_reg:=0;
            end
         else
            IMPRIME_FIJA_R_DIVISA;
         LEE_SIGUIENTE_DIVISA(wr_divisa, llave_divisa, status_io);
         end; {end while}
      if status_io <> fin_archivo then
         DESPLIEGA_ERROR('REP_DIV.LEE_SIG', status_io);
   END; { PROCEDURE GENERACION_DIVISAS}

   PROCEDURE GENERA_REPORTE;
   { genera reporte }
   BEGIN
      MENSAJE('AVISO',' IMPRIMIENDO REPORTE DE DIVISAS', 21, aviso);
      LEE_PRIMER_DIVISA(wr_divisa, llave_divisa, status_io);
      if (status_io = ok_io) then
         begin
         GENERACION_R_DIVISA;
         OTRA_HOJA_CON_ESPERA;
         end
      else
         begin
         ENCABEZADO;
         writeln(lst,'NO EXISTEN DIVISAS EN EL ARCHIVO');
         OTRA_HOJA_CON_ESPERA;
         end;
   END; { PROCEDURE GENERA_REPORTE }

BEGIN       { BEGIN   REP_DIV   procedimiento principal }
   INICIALIZA;
   FECHA_REPORTES(fecha_sistema,fecha_rep);
   GENERA_REPORTE;
   FIN_REPORTE;
END. { REP_DIV}
```

Programa que generar el reporte de beneficiarios:

```
{$I-,V-,R-,N+,E+}
PROGRAM REP_BEN;
{  Produce  el reporte del archivo de BENEFICIARIOS }

USES
   DOS,                 { funciones del DOS }
   TPCRT,               { manejo de teclado, pantalla, colores y sonido }
   TPSTRING,            { manejo de strings }
   TPEDIT,              { edicion de datos }
   TPENTRY,             { pantallas de captura }
   COLORDEF,            { definicion de colores para pantallas }
   PRINTER,             { manejo de la impresora}
   UCONTROL,            { manejo del archivo de control del sistema }
   UDCLGIRO,            { manejo de declaraciones globales }
   UGRALGIR,            { rutinas generales }
   UIMPRIME,            { rutinas de impresion }
   UMSJGIRO,            { rutinas de mensajes a pantalla }
   UBEN_GIR;            { manejo del archivo de beneficiarios }

CONST
  archivo_salida = 'f:\giros\datos\reporte.lst';
  reg_por_hoja = 60;
  fmto_monto = '###,###,###,###,###.##';
  lmonto     = 22;

VAR
   wr_benef             : r_benefgir;
   llave_benef          : llavel_ben;
   status_io,
   num_reg              : integer;
   nomina_str           : string[10];
   total_nomina         : extended;
   total_beneficiarios: integer;

   PROCEDURE LEE_NOMINA;
   { captura la nomina  aprocesar }
    VAR
      escape: boolean;

    BEGIN
    nomina_str := copy(blancos, 1, 10);
    ENCABEZADO_PANTALLA('REPORTE DE BENEFICIARIOS');
    Fastwrite(Pad('Esc-Todas', 80), 25,1, BlackonLTGray);
    HouseCursoratEnd := false;
    CursortoEnd := False;
    ForceUpper := true;
    ReadString('NOMINA A PROCESAR :',
             14, 14,
             10, LtGrayonBlack, BlackonLtGray, BlackonLtGray,
             escape, nomina_str);
    Fastwrite(Pad(' ', 80), 25,1, LtGrayonBlack);
    nomina_str := Trim(nomina_str);
    if escape then
       nomina_str := copy(blancos, 1, 10);
    END; {end lee_nomina}

   PROCEDURE INICIALIZA;
   { inicializa variables y abre archivos }
      VAR
         wtu,
         status_io : integer;
         string_aux : string;

      begin
      base := nil;
      for wtu := 1 to 255 do
          blancos[wtu] := ' ';
      status_io := ABRES(base, 'BEN_GIR', abre_benef, modo_lectura,
                         cierra_benef);
      if status_io <> ok_io then
         DESPLIEGA_ERROR('REP_BEN.ABRE_BENEF', status_io);
```

```
    LEE_NOMINA;
    INICIA_REPORTE(archivo_salida);
    CONTROL_DE_IMPRESORAS (margen_superior,margen_izquierdo, carta,
                           o_horizontal, cpp_16, lpp_8, 120);
    renglones_x_hoja := 60;
    titulo1 := '';
    titulo2 := 'REPORTE DEL ARCHIVO DE BENEFICIARIOS';
    titulo3 := '';
    titulo4 := '';
    total_nomina := 0;
    total_beneficiarios := 0;
    end; {end inicializa}

 PROCEDURE FIN_REPORTE;
 { termina reporte y cierra archivos }
    VAR
      status_io : integer;

    begin
    status_io := CIERRAS(base);
    if status_io <> ok_io then
       DESPLIEGA_ERROR('REP_BEN.CIERRAS', status_io);
    TERMINA_REPORTE;
    end; {end fin_reporte}

PROCEDURE TITULOS_R_BENEFICIARIOS;
{ imprime titulo del reporte }
   BEGIN
    ENCABEZADO;
    writeln(lst, 'Beneficiario    Nombre Beneficiario
                  '+'
                  '+       '                                          Monto
                  Extra');
    num_reg := 6;
   END; { PROCEDURE TITULOS_R_BENEFICIARIOS}

PROCEDURE INICIALIZA_R_BENEFICIARIOS;
{ inicializa variables }
   BEGIN
    pagina:=0;
    num_reg:=0;
   END; { PROCEDURE INICIALIZA_R_BENEFICIARIOS}


PROCEDURE IMPRIME_FIJA_R_BENEFICIARIOS;
{ imprime registro }
   VAR
   string_aux : string;

   BEGIN
    with wr_benef do
       writeln(lst, clave_ben, '      ', nombre_ben, '   ',
                    leftpad(Form(fmto_monto, monto_ben), lmonto), '   ',
                     extra_ben);
    num_reg:=num_reg+1;
    total_nomina := total_nomina + wr_benef.monto_ben;
    inc(total_beneficiarios);
   END; { PROCEDURE IMPRIME_FIJA_R_BENEFICIARIOS}

PROCEDURE IMPRIME_TOTAL_NOMINA;
{ imprime totales }
    BEGIN
    if num_reg > reg_por_hoja then
       ENCABEZADO;
    writeln(lst);
    writeln(lst);
    writeln(lst, '-------------------------------------------------------
                  ------');
    writeln(lst, 'BENEFICIARIOS EN NOMINA : ', total_beneficiarios);
    writeln(lst, 'TOTAL POR NOMINA = ', leftpad(Form(fmto_monto,
                  total_nomina), lmonto));
    END;

PROCEDURE LEE_PRIMERO(var wr_beneficiario : r_benefgir;
                      var llave           : llave1_ben;
```

```
                        var status          : integer);
  { lee el primer registro del archivo de beneficiarios }
     begin
     LEE_PRIMER_BENEF(wr_beneficiario, llave, status_io);
     if Trim(nomina_str) <> '' then
        while (status = ok_io) and
              (status <> fin_archivo) and
              (Trim(wr_beneficiario.extra_ben) <> Trim(nomina_str)) do
               LEE_SIGUIENTE_BENEF(wr_beneficiario, llave, status);
     end; {end lee_primero}


  PROCEDURE LEE_SIGUIENTE(var wr_beneficiario : r_benefgir;
                          var llave           : llave1_ben;
                          var status          : integer);

  { lee el siguiente registro del archivo de beneficiarios }
     begin
     LEE_SIGUIENTE_BENEF(wr_beneficiario, llave, status_io);
     if Trim(nomina_str) <> '' then
        while (status = 0) and
              (status <> fin_archivo) and
              (Trim(wr_beneficiario.extra_ben) <> Trim(nomina_str)) do
               LEE_SIGUIENTE_BENEF(wr_beneficiario, llave, status);
     end; {end lee_siguiente}

  PROCEDURE GENERACION_R_BENEFICIARIOS;
  { imprime el reporte de beneficiarios }
     BEGIN
      INICIALIZA_R_BENEFICIARIOS;
      TITULOS_R_BENEFICIARIOS;
      while status_io = ok_io do
         begin
         if num_reg >= reg_por_hoja then
            begin
            TITULOS_R_BENEFICIARIOS;
            num_reg:=0;
            end
         else
            IMPRIME_FIJA_R_BENEFICIARIOS;
         LEE_SIGUIENTE(wr_benef, llave_benef, status_io);
         end; {end while}
      if status_io <> fin_archivo then
         DESPLIEGA_ERROR('REP_BEN.LEE_SIG', status_io);
      if status_io = fin_archivo then
         IMPRIME_TOTAL_NOMINA;
     END; { PROCEDURE GENERACION_BENEFICIARIOS}

  PROCEDURE GENERA_REPORTE;
  { genera el reporte de beneficiarios }
     BEGIN
      MENSAJE('AVISO',' IMPRIMIENDO REPORTE DE BENEFICIARIOS', 21, aviso);
      LEE_PRIMERO(wr_benef, llave_benef, status_io);
      if (status_io = ok_io) then
         begin
         GENERACION_R_BENEFICIARIOS;
         OTRA_HOJA_CON_ESPERA;
         end
      else
         begin
         ENCABEZADO;
         writeln(lst,'NO EXISTEN BENEFICIARIOS EN EL ARCHIVO');
         OTRA_HOJA_CON_ESPERA;
         end;
     END; { PROCEDURE GENERA_REPORTE }

BEGIN    { BEGIN   REP_BEN   procedimiento principal }
   INICIALIZA;
   GENERA_REPORTE;
   FIN_REPORTE;
END. { REP_CORR}
```

Programa para realizar cancelaciones de giros:

```
{$I-,V-,R-,N+,E+}
PROGRAM MOD_CTRL;
{ para captura de giros cancelados }

USES
   UCTRLGIR,          { manejo del archivo de control de giros }
   UERRORES,          { manejo del archivo de errores }
   TPCRT,             { manejo de teclado, pantalla, colores y sonido }
   TPSTRING,          { manejo de strings }
   TPEDIT,            { edicion de datos }
   TPENTRY,           { pantallas de captura }
   COLORDEF,          { definicion de colores para pantallas }
   PRINTER,           { manejo de la impresora}
   UBTRIEVE,          { manejo de archivos }
   UCONTROL,          { manejo del archivo de control del sistema }
   UDCLGIRO,          { manejo de declaraciones globales }
   UGRALGIR,          { rutinas generales }
   UMSJGIRO,          { rutinas de mensajes a pantalla }
   UGIROS;            { manejo de la cartera del sistema }
   TPWINDOW,          { manejo de ventanas }
   UNUMGIR,           { manejo del archivo de cifras de control del sistema }
   UPROGIR;           { manejo de constantes del archivo de control }

CONST
  numero_cifras_leer    = 8;
  max_intentos          = 3;

VAR
  wr_errores            : r_errores;
  wr_num_gir            : r_num_gir;
  wr_giros              : r_giros;
  wr_ctrl_gir           : r_ctrl_gir;
  wtu,
  status_io             : integer;
  llave_num_gir         : llave1_num_gir;
  llave_giros           : llave1_giros;
  llave_ctrl_giros      : llave2_ctrl_gir;
  proceso               : longint;
  no_existe,
  press_escape          : boolean;
  errorwin              : windowptr;
  opcion                : char;

  PROCEDURE INICIALIZA;
  { inicializa variables y abre archvios }
    begin
    explode := true;
    shadow := true;
    SoundFlagW := false;
    ShadowAttr := DkGrayonCyan;
    if not makewindow(errorwin, 25, 10, 55, 20, true, true, true,
                      BlackonLtGray, BlackonLtGray, LtGrayonBlack,
'ERRORES')
     then  DESPLIEGA_ERROR('ERROR EN INICIALIZA VENTANA', 8);
    status_io := ABRES(base,'GIROS',abre_giros, modo_recuperacion,
                      cierra_giros);
    if status_io <> ok_io then
       DESPLIEGA_ERROR('CIFRAS_CONTROL: ABRE_GIROS',status_io);
    status_io := ABRES(base,'CTRL_GIROS',abre_ctrl_gir, modo_recuperacion,
                      cierra_ctrl_gir);
    if status_io <> ok_io then
       DESPLIEGA_ERROR('CIFRAS_CONTROL: CTRL_GIROS',status_io);
    status_io := ABRES(base,'NUM_GIRO',abre_num_gir, modo_recuperacion,
                      cierra_num_gir);
    if status_io <> ok_io then
       DESPLIEGA_ERROR('CIFRAS_CONTROL: ABRE NUM_GIR',status_io);
    status_io := ABRES(base,'ERRORES',abre_errores, modo_recuperacion,
                      cierra_errores);
    if status_io <> ok_io then
       DESPLIEGA_ERROR('CIFRAS_CONTROL: ABRE ERRORES',status_io);
    SetNumeric(on);
```

```
 SetPromptAttr(BlackonLtGray);
 SetStringAttr(LtGrayOnBlack);
 HouseCursorAtEnd := off;
  end;  {end inicializa}

PROCEDURE LIBERA_VENTANAS;
{ borra y libera espacio de la ventana }
  begin
  DisposeWindow(errorwin);
  end; {end libera_ventanas}


PROCEDURE OBTEN_GIRO_CANCELAR(var wr_giros : r_giros);
{ obtiebe el folio del giro a cancelar }
   VAR
      folio_giro_pregunta,
      string_aux            : string;
      dummy                 : boolean;

   begin
   if not activatewindow(errorwin) then
      DESPLIEGA_ERROR('MOD_CTRL.ACTIVA VENTANA', 1008);
   clrscr;
   no_existe := false;
   folio_giro_pregunta:= copy(blancos, 1, numero_cifras_leer);
   EditString('FOLIO GIRO :',14 , 26,numero_cifras_leer, '99999999',
              0, press_escape, folio_giro_pregunta);
   if not press_escape then
      begin
      move(blancos[1], llave_giros, sizeof(llave_giros));
      folio_giro_pregunta := Leftpadch(folio_giro_pregunta, '0',
                                       numero_cifras_leer);
      move(folio_giro_pregunta[1], llave_giros.folio_girk[1],
      sizeof(tpo_folio));
      LEE1_GE_GIROS(wr_giros, llave_giros, status_io);
      if (status_io = ok_io) or (status_io = fin_archivo) then
         begin
         if wr_giros.folio_gir <> llave_giros.folio_girk then
            begin
            _DESPLIEGA_ERROR('NO EXISTE EL FOLIO', dummy);
            no_existe := true;
            end;
         end
      else
         DESPLIEGA_ERROR('MOD_CTRL.LEE1_GE_GIROS', status_io);
      if (not press_escape) and (not no_existe) then
         if wr_giros.estado_gir = giro_generado then
            begin
            move(wr_giros.folio_gir[1], llave_ctrl_giros[1],
            sizeof(tpo_folio));
            LEE2_CTRL_GIR(wr_ctrl_gir, llave_ctrl_giros, status_io);
            if status_io <> ok_io then
               DESPLIEGA_ERROR('MOD_CTRL.EE2_CTRL_GIR', status_io);
            end
         else
            begin
            if wr_giros.estado_gir = 'N' then
               string_aux := 'EL GIRO NO SE HA GENERADO, CANCELAR POR
                                 MENU DE CAPTURA'
            else
               string_aux := 'EL GIRO YA ESTA CANCELADO';
            no_existe := true;
            MENSAJE('ERROR', string_aux, 21, error_usuario);
            end;
      end;
   end; { OBTEN GIRO CANCELAR}


PROCEDURE CAPTURA_CANCELACION(var wr_errores : r_errores);
{ captura datos del giro a cancelar }
  VAR
    dummy               : boolean;
    num_giro_pregunta,
    string_aux          : string;
```

```
begin
if not activatewindow(errorwin) then
   DESPLIEGA_ERROR('MOD_CTRL.ACTIVA VENTANA', 1008);
FastWrite('FOLIO : ', 12, 26, BlackonLtGray);
string_aux := copy(blancos, 1, sizeof(tpo_folio));
move(wr_giros.folio_gir[1], string_aux[1], sizeof(tpo_folio));
FastWrite(string_aux, 12, 34, LtGrayonBlack);
repeat
   no_existe := false;
   num_giro_pregunta := copy(blancos, 1, sizeof(tpo_folio));
   EditString('NUMERO GIRO :',14 , 26, numero_cifras_leer, '99999999',
              0, press_escape, num_giro_pregunta);
   num_giro_pregunta := leftpadch(num_giro_pregunta, '0',
                                  numero_cifras_leer);
   if num_giro_pregunta <> wr_giros.numero_cheque_gir then
       begin
       _DESPLIEGA_ERROR('EL NUMERO DEL GIRO NO COINCIDE CON EL
                         ASIGNADO AL FOLIO', dummy);
       no_existe := true;
       end;
until (press_escape) or (not no_existe);
if (not press_escape) or (not no_existe) then
   begin
   string_aux := copy(blancos, 1, sizeof(tpo_beneficiario));
   SetNumeric(off);
   repeat
     EditString('MOTIVO : ',16, 26, sizeof(tpo_beneficiario),
                '!!!!!!!!!!!!!!!!!!!!', 0, press_escape, string_aux);
   until not press_escape;
   SetNumeric(on);
   with wr_errores do
     begin
     folio_err := wr_giros.folio_gir;
     corresponsal_err := wr_giros.corresponsal_gir;
     divisa_err := wr_giros.divisa_gir;
     num_giro_err := wr_giros.numero_cheque_gir;
     move(string_aux[1], causa_err[1], sizeof(tpo_beneficiario));
     FECHA_HOY(fecha_err, status_io);
     if status_io <> ok_io then
        DESPLIEGA_ERROR('MOD_CTRL.FECHA_HOY', status_io);
     end; {end with}
   no_existe := false;
   end; {end if}
end; {end CAPTURA CANCELACION}

PROCEDURE ACTUALIZA_CANCELACION;
{ actualiza los archivos en base a la cancelacion realizada }
  VAR
     proceso_aux : longint;
     dummy       : boolean;

  begin
  with llave_giros, wr_giros do
     begin
     folio_glrk := folio_gir;
     fecha_glrk := fecha_gir;
     corresponsal_glrk := corresponsal_gir;
     divisa_glrk := divisa_gir;
     end;
  LEE1_GIROS(wr_giros, llave_giros, status_io);
  if status_io <> ok_io then
     DESPLIEGA_ERROR('MOD_CTRL.LEE1_GIROS', status_io);
  wr_giros.estado_gir := giro_cancelado;
  ACTUALIZA1_GIROS(wr_giros, status_io);
  if status_io <> ok_io then
     DESPLIEGA_ERROR('MOD_CTRL.ACTUALIZA1_GIROS', status_io);

  move(wr_giros.folio_gir[1], llave_ctrl_giros[1], sizeof(tpo_folio));
  LEE2_CTRL_GIR(wr_ctrl_gir, llave_ctrl_giros, status_io);
  if status_io <> ok_io then
     DESPLIEGA_ERROR('MOD_CTRL.LEE2_CTRL_GIR', status_io);
  wr_ctrl_gir.estado_giro_ctrgir := giro_cancelado;
  ACTUALIZA2_CTRL_GIR(wr_ctrl_gir, status_io);
  if status_io <> ok_io then
```

```
            DESPLIEGA_ERROR('MOD_CTRL.ACTUALIZA2_CTRL_GIR', status_io);
         ESCRIBE_ERRORES(wr_errores, status_io);
         if status_io <> ok_io then
            DESPLIEGA_ERROR('MOD_CTRL.ESCRIBE_ERRORES', status_io);
         if Trim(wr_giros.corresponsal_gir) = citi then
            proceso_aux := errores_citi
         else
            if Trim(wr_giros.corresponsal_gir) = chemical then
               proceso_aux := errores_chemical

            else
               proceso_aux := errores_libre;
         LEE1_NUM_GIR(wr_num_gir, proceso_aux, status_io);
         if status_io <> ok_io then
            DESPLIEGA_ERROR('MOD_CTRL.LEE1_NUM_GIR', status_io);
         inc(wr_num_gir.num_aux_num_gir);
         ACTUALIZA1_NUM_GIR(wr_num_gir, status_io);
         if status_io <> ok_io then
            DESPLIEGA_ERROR('MOD_CTRL.ACTUALIZA_NUM_GIR', status_io);
         DESPLIEGA_ERROR('CANCELACION REALIZADA', dummy);
         end; {end actualiza_cancelacion}

BEGIN                { BEGIN MOD_CTRL     procedimiento principal }
   base := nil;
   for wtu:=1 to 255 do
       blancos[wtu] := ' ';
   INICIALIZA;
   NormalCursor;
   ENCABEZADO_PANTALLA('CIFRAS CONTROL DE GIROS');
   if not displaywindow(errorwin) then
      DESPLIEGA_ERROR('ERROR EN DESPLIEGA VENTANA', 8);
   repeat
     OBTEN_GIRO_CANCELAR(wr_giros);
     if (not press_escape) and (not no_existe) then
        CAPTURA_CANCELACION(wr_errores);
     if (not press_escape) and (not no_existe) then
       ACTUALIZA_CANCELACION;
     ReadCharacter (Pad('DESEA CONTINUAR (S/N)?', 80),
                    25,1, BlackOnLtGray, ['S', 's', 'N', 'n'], opcion);
     fastwrite(pad(' ',80), 25,1 ,LtGrayonBlack);
     press_escape := (opcion in ['N', 'n']);
   until press_escape;
   status_io := CIERRAS(base);
   if status_io <> ok_io then
      DESPLIEGA_ERROR('MOD_CTRL.CIERRAS', status_io);
   LIBERA_VENTANAS;
END.    { MOD_CTRL }
```

Programa para modificar cifras de control del sistema:

```
{$I-,V-,R-,N+,E+}
PROGRAM MOD_NUM;
{ para modificar y actualizar cifras de control }

USES
   TPCRT,           { manejo de teclado, pantalla, colores y sonido }
   TPSTRING,        { manejo de strings }
   TPEDIT,          { edicion de datos }
   TPENTRY,         { pantallas de captura }
   COLORDEF,        { definicion de colores para pantallas }
   PRINTER,         { manejo de la impresora}
   UBTRIEVE,        { manejo de archivos }
   UCONTROL,        { manejo del archivo de control del sistema }
   UDCLGIRO,        { manejo de declaraciones globales }
   UGRALGIR,        { rutinas generales }
   UMSJGIRO,        { rutinas de mensajes a pantalla }
   TPWINDOW,        { manejo de ventanas }
   UNUMGIR,         { manejo del archivo de cifras de control del sistema }
   UPROGIR;         { manejo de constantes del archivo de control }

CONST
  numero_cifras_leer   = 8;

VAR
  wr_num_gir              : r_num_gir;
  wtu,
  status_io               : integer;
  llave_num_gir           : llavel_num_gir;
  ident_num1,
  ident_num2,
  ident_num3,
  ident_num4,
  proceso                 : longint;
  no_existe,
  ok_datos,
  pres_escape             : boolean;
  datoswin                : windowptr;
  opcion                  : char;
  chequera_str            : string;
  num_giros_inicio_cap,
  num_primer_giro_cap,
  num_ultimo_giro_cap,
  num_primer_giro_nvo_cap,
  num_ultimo_giro_nvo_cap,
  num_giros_nvo_cap,
  num_primer_inicio,
  num_ultimo_inicio       : longint;

  PROCEDURE INICIALIZA;
 { inicializa variables y abre archivos }
    begin
    explode := true;
    shadow := true;
    SoundFlagW := false;
    ShadowAttr := DkGrayonCyan;
    if not makeWindow(datoswin, 20, 8, 60, 23, true, true, true,
                      BlackonLtGray, BlackonLtGray, LtGrayonBlack,
                      'ACTUALIZACION CONTROL')
    then  DESPLIEGA_ERROR('ERROR EN INICIALIZA VENTANA', 8);
   status_io := ABRES(base,'NUM_GIRO',abre_num_gir, modo_recuperacion,
                      cierra_num_gir);
   if status_io <> ok_io then
      DESPLIEGA_ERROR('MOD_NUM: ABRE NUM_GIR',status_io);
   SetNumeric(on);
   SetPromptAttr(BlackonLtGray);
   SetStringAttr(LtGrayOnBlack);
   HouseCursorAtEnd := off;
    end;  {end inicializa}

  PROCEDURE LIBERA_VENTANAS;
 { borra y libera espacio de la ventana }
```

```
    begin
    DisposeWindow(datoswin);
    end; {end libera_ventanas}

  PROCEDURE CHECA_DATOS (cuales : byte);
 { checa los datos capturados con los calculados }
    VAR
       wr_num_gir1,
       wr_num_gir2          : r_num_gir;
       continuar            : boolean;
       status_io            : integer;

    BEGIN
    case cuales of
      1:
        begin
        LEE1_NUM_GIR(wr_num_gir1, ident_num1, status_io);
        if status_io <> ok_io then
          DESPLIEGA_ERROR('MOD_NUM.LEE_NUM_GIR1 ', status_io);
        LEE1_NUM_GIR(wr_num_gir2, ident_num2, status_io);
        if status_io <> ok_io then
           DESPLIEGA_ERROR('MOD_NUM.LEE_NUM_GIR2 ', status_io);
        if not( Str2Long(wr_num_gir1.numero_gir, num_primer_inicio) and
                Str2Long(wr_num_gir2.numero_gir, num_ultimo_inicio)     )
        then
           DESPLIEGA_ERROR('MOD_NUM.CHECA_DATOS1', 999);
        ok_datos := ((num_giros_inicio_cap = num_ultimo_giro_cap -
                        num_primer_giro_cap + 1) and
                     (num_giros_inicio_cap = num_ultimo_inicio -
                        num_primer_inicio + 1)) or
                    ((num_giros_inicio_cap = 0) and (num_primer_inicio =
                      0) and
                     (num_ultimo_giro_cap = 0) and (num_ultimo_inicio =
                      0));
        end;
      2:
        begin
        ok_datos := ((num_giros_nvo_cap = num_ultimo_giro_nvo_cap -
                        num_primer_inicio + 1) or
                     ((num_giros_nvo_cap = num_ultimo_giro_nvo_cap -
                         num_primer_giro_nvo_cap +1) and
                      (num_primer_inicio = 0))and
                    (num_giros_nvo_cap <> 0)  and (num_ultimo_giro_nvo_cap
                      <>0);
        end;
      end; {end case}
      if not ok_datos then
         begin
         windowrelative := false;
         _DESPLIEGA_ERROR('CIFRAS ERRONEAS', continuar);
         windowrelative := true;
         press_escape := not continuar;
         end;
    END;{end checa_datos}

  PROCEDURE CAPTURA_DATOS;
 { captura los datos }
    VAR
      dummy               : boolean;
      num_giro_pregunta,
      string_aux          : string;

    begin
    ok_datos := true;
    press_escape := false;
    windowrelative := true;
    num_giros_inicio_cap := 0;
    num_primer_giro_cap := 0;
    num_ultimo_giro_cap:= 0;
    num_ultimo_giro_nvo_cap:= 0;
    num_giros_nvo_cap:= 0;
    num_primer_giro_nvo_cap := 0;
    num_ultimo_giro_nvo_cap := 0;

    if not activatewindow(datoswin) then
```

```
   DESPLIEGA_ERROR('MOD_NUM.ACTIVA VENTANA', 1008);
ForceUpper:= true;
repeat
   chequera_str := '';
   ReadString('CHEQUERA : ',
              2, 1, sizeof(tpo_corresponsal),
              BlackonLtGray,LtGrayonBlack,LtGrayonBlack,
              press_escape,
              chequera_str);
until (Trim(chequera_str) = citi) or
      (Trim(chequera_str) = chemical) or
      (Trim(chequera_str) = libre) or
      (press_escape);
if (not press_escape) then
   begin
   if Trim(chequera_str) = citi then
      begin
      ident_num1 := p_giro_inicio_citi;
      ident_num2 := u_giro_inicio_citi;
      ident_num3 := sig_num_imprimir_citi;
      ident_num4 := p_giro_usado_citi;
      end
   else
   if Trim(chequera_str) = chemical then
      begin
      ident_num1 := p_giro_inicio_chemical;
      ident_num2 := u_giro_inicio_chemical;
      ident_num3 := sig_num_imprimir_chemical;
      ident_num4 := p_giro_usado_chemical;
      end
   else
   if Trim(chequera_str) = libre then
      begin
      ident_num1 := p_giro_inicio_libre;
      ident_num2 := u_giro_inicio_libre;
      ident_num3 := sig_num_imprimir_libre;
      ident_num4 := p_giro_usado_libre;
      end;
   Repeat
      ReadLongInt('No. GIROS AL INICIO : ',
                  4,1, sizeof(tpo_folio),
                  BlackonLtGray,LtGrayonBlack,
                  0, 99999999,
                  press_escape,
                  num_giros_inicio_cap);
      ReadLongInt('No. PRIMER GIRO AL INICIO : ',
                  6, 1, sizeof(tpo_folio),
                  BlackonLtGray,LtGrayonBlack,
                  0, 99999999,
                  press_escape,
                  num_primer_giro_cap);
      ReadLongInt('No. ULTIMO GIRO AL INICIO : ',
                  8, 1, sizeof(tpo_folio),
                  BlackonLtGray,LtGrayonBlack,
                  0, 99999999,
                  press_escape,
                  num_ultimo_giro_cap);
      if (not press_escape) then
         CHECA_DATOS(1);
   until (press_escape) or (ok_datos);

   if ok_datos then
      begin
      ReadLongInt('NUEVO No. GIROS : ',
                  10, 1, sizeof(tpo_folio),
                  BlackonLtGray, LtGrayonBlack,
                  0, 99999999,
                  press_escape,
                  num_giros_nvo_cap);
      ReadLongInt('NUEVO No. PRIMER GIRO : ',
                  12, 1, sizeof(tpo_folio),
                  BlackonLtGray,LtGrayonBlack,
                  0, 99999999,
                  press_escape,
                  num_primer_giro_nvo_cap);
```

```
            ReadLongInt('NUEVO No. ULTIMO GIRO : ',
                        14, 1, sizeof(tpo_folio),
                        BlackonLtGray,LtGrayonBlack,
                        0, 99999999,
                        press_escape,
                        num_ultimo_giro_nvo_cap);
            if (not press_escape) then
               CHECA_DATOS(2);
            end;
         end; {end escape}
      windowrelative := false;
      end; {end CAPTURA DATOS}

    PROCEDURE ACTUALIZA_DATOS;
    { actualiza archivo con nuevas cifras }
      var
         proceso_aux : longint;
         dummy       : boolean;
         wr_num_gir  : r_num_gir;
         string_aux  : string;

      begin
      LEE1_NUM_GIR(wr_num_gir, ident_num1, status_io);
      if status_io <> ok_io then
        DESPLIEGA_ERROR('MOD_NUM.LEE_NUM_GIR 1 ', status_io);
      string_aux := leftpadch(Long2Str(num_primer_giro_nvo_cap), '0',
                              sizeof(tpo_folio));
      move(string_aux[1], wr_num_gir.numero_gir[1], sizeof(tpo_folio));
      ACTUALIZA1_NUM_GIR(wr_num_gir, status_io);
      if status_io <> ok_io then
         DESPLIEGA_ERROR('MOD_NUM.ACTUALIZA_NUM_GIR 1', status_io);

      LEE1_NUM_GIR(wr_num_gir, ident_num2, status_io);
      if status_io <> ok_io then
        DESPLIEGA_ERROR('MOD_NUM.LEE_NUM_GIR 2 ', status_io);
      string_aux := leftpadch(Long2Str(num_ultimo_giro_nvo_cap), '0',
                              sizeof(tpo_folio));
      move(string_aux[1], wr_num_gir.numero_gir[1], sizeof(tpo_folio));
      ACTUALIZA1_NUM_GIR(wr_num_gir, status_io);
      if status_io <> ok_io then
         DESPLIEGA_ERROR('MOD_NUM.ACTUALIZA_NUM_GIR 2', status_io);

      LEE1_NUM_GIR(wr_num_gir, ident_num3, status_io);
      if status_io <> ok_io then
        DESPLIEGA_ERROR('MOD_NUM.LEE_NUM_GIR 3 ', status_io);
      if (wr_num_gir.numero_gir = '00000000') then
         begin
         string_aux := leftpadch(Long2Str(num_primer_giro_nvo_cap), '0',
                                 sizeof(tpo_folio));
         move(string_aux[1], wr_num_gir.numero_gir[1], sizeof(tpo_folio));
         ACTUALIZA1_NUM_GIR(wr_num_gir, status_io);
         if status_io <> ok_io then
            DESPLIEGA_ERROR('MOD_NUM.ACTUALIZA_NUM_GIR 3', status_io);

         LEE1_NUM_GIR(wr_num_gir, ident_num4, status_io);
         if status_io <> ok_io then
           DESPLIEGA_ERROR('MOD_NUM.LEE_NUM_GIR 4 ', status_io);
         string_aux := leftpadch(Long2Str(num_primer_giro_nvo_cap), '0',
                                 sizeof(tpo_folio));
         move(string_aux[1], wr_num_gir.numero_gir[1], sizeof(tpo_folio));
         ACTUALIZA1_NUM_GIR(wr_num_gir, status_io);
         if status_io <> ok_io then
            DESPLIEGA_ERROR('MOD_NUM.ACTUALIZA_NUM_GIR 4', status_io);
         end;
      end; {end actualiza_cancelacion}

BEGIN      { BEGIN MOD_NUM   procedimiento principal }
   base := nil;
   for wtu:=1 to 255 do
       blancos[wtu] := ' ';
   INICIALIZA;
   NormalCursor;
   ENCABEZADO_PANTALLA('MODIFICA CIFRAS CONTROL DE GIROS');
   if not displaywindow(datoswin) then
```

```
    DESPLIEGA_ERROR('ERROR EN DESPLIEGA VENTANA', 8);
  CAPTURA_DATOS;
  if ok_datos then
    ACTUALIZA_DATOS;
  status_io := CIERRAS(base);
  if status_io <> ok_io then
    DESPLIEGA_ERROR('MOD_NUM.CIERRAS', status_io);
  LIBERA_VENTANAS;
  if (not press_escape) then
    EJECUTA('f:\giros\sistema\rep_ctrl.exe', Trim(chequera_str), 'REPORTE
DE CONTROL',
            proceso_dummy, no, no);
END.   { MOD_NUM }
```

A continuación se muestran los programas para manejo de archivos (btrieve).

Programa para manejo del archivo de beneficiarios:

```
UNIT UBEN_GIR;

{$I-,V-,R-,N+,E+,D+}

INTERFACE

   USES
        UDCLGIRO;

   CONST

    { Nombre del archivo en MS-DOS: }
      archivo_benefgir= 'F:\GIROS\DATOS\BENEFGIR.DAT';

   TYPE
      LLAVE1_BEN  = tpo_corresponsal;

      R_BENEFGIR = RECORD
         clave_ben   : tpo_corresponsal;
         nombre_ben  : tpo_beneficiario150;
         monto_ben   : extended;
         extra_ben   : array [1..25] of char;
      END; { R_BENEFGIR}

   FUNCTION ABRE_BENEF( modo : char ) : integer;
   FUNCTION CIERRA_BENEF : integer;
   PROCEDURE ACTUALIZA_BENEF(     wr_benefgir : r_benefgir;
                              var status_io   : integer);
   PROCEDURE ESCRIBE_BENEF(    wr_benefgir : r_benefgir;
                           var status_io   : integer          );
   PROCEDURE LEE_PRIMER_BENEF(var wr_benefgir : r_benefgir;
                              var llave_ben   : llavel_ben;
                              var status_io   : integer);
   PROCEDURE LEE_SIGUIENTE_BENEF(var wr_benefgir : r_benefgir;
                                 var llave_ben   : llavel_ben;
                                    var status_io   : integer);
   PROCEDURE LEE_BENEF(var wr_benefgir : r_benefgir;
                           llave_ben   : llavel_ben;
                       var status_io   : integer                );
   PROCEDURE LEE_ULT_BENEF(var wr_benefgir : r_benefgir;
                            var status_io   : integer          );
   PROCEDURE BORRA_BENEF(var wr_benefgir : r_benefgir;
                            llave_ben   : llavel_ben;
                         var status_io   : integer);

IMPLEMENTATION

   USES
      UBTRIEVE;

   VAR

      long_reg_ben   : integer;
      bloque_pos_ben : array [1..128] of byte;

FUNCTION ABRE_BENEF( modo : char ): integer;
   { Abre el archivo de benefgir. }
   VAR
      dueno_archivo  : string[10];
      nombre_archivo : string[80];
```

```
    modo_int        : integer;
  BEGIN
  dueno_archivo := #0;
  nombre_archivo := archivo_benefgir + #0;
  case modo of
     modo_lectura      : modo_int := solo_lectura;
     modo_recuperacion : modo_int := con_recuperacion;
     end; {case}
  ABRE_BENEF:= BTRV (abre_archivo, bloque_pos_ben, dueno_archivo[1],
                     long_reg_ben, nombre_archivo[1], modo_int      );
  END; { ABRE_BENEF}


FUNCTION CIERRA_BENEF: integer;
  BEGIN
  CIERRA_BENEF:= BTRV (cierra_archivo, bloque_pos_ben, dummy,
                       dummy, dummy, dummykey );
  END;


PROCEDURE ACTUALIZA_BENEF(    wr_benefgir   : r_benefgir;
                          var status_io  : integer);
  { Update con llave 1 }
  VAR
     llave1_control : llave1_ben;
  BEGIN
  status_io := BTRV (actualiza, bloque_pos_ben, wr_benefgir,
                     long_reg_ben, llave1_control, llave1 );
  END;


PROCEDURE ESCRIBE_BENEF(    wr_benefgir   : r_benefgir;
                        var status_io  : integer);
  VAR
     llave1_control : llave1_ben;
  BEGIN
  status_io := BTRV (escribe, bloque_pos_ben, wr_benefgir,
                     long_reg_ben, llave1_control, llave1);
  END;


PROCEDURE LEE_PRIMER_BENEF(var wr_benefgir : r_benefgir;
                           var llave_ben   : llave1_ben;
                           var status_io   : integer);
  { Pone en wr_benefgir el primer registro de benefgir con llave 1
indicada. }
  BEGIN
  status_io := BTRV (get_lowest, bloque_pos_ben, wr_benefgir,
                     long_reg_ben, llave_ben, llave1        );
  END;

PROCEDURE LEE_SIGUIENTE_BENEF(var wr_benefgir : r_benefgir;
                              var llave_ben   : llave1_ben;
                              var status_io   : integer);
  { Pone en wr_benefgir el siguiente registro de benefgir con llave 1
     indicada. }
  BEGIN
  status_io := BTRV (get_next, bloque_pos_ben, wr_benefgir,
                     long_reg_ben, llave_ben, llave1        );
  END;

PROCEDURE LEE_BENEF(var wr_benefgir : r_benefgir;
                        llave_ben   : llave1_ben;
                    var status_io   : integer);
  { Pone en wr_benefgir el registro de benefgir con llave 1  indicada. }
  BEGIN
  status_io := BTRV (get_equal, bloque_pos_ben, wr_benefgir,
                     long_reg_ben, llave_ben, llave1        );
  END;


PROCEDURE LEE_ULT_BENEF(var wr_benefgir  : r_benefgir;
                        var status_io : integer);
  { Pone en wr_benefgir el siguiente registro del archivo, usando la llave
     1. }
```

```
   VAR
      llavel_control : llavel_ben;
   BEGIN
   status_io := BTRV (get_highest, bloque_pos_ben, wr_benefgir,
                      long_reg_ben, llavel_control, llavel);
   END;


PROCEDURE BORRA_BENEF(var wr_benefgir : r_benefgir;
                          llave_ben       : llavel_ben;
                      var status_io       : integer);
   { Borra el registro  wr_benefgir de benefgir con  llave 1 indicada. }
   BEGIN
   status_io := BTRV (borra, bloque_pos_ben, wr_benefgir,
                      long_reg_ben, llave_ben, llavel        );
   END;


BEGIN
   long_reg_ben:= sizeof(r_benefgir);
END. { UNIT UBENEFGIR}
```

Programa para manejo del archivo btrieve:

```
{ UNIDAD PARA EL MANEJO DE BTRIEVE }
{$I-,V-,R-,N+,E+}

UNIT UBTRIEVE;

INTERFACE
  USES
    Udclgiro;

  CONST
    abre_archivo       =  0;
    actualiza          =  3;
    borra              =  4;
    cierra_archivo     =  1;
    crea               = 14;
    escribe            =  2;
    estado_archivo     = 15;
    get_equal          =  5;
    get_greater        =  8;
    get_greater_equal  =  9;
    get_highest        = 13;
    get_last           = 13;
    get_less           = 10;
    get_less_equal     = 11;
    get_lowest         = 12;
    get_next           =  6;
    get_previous       =  7;
    set_owner          = 29;
    clear_owner        = 30;
    crea_sup_key       = 31;
    destruye_sup_key   = 32;
    llave1             =  0;       {  Primera llave de cualq. archivo}
    llave2             =  1;       {  Segunda llave de cualq. archivo}
    llave3             =  2;       {  Tercera llave de cualq. archivo}
    llave4             =  3;       {  Cuarta  llave de cualq. archivo}
    llave5             =  4;       {  Quinta  llave de cualq. archivo}
    llave6             =  5;       {  Sexta   llave de cualq. archivo}
    llave7             =  6;       {  Séptima llave de cualq. archivo}
    llave8             =  7;       {  Octava  llave de cualq. archivo}
    llave9             =  8;       {  Nóvena  llave de cualq. archivo}
    llave10            =  9;       {  Décima  llave de cualq. archivo}


    con_recuperacion   =  0;       {  Abre archivo permitiendo recuperacio
                                      aut.}
    solo_lectura       = -2;       {  Abre archivo solo para lectura}
    exclusivo          = -4;       {  Abre el archivo en forma exclusiva}
    no_esta_btrieve      = 20;     {status de btrieve no presente}
    permite_duplicados   = 1;
    llave_modificable    = 2;
    llave_normal         = 0;
    llave_segmentada     = 16;
    pagina_btrieve       = 1024;
    pagina_btrieve_512   = 512;
    reg_long_fija        = 0;
    llave_duplicada      = 5;


  VAR
    dummy              : integer; { parametro falso p/llamadas a btrieve }
    dummykey           : byte;    { parametro de num. de llave falso para
                                    llamadas a btrieve.}

  FUNCTION  BTRV                (     OP         : integer;
                                  var POS,
                                      DATA;
                                  var DATALEN    : integer;
                                  var KBUF;
                                      KEY        : integer ) : integer;
  PROCEDURE INICIO_TRANSACCION  ( var status_io : integer );
  PROCEDURE FIN_TRANSACCION     ( var status_io : integer );
```

```
    PROCEDURE ABORTA_TRANSACCION (var status_io : integer);
    PROCEDURE RESETEA_BTRIEVE;

    PROCEDURE REGISTROS_ARCHIVO (var registros : extended;
                                     bloque_pos : tpo_bloque_pos;
                                 var status_io : integer);
    PROCEDURE CREA_ARCHIVO (     path_archivo : string;
                             var bloque_pos : tpo_bloque_pos;
                             var status_io : integer);

    FUNCTION ERROR_IO              ( codigo_error  : integer ) : integer;


IMPLEMENTATION

    USES
       DOS;
    TYPE
    espec_llave = RECORD
                        posicion          : integer;
                        longitud          : integer;
                        banderas          : integer;
                        no_llave          : array [1..4] of char;
                        reservado_llave_a: array [1..2] of char;
                        reservado_llave_b: array [1..4] of char;
                  END;

    espec_archivo = RECORD
    case integer of
        1: (
                        longitud_reg    : integer;
                        tamanio_pagina : integer;
                        num_llaves      : integer;
                        no_registros    : array [1..4] of byte;
                        variable        : integer;
                        reservado_reg   : array [1..4] of char;
                        bufer_llave     : array [1..16] of espec_llave;
            );
        2:              (Especifica_bufer : integer);
                  END;




    FUNCTION  BTRV (OP:integer;            var POS,        DATA;
                    var DATALEN: integer;  var KBUF;       KEY: integer):
integer;

          { Rutina de interfaz de BTRIEVE con turbopascal}

    CONST
         PASCAL_ID      = $AAAA;                {Pascal language id}
         VAR_ID         = $6176;    {id for variable length records - 'va'}
         BTR_INT        = $7B;
         BTR2_INT       = $2F;
         BTR_OFFSET     = $0033;
         MULTI_FUNCTION     = $AB;
         ProcId: integer = 0;            { initialize to no process id }
         MULTI: boolean = false;           { set to true if BMulti is
                                             loaded }
         VSet: boolean = false;       { set to true if we have checked for
                                        BMulti }
    TYPE
        ADDR32 = record                          {32 bit address}
        OFFSET: integer;
        SEGMENT: integer;
                end;

        BTR_PARMS = record
        USER_BUF_ADDR: ADDR32;            {data buffer address}
        USER_BUF_LEN: integer;            {data buffer length}
        USER_CUR_ADDR: ADDR32;            {currency block address}
        USER_FCB_ADDR: ADDR32;            {file control block address}
```

```
    USER_FUNCTION: integer;              {Btrieve operation}
    USER_KEY_ADDR: ADDR32;               {key buffer address}
    USER_KEY_LENGTH: BYTE;               {key buffer length}
    USER_KEY_NUMBER: BYTE;               {key number}
    USER_STAT_ADDR: ADDR32;              {return status address}
    XFACE_ID: integer;                   {language interface id}
              end;

     Result = record
          AX,BX,CX,DX,BP,SI,DI,DS,ES,FLAGS: integer;
        end;

VAR
     STAT: integer;                  {Btrieve status code}
     XDATA: BTR_PARMS;               {Btrieve parameter block}
     REGS: Registers;                {register structure used on interrrupt
                                        call}
     DONE: boolean;

BEGIN
REGS.AX := $3500 + BTR_INT;
INTR ($21, REGS);
if (REGS.BX <> BTR_OFFSET) then          {make sure Btrieve is
                                            installed}
    STAT := 20
else
    begin
    if (not VSet) then    {if we haven't checked for Multi-User version}
       begin
       REGS.AX := $3000;
       INTR ($21, REGS);
       if ((REGS.AX AND $00FF) >= 3) then
          begin
             VSet := true;
             REGS.AX := MULTI_FUNCTION * 256;
             INTR (BTR2_INT, REGS);
             MULTI := ((REGS.AX AND $00FF) = $004D);
          end
       else
          MULTI := false;
       end;
                          {make normal btrieve call}
      with XDATA do
         begin
         USER_BUF_ADDR.SEGMENT := SEG (DATA);
         USER_BUF_ADDR.OFFSET := OFS (DATA); {set data buffer address}
         USER_BUF_LEN := DATALEN;
         USER_FCB_ADDR.SEGMENT := SEG (POS);
         USER_FCB_ADDR.OFFSET := OFS (POS);          {set FCB address}
         USER_CUR_ADDR.SEGMENT := USER_FCB_ADDR.SEGMENT; {set cur seg}
         USER_CUR_ADDR.OFFSET := USER_FCB_ADDR.OFFSET+38;{set cur ofs}
         USER_FUNCTION := OP;          {set Btrieve operation code}
         USER_KEY_ADDR.SEGMENT := SEG (KBUF);
         USER_KEY_ADDR.OFFSET := OFS (KBUF);  {set key buffer address}
         USER_KEY_LENGTH := 255;          {assume its large enough}
         USER_KEY_NUMBER := KEY;                {set key number}
         USER_STAT_ADDR.SEGMENT := SEG (STAT);
         USER_STAT_ADDR.OFFSET := OFS (STAT);       {set status address}
         XFACE_ID := VAR_ID;                    {set lamguage id}
         end;

       REGS.DX := OFS (XDATA);
       REGS.DS := SEG (XDATA);

       if (NOT MULTI) then              {MultiUser version not installed}
          INTR (BTR_INT, REGS)
       else
          begin
          DONE := FALSE;
          repeat
            REGS.BX := ProcId;
            REGS.AX := 1;
            if (REGS.BX <> 0) then REGS.AX := 2;
            REGS.AX := REGS.AX + (MULTI_FUNCTION * 256);
            INTR (BTR2_INT, REGS);
```

```
            if ((REGS.AX AND $00FF) = 0) then DONE := TRUE
            else begin
                 REGS.AX := $0200;
                 INTR ($7F, REGS);
                 DONE := FALSE;
                 end;
          until (DONE);
          if (ProcId = 0) then ProcId := REGS.BX;
          end;
       DATALEN := XDATA.USER_BUF_LEN;
    end;
    BTRV := STAT;
END;    { BTRV-Function definition }



{           Mensajes de error para io

    1:  msg[1]:='OPERACION INVALIDA';
    2:  msg[1]:='ERROR DE E/S';
    3:  msg[1]:='EL ARCHIVO NO ESTA ABIERTO';
    4:  msg[1]:='NO EXISTE LA LLAVE ESPECIFICADA';
    5:  msg[1]:='LA LLAVE NO ADMITE REGISTROS DUPLICADOS';
    6:  msg[1]:='NUMERO DE LLAVE INVALIDO';
    7:  msg[1]:='NUMERO DE LLAVE DIFERENTE';
    8:  msg[1]:='POSICIONAMIENTO INVALIDO';
    9:  msg[1]:='FIN   DE   ARCHIVO';
   10:  msg[1]:='LA LLAVE NO ES MODIFICABLE';
   12:  msg[1]:='EL ARCHIVO NO EXISTE O NO ES ACCESIBLE A ESTE USUARIO';
   14:  msg[1]:='EL ARCHIVO DE "PRE-IMAGE" NO PUDO SER ABIERTO O
              CREADO';
   15:  msg[1]:='ERROR DURANTE UNA FUNCION DE "PRE-IMAGE"';
   17:  msg[1]:='ERROR   EN   EL   CLOSE';
   18:  msg[1]:='EL DISCO ESTA LLENO !!!!';
   20:  msg[1]:='NO HA SIDO ACTIVADO EL MANEJADOR DE ARCHIVOS
              (BTRIEVE)';
   21:  msg[1]:='ERROR EN EL PARAMETRO "KEY-BUFFER"';
   22:  msg[1]:='ERROR EN EL PARAMETRO "RECORD-BUFFER"';
   24:  msg[1]:='TAMA#O DE PAGINA O "DATA-BUFFER" INVALIDOS';
   35:  msg[1]:='ERROR AL CAMBIAR DE DIRECTORIO';
   36:  msg[1]:='LA OPCION /T PARA TRANSACCIONES NO SE INDICO AL
              INICIO';
   37:  msg[1]:='"ABRE_TRANSACCION" INVALIDA. NO SE PUEDEN ANIDAR
              TRANSACCIONES';
   38:  msg[1]:='ERROR AL ESCRIBIR SOBRE AL ARCHIVO DE CONT.DE
              TRANSACCIONES';
   39:  begin
        msg[1]:='"ABORTA O CIERRA_TRANSACCION" INVALIDA .';
        msg[2]:='  NO SE TIENE ABIERTA NINGUNA TRANSACCION';
        end;
   40:  msg[1]:='NO SE PUEDEN AFECTAR MAS DE 12 ARCHIVOS DENTRO DE LA
              TRANSACCION';
   41:  msg[1]:='NO SE PUEDEN ABRIR/CERRAR ARCHIVOS DENTRO DE UNA
              TRANSACCION';
   42:  msg[1]:='EL ARCHIVO HABIA SIDO PREVIAM. ABIERTO PARA ACCESO
              ACELERADO';
   47:  begin
        msg[1]:='SE HA EXCEDIDO EL NUMERO DE ARCHIVOS ABIERTOS PARA
              ACCESO ACELERADO';
        msg[2]:=' INTENTE AJUSTAR LAS OPCIONES /M (AUMENTAR) Y /P
              (REDUCIR).';
        end;
   82:  begin
        msg[1]:='ERROR DE POSICIONAMIENTO. SE HA QUERIDO ACCESAR UN
              REGISTRO QUE';
        msg[2]:=' FUE BORRADO O CUYA LLAVE FUE MODIFICADA POR OTRO
              USUARIO.';
        end;
   83:  begin
        msg[1]:='EL REGISTRO QUE SE QUIERE BORRAR/ACTUALIZAR NO FUE
              ACCESADO';
        msg[2]:=' DENTRO DE LA TRANSACCION.';
        end;
```

```
   86:  msg[1]:='LA TABLA DE ARCHIVOS ESTA LLENA. AUMENTE EL TAMAÑO EN
                /F';

   ELSE  str(status:4,numstat);
         msg[1]:=' CONSULTE EL MANUAL >>> Status = '+numstat;
}


FUNCTION ERROR_IO ( codigo_error : integer ):integer;
   { Transforma el codigo_error de turbo en IORESULT a codigos BTRIEVE.}
BEGIN
    case codigo_error of
         2 : ERROR_IO := 12;
       103 : ERROR_IO := 3;
       100,                    {read  despues del fin de archivo}
       153 : ERROR_IO := 9;
       242 : ERROR_IO := 18;
         4 : ERROR_IO := 86;

    else
       ERROR_IO := codigo_error;
   end;   {endcase}
END;   { ERROR_IO }



PROCEDURE INICIO_TRANSACCION (var status_io : integer);
        { Inicia transaccion en archivos btrieve. }
   CONST
      abre_trans         = 19;
BEGIN
    status_io := BTRV ( abre_trans,     dummy,         dummy,
                        dummy,          dummy,         dummykey);
END;   { INICIO_TRANSACCION }


PROCEDURE FIN_TRANSACCION (var status_io : integer);
        { Termina transaccion en archivos btrieve. }
   CONST
      cierra_trans       = 20;
BEGIN
    status_io := BTRV ( cierra_trans,   dummy,         dummy,
                        dummy,          dummy,         dummykey);
END;   { FIN_TRANSACCION }


PROCEDURE ABORTA_TRANSACCION (var status_io : integer);
        { Aborta transaccion en archivos btrieve. }
   CONST
      aborta_trans = 21;
BEGIN
    status_io := BTRV ( aborta_trans,   dummy,         dummy,
                        dummy,          dummy,         dummykey);
END;   { ABORTA_TRANSACCION }


PROCEDURE RESETEA_BTRIEVE;

   CONST
      breset = 28;
VAR
   stat_io : integer;

BEGIN
   stat_io := BTRV ( breset,    dummy,    dummy,
                     dummy,      dummy,    0);
END;   {RESETEA_BTRIEVE}

PROCEDURE REGISTROS_ARCHIVO (var registros : extended;
                             bloque_pos : tpo_bloque_pos;
```

```
                                    var status_io : integer);
    VAR
       llave           : string[64];
       bufer           : integer;
       wr_estadistica  : espec_archivo;
    BEGIN
    bufer := sizeof(wr_estadistica);
    status_io := BTRV (estado_archivo, bloque_pos, wr_estadistica,
                       bufer, llave, llavel);
    if (status_io = 0) then
       with wr_estadistica do
          registros := no_registros[1]     + no_registros[2]*256+
                       no_registros[3]*65536+ no_registros[4]*16777216;
    END;

  PROCEDURE CREA_ARCHIVO (    path_archivo : string;
                          var bloque_pos : tpo_bloque_pos;
                          var status_io : integer);
    VAR
       llave           : string[64];
       bufer           : integer;
       wr_estadistica  : espec_archivo;
       nom_archivo     : array[1..50] of char;
       i : integer;
    BEGIN
    bufer:=sizeof(WR_Estadistica);
    status_io := BTRV (estado_archivo, bloque_pos,
                          WR_Estadistica.especifica_bufer,
                       bufer, llave,llavel);
    if Status_Io = 0 then
       begin
       status_io:=BTRV (cierra_archivo, bloque_pos, dummy,
                        dummy, dummy, dummykey );
       if status_io = 0 then
          begin
          for i:=1 to 50 do nom_archivo[i] := ' ';
          move (path_archivo[1], nom_archivo[1], length (path_archivo));
          bufer:=sizeof(WR_Estadistica);
          status_io:= BTRV(crea, bloque_pos,
                           WR_Estadistica.especifica_bufer, bufer,
                           nom_archivo, 0);
          end
       end
    END;

END.
```

Programa para manejo del archivo de control del sistema:

```
{ UNIDAD PARA EL MANEJO DEL ARCHIVO CONTROL}
{$I-,V-,R-,N+,E+,D+}

UNIT UCONTROL;

{ Unidad que contine las llamadas de btrieve el archivo
  de CONTROL. }
{ Archivo   : CONTROL
  Objetivo  : Llevar informacion de control para permitir la
              recuperacion de los distintos procesos en la micro.}

INTERFACE

   USES
      UDCLGIRO;


   CONST
      no_realizado_ctr = 0;
      realizado_ctr    = 1;
      con_error_ctr    = 2;
      bloqueado_ctr    = 3;


   TYPE
      LLAVE_CONTROL = longint;

      R_CONTROL     =  RECORD
        proceso_id_ctr      : longint;
        status_ctr          : integer;
        descripcion_ctr     : tpo_descripcion_proceso;
        fecha_ctr           : tpo_fecha;
        hora_ctr            : tpo_hora;
        END; { R_CONTROL }

   FUNCTION ABRE_CONTROL       (     modo          : char ) : integer;

   FUNCTION CIERRA_CONTROL :integer;

   PROCEDURE ACTUALIZA_CONTROL (     wr_control : r_control;
                                 var status_io    : integer );

   PROCEDURE LEE_PRIMER_CONTROL ( var wr_control : r_control ;
                                    var status_io    : integer );

   PROCEDURE LEE_SIG_CONTROL ( var  wr_control : r_control ;
                                 var status_io    : integer );

   PROCEDURE LEE_CONTROL       ( var  wr_control : r_control ;
                                        llave_ctr   : llave_control;
                                   var status_io    : integer );

   PROCEDURE FECHA_HOY (var FechaHoy   : Tpo_Fecha;
                        var Estado_Io  : integer);

   PROCEDURE FECHA_SIGUIENTE (var FechaSig   : Tpo_Fecha;
                              var Estado_Io  : integer);


   PROCEDURE LEE_STATUS_PROCESO (    proceso        : longint;
                                  var estado_proceso : integer;
                                  var Estado_io      : integer);

   PROCEDURE ACTUALIZA_STATUS_PROCESO (    proceso        : longint;
                                        estado_proceso : integer;
                                       var Estado_io      : integer);


IMPLEMENTATION

   USES
```

```
   UGRALGIR,
   UPROGIR,
   UMSJGIRO,
   URUTGIRO,
   UBTRIEVE;

VAR
   bloque_pos_ctr      : array [1..128] of byte;
   long_reg_ctr        : integer;  { Tamanio registro del archivo }

FUNCTION ABRE_CONTROL       (    modo          : char ) : integer;
      { Abre archivo de control }
   VAR
      wr_control  : r_control;
      status_io   : integer;
      nom_arch    : string [50];
      nom_archivo : array [1..50] of char;

   BEGIN
   nom_arch := 'f:\giros\datos\control.dat';
   move (blancos[1], nom_archivo[1], 50);
   move (nom_arch[1], nom_archivo[1], length (nom_arch));
   long_reg_ctr := sizeof (wr_control) ;
   if (modo = modo_lectura) then
      status_io := BTRV (abre_archivo, bloque_pos_ctr, wr_control,
                         long_reg_ctr, nom_archivo, solo_lectura )
   else
   if (modo = modo_recuperacion) then
      status_io := BTRV (abre_archivo, bloque_pos_ctr, wr_control,
                         long_reg_ctr, nom_archivo, con_recuperacion );

   if (status_io = 20) then   (No esta btrieve)
      write('**********     Falta cargar Btrieve     ********** ');
   abre_control := status_io;
   END;   { ABRE_CONTROL }


FUNCTION CIERRA_CONTROL :integer;
   BEGIN
   (*WRITELN(LST,'  CIERRA CONTROL$$');*)
   cierra_control := BTRV (cierra_archivo, bloque_pos_ctr, dummy,
                           dummy, dummy, dummykey );
   END;


PROCEDURE ACTUALIZA_CONTROL (     wr_control : r_control;
                              var  status_io    : integer );
      { Reescribe en archivo control el registro wr_control }
   VAR
     llave_ctr  : llave_control ;
   BEGIN
   status_io := BTRV  ( actualiza,    bloque_pos_ctr, wr_control,
                        long_reg_ctr, llave_ctr,      llave1    );
   END;   { ACTUALIZA_CONTROL }


PROCEDURE LEE_PRIMER_CONTROL ( var  wr_control : r_control ;
                               var  status_io    : integer );
      { Pone en wr_control registro el registro de control}
   VAR
   llave_ctr : llave_control;
   BEGIN
   status_io := BTRV (get_lowest,     bloque_pos_ctr,  wr_control,
                      long_reg_ctr,  llave_ctr,       llave1 );
   END;


PROCEDURE LEE_SIG_CONTROL ( var  wr_control : r_control ;
                            var  status_io    : integer );
```

```
        { Pone en wr_control registro el registro de control}
    VAR
    llave_ctr : llave_control;
    BEGIN
    llave_ctr := wr_control.proceso_id_ctr;
    status_io := BTRV (get_next,     bloque_pos_ctr,  wr_control,
                       long_reg_ctr, llave_ctr,       llave1 );
    END;


PROCEDURE LEE_CONTROL          ( var  wr_control : r_control ;
                                      llave_ctr   : llave_control;
                                 var  status_io   : integer );
        { Pone en wr_control registro el registro de control}
    BEGIN
    status_io := BTRV (get_equal,    bloque_pos_ctr,  wr_control,
                       long_reg_ctr, llave_ctr,       llave1 );
    END;


PROCEDURE FECHA_HOY (var FechaHoy   : Tpo_Fecha;
                     var Estado_Io  : integer);

    VAR
       wr_control : r_control;
       estado_aux   : integer;

    BEGIN
    estado_io := ABRE_CONTROL(modo_lectura);
    if estado_io = ok_io then
       begin
       LEE_CONTROL (wr_control, fecha_control, Estado_Io);
       if estado_io = ok_io then
          begin
          FechaHoy  := wr_control.fecha_ctr;
          estado_io := CIERRA_CONTROL;
          end
       else
          estado_aux := CIERRA_CONTROL;
       end;
    END;

PROCEDURE FECHA_SIGUIENTE (var FechaSig   : Tpo_Fecha;
                           var Estado_Io  : integer);

    VAR
       wr_control : r_control;
       estado_aux   : integer;

    BEGIN
    estado_io := ABRE_CONTROL(modo_lectura);
    if estado_io = ok_io then
       begin
       LEE_CONTROL (wr_control, fecha_sig_control, Estado_Io);
       if estado_io = ok_io then
          begin
          FechaSig  := wr_control.fecha_ctr;
          estado_io := CIERRA_CONTROL;
          end
       else
          estado_aux := CIERRA_CONTROL;
       end;
    END;


PROCEDURE LEE_STATUS_PROCESO (      proceso        : longint;
                              var estado_proceso : integer;
                              var Estado_io      : integer);

    VAR
    wr_control : r_control;
    estado_aux   : integer;
```

```
   BEGIN
   estado_io := ABRE_CONTROL(modo_lectura);
   if estado_io = ok_io then
      begin
      LEE_CONTROL(wr_control, proceso, estado_io);
      if estado_io = ok_io then
         begin
         estado_proceso := wr_control.status_ctr;
         estado_io := CIERRA_CONTROL;
         end
      else
        estado_aux := CIERRA_CONTROL;
      end;
   END;


PROCEDURE ACTUALIZA_STATUS_PROCESO (    proceso        : longint;
                                    estado_proceso : integer;
                                   var Estado_io      : integer);

   VAR
   wr_control : r_control;
   estado_aux   : integer;

   BEGIN
   estado_io := ABRE_CONTROL(modo_recuperacion);
   if estado_io = ok_io then
      begin
      LEE_CONTROL(wr_control, proceso, estado_io);
      if estado_io = ok_io then
         begin
         wr_control.status_ctr := estado_proceso;
         DA_FECHA_Y_HORA_MSDOS(wr_control.fecha_ctr,
                               wr_control.hora_ctr);
         ACTUALIZA_CONTROL(wr_control, estado_io);
         if estado_io = ok_io then
            estado_io := CIERRA_CONTROL
         else
            estado_aux := CIERRA_CONTROL;
         end
      else
         estado_aux := CIERRA_CONTROL;
      end;
   END;

BEGIN
long_reg_ctr := sizeof(r_control);

END.
```

Programa para manejo del archivo de corresponsales:

```
UNIT UCORRESP;

{$I-,V-,R-,N+,E+,D+}

INTERFACE

   USES
        UDCLGIRO;

   CONST

     { Nombre del archivo en MS-DOS: }
      archivo_corresponsales= 'f:\GIROS\DATOS\CORRESP.DAT';

   TYPE
      LLAVE1_CORR = tpo_corresponsal;


      R_CORRESP = RECORD
         clave_corr      : tpo_corresponsal;
         nombre_corr     : array [1..45] of char;
         direccion_corr  : array[1..45] of char;
         cuenta_corr     : tpo_cuenta;
      END; { R_CORRESP}


   FUNCTION ABRE_CORRESP( modo : char ) : integer;
   FUNCTION CIERRA_CORRESP: integer;
   PROCEDURE ACTUALIZA_CORRESP(     wr_corresp: r_corresp;
                                var status_io : integer);
   PROCEDURE ESCRIBE_CORRESP(     wr_corresp: r_corresp;
                               var status_io : integer          );
   PROCEDURE LEE_PRIMER_CORRESP(var wr_corresp: r_corresp;
                                    var  llave_corr: llave1_corr;
                                var status_io : integer);
   PROCEDURE LEE_SIGUIENTE_CORRESP(var wr_corresp: r_corresp;
                                       var  llave_corr: llave1_corr;
                                    var status_io : integer);
   PROCEDURE LEE_CORRESP(var wr_corresp: r_corresp;
                              llave_corr: llave1_corr;
                           var status_io : integer                );
   PROCEDURE LEE_ULT_CORRESP(var wr_corresp: r_corresp;
                             var status_io : integer          );
   PROCEDURE BORRA_CORRESP(var wr_corresp: r_corresp;
                                llave_corr: llave1_corr;
                            var status_io : integer);

IMPLEMENTATION

   USES
      UBTRIEVE;

   VAR

      long_reg_corr: integer;
      bloque_pos_corr: array [1..128] of byte;


FUNCTION ABRE_CORRESP( modo : char ): integer;
   { Abre el archivo de corresponsales. }
   VAR
      dueno_archivo  : string[10];
      nombre_archivo : string[80];
      modo_int       : integer;
   BEGIN
   dueno_archivo := #0;
```

```
    nombre_archivo := archivo_corresponsales + #0;
    case modo of
       modo_lectura      : modo_int := solo_lectura;
       modo_recuperacion : modo_int := con_recuperacion;
       end; {case}
    ABRE_CORRESP:= BTRV (abre_archivo, bloque_pos_corr,dueno_archivo[1],
                         long_reg_corr,nombre_archivo[1], modo_int      );
    END; { ABRE_CORRESP}


FUNCTION CIERRA_CORRESP: integer;
    BEGIN
    CIERRA_CORRESP:= BTRV (cierra_archivo, bloque_pos_corr, dummy,
                                  dummy, dummy, dummykey );
    END;


PROCEDURE ACTUALIZA_CORRESP(   wr_corresp: r_corresp;
                           var status_io  : integer);
    { Update con llave 1 }
    VAR
       llave1_control : llave1_corr;
    BEGIN
    status_io := BTRV (actualiza, bloque_pos_corr, wr_corresp,
                       long_reg_corr, llave1_control, llave1 );
    END;


PROCEDURE ESCRIBE_CORRESP(   wr_corresp: r_corresp;
                         var status_io  : integer);
    VAR
       llave1_control : llave1_corr;
    BEGIN
    status_io := BTRV (escribe, bloque_pos_corr, wr_corresp,
                       long_reg_corr, llave1_control, llave1);
    END;


PROCEDURE LEE_PRIMER_CORRESP(var wr_corresp: r_corresp;
                                 var llave_corr: llave1_corr;
                               var status_io : integer);
    { Pone en wr_corresp el primer registro de corresponsales con
      llave 1 indicada. }
    BEGIN
    status_io := BTRV (get_lowest, bloque_pos_corr, wr_corresp,
                       long_reg_corr, llave_corr, llave1        );
    END;

PROCEDURE LEE_SIGUIENTE_CORRESP(var wr_corresp: r_corresp;
                                    var llave_corr: llave1_corr;
                                  var status_io : integer);
    { Pone en wr_corresp el siguiente registro de corresponsales con
      llave 1 indicada. }
    BEGIN
    status_io := BTRV (get_next, bloque_pos_corr, wr_corresp,
                       long_reg_corr, llave_corr, llave1        );
    END;

PROCEDURE LEE_CORRESP(var wr_corresp: r_corresp;
                          llave_corr: llave1_corr;
                        var status_io : integer);
    { Pone en wr_corresp el registro de corresponsales con llave 1
       indicada.}
    BEGIN
    status_io := BTRV (get_equal, bloque_pos_corr, wr_corresp,
                       long_reg_corr, llave_corr, llave1        );
    END;


PROCEDURE LEE_ULT_CORRESP(var wr_corresp: r_corresp;
                            var status_io : integer);
    { Pone en wr_corresp el siguiente registro del archivo, usando la llave
1. }
    VAR
       llave1_control : llave1_corr;
```

```
    BEGIN
    status_io := BTRV (get_highest, bloque_pos_corr, wr_corresp,
                       long_reg_corr, llave1_control, llave1);
    END;


PROCEDURE BORRA_CORRESP(var wr_corresp: r_corresp;
                            llave_corr: llave1_corr;
                        var status_io : integer);
    { Borra el registro  wr_corresp de corresponsales con  llave 1 indicada.
}
    BEGIN
    status_io := BTRV (borra, bloque_pos_corr, wr_corresp,
                       long_reg_corr, llave_corr, llave1       );
    END;


BEGIN
   long_reg_corr:= sizeof(r_corresp);
END. { UNIT UCORRESP}
```

Programa para manejo del archivo de cuentahabientes:

```
UNIT UCTAHAB;

{$I-,V-,R-,N+,E+,D+}

INTERFACE

   USES
        UDCLGIRO;

   CONST

    { Nombre del archivo en MS-DOS: }
      archivo_cuentahablentes= 'F:\GIROS\DATOS\CTAHABTE.DAT';

   TYPE
      LLAVE1_CTAHABTE = tpo_corresponsal;


      R_CTAHABTE = RECORD
         clave_ctahabte         : tpo_corresponsal;
         nombre_ctahabte        : tpo_beneficiario;
         cta_tesofe_ctahabte    : tpo_cuenta;
         cuenta_ctahabte        : tpo_cuenta;
         divisa_monto_ctahabte  : tpo_divisa;
      END; { R_CTAHABTE}


   FUNCTION ABRE_CTAHABTE( modo : char ) : integer;
   FUNCTION CIERRA_CTAHABTE: integer;
   PROCEDURE ACTUALIZA_CTAHABTE(    wr_ctahabte: r_ctahabte;
                                    var status_io  : integer);
   PROCEDURE ESCRIBE_CTAHABTE(    wr_ctahabte : r_ctahabte;
                                  var status_io   : integer          );
   PROCEDURE LEE_PRIMER_CTAHABTE(var wr_ctahabte: r_ctahabte;
                                  var llave_ctahabte: llavel_ctahabte;
                                  var status_io : integer);
   PROCEDURE LEE_SIGUIENTE_CTAHABTE(var wr_ctahabte: r_ctahabte;
                                    var llave_ctahabte: llavel_ctahabte;
                                    var status_io : integer);
   PROCEDURE LEE_CTAHABTE(var wr_ctahabte: r_ctahabte;
                                llave_ctahabte: llavel_ctahabte;
                           var status_io : integer                 );
   PROCEDURE LEE_ULT_CTAHABTE(var wr_ctahabte: r_ctahabte;
                              var status_io : integer          );
   PROCEDURE BORRA_CTAHABTE(var wr_ctahabte: r_ctahabte;
                                llave_ctahabte: llavel_ctahabte;
                            var status_io : integer);

IMPLEMENTATION


   USES
      UBTRIEVE;

   VAR

      long_reg_ctahabte: integer;
      bloque_pos_ctahabte: array [1..128] of byte;


FUNCTION ABRE_CTAHABTE( modo : char ): integer;
   { Abre el archivo de corresponsales. }
   VAR
      dueno_archivo  : string[10];
      nombre_archivo : string[80];
      modo_int       : integer;
   BEGIN
   dueno_archivo := #0;
```

```
   nombre_archivo := archivo_cuentahabientes + #0;
   case modo of
      modo_lectura      : modo_int := solo_lectura;
      modo_recuperacion : modo_int := con_recuperacion;
      end; {case}
   ABRE_CTAHABTE:= BTRV (abre_archivo, bloque_pos_ctahabte,
                         dueno_archivo[1],
                   long_reg_ctahabte,nombre_archivo[1], modo_int      );
   END; { ABRE_CTAHABTE}


FUNCTION CIERRA_CTAHABTE: integer;
   BEGIN
   CIERRA_CTAHABTE:= BTRV (cierra_archivo, bloque_pos_ctahabte, dummy,
                           dummy, dummy, dummykey );
   END;


PROCEDURE ACTUALIZA_CTAHABTE(    wr_ctahabte: r_ctahabte;
                         var status_io  : integer);
   { Update con llave 1 }
   VAR
      llave1_control : llave1_ctahabte;
   BEGIN
   status_io := BTRV (actualiza, bloque_pos_ctahabte, wr_ctahabte,
                   long_reg_ctahabte, llave1_control, llave1 );
   END;


PROCEDURE ESCRIBE_CTAHABTE(    wr_ctahabte: r_ctahabte;
                          var status_io  : integer);
   VAR
      llave1_control : llave1_ctahabte;
   BEGIN
   status_io := BTRV (escribe, bloque_pos_ctahabte, wr_ctahabte,
                   long_reg_ctahabte, llave1_control, llave1);
   END;


PROCEDURE LEE_PRIMER_CTAHABTE(var wr_ctahabte: r_ctahabte;
                           var llave_ctahabte: llave1_ctahabte;
                           var status_io : integer);
   { Pone en wr_ctahabte el primer registro de corresponsales con
     llave 1 indicada. }
   BEGIN
   status_io := BTRV (get_lowest, bloque_pos_ctahabte, wr_ctahabte,
                   long_reg_ctahabte, llave_ctahabte, llave1         );
   END;

PROCEDURE LEE_SIGUIENTE_CTAHABTE(var wr_ctahabte: r_ctahabte;
                              var llave_ctahabte: llave1_ctahabte;
                              var status_io : integer);
   { Pone en wr_ctahabte el siguiente registro de corresponsales con
    llave 1 indicada. }
   BEGIN
   status_io := BTRV (get_next, bloque_pos_ctahabte, wr_ctahabte,
                   long_reg_ctahabte, llave_ctahabte, llave1         );
   END;

PROCEDURE LEE_CTAHABTE(var wr_ctahabte: r_ctahabte;
                          llave_ctahabte: llave1_ctahabte;
                       var status_io : integer);
   { Pone en wr_ctahabte el registro de corresponsales con llave 1
      indicada.}
   BEGIN
   status_io := BTRV (get_equal, bloque_pos_ctahabte, wr_ctahabte,
                   long_reg_ctahabte, llave_ctahabte, llave1         );
   END;


PROCEDURE LEE_ULT_CTAHABTE(var wr_ctahabte: r_ctahabte;
                        var status_io : integer);
   { Pone en wr_ctahabte el siguiente registro del archivo, usando la llave
1. }
   VAR
```

```
    llave1_control : llave1_ctahabte;
  BEGIN
  status_io := BTRV (get_highest, bloque_pos_ctahabte, wr_ctahabte,
                     long_reg_ctahabte, llave1_control, llave1);
  END;

PROCEDURE BORRA_CTAHABTE(var wr_ctahabte: r_ctahabte;
                         llave_ctahabte: llave1_ctahabte;
                     var status_io : integer);
  { Borra el registro wr_ctahabte de corresponsales con  llave 1
    indicada. }
  BEGIN
  status_io := BTRV (borra, bloque_pos_ctahabte, wr_ctahabte,
                     long_reg_ctahabte, llave_ctahabte, llave1        );
  END;

BEGIN
  long_reg_ctahabte:= sizeof(r_ctahabte);
END. { UNIT UCTAHABTE}
```

Programa para manejo del archivo de divisas:

```
UNIT UDIVISA;

{$I-,V-,R-,N+,E+,D+}

INTERFACE

  USES
      UDCLGIRO;

  CONST

    { Nombre del archivo en MS-DOS: }
     archivo_divisas= 'F:\GIROS\DATOS\DIVISAS.DAT';

  TYPE
     LLAVE1_DIV= tpo_divisa;


     R_DIVISA = RECORD
        clave_div      : tpo_divisa;
        nombre_esp_div : tpo_beneficiario;
        nombre_ing_div : tpo_beneficiario;
(*        con_centavos_div : char; *)
     END; { R_DIVISA}


  FUNCTION ABRE_DIVISA( modo : char ) : integer;
  FUNCTION CIERRA_DIVISA: integer;
  PROCEDURE ACTUALIZA_DIVISA(    wr_divisa: r_divisa;
                             var status_io : integer);
  PROCEDURE ESCRIBE_DIVISA(    wr_divisa: r_divisa;
                           var status_io : integer          );
  PROCEDURE LEE_PRIMER_DIVISA(var wr_divisa: r_divisa;
                              var  llave_div : llavel_div;
                              var status_io : integer);
  PROCEDURE LEE_SIGUIENTE_DIVISA(var wr_divisa: r_divisa;
                                 var  llave_div : llavel_div;
                                  var status_io : integer);
  PROCEDURE LEE_DIVISA(var wr_divisa: r_divisa;
                           llave_div: llavel_div;
                       var status_io : integer                );
  PROCEDURE LEE_ULT_DIVISA(var wr_divisa: r_divisa;
                       var status_io : integer          );
  PROCEDURE BORRA_DIVISA(var wr_divisa: r_divisa;
                             llave_div : llavel_div;
                         var status_io : integer);

IMPLEMENTATION

  USES
    UBTRIEVE;

  VAR

     long_reg_div: integer;
     bloque_pos_div: array [1..128] of byte;


FUNCTION ABRE_DIVISA( modo : char ): integer;
  { Abre el archivo de divisas. }
  VAR
     dueno_archivo  : string[10];
     nombre_archivo : string[80];
     modo_int       : integer;
  BEGIN
  dueno_archivo := #0;
  nombre_archivo := archivo_divisas + #0;
```

```
   case modo of
      modo_lectura       : modo_int := solo_lectura;
      modo_recuperacion  : modo_int := con_recuperacion;
      end; {case}
   ABRE_DIVISA:= BTRV (abre_archivo, bloque_pos_div, dueno_archivo[1],
                       long_reg_div,nombre_archivo[1], modo_int       );
   END; { ABRE_DIVISA}


FUNCTION CIERRA_DIVISA: integer;
   BEGIN
   CIERRA_DIVISA:= BTRV (cierra_archivo, bloque_pos_div, dummy,
                                 dummy, dummy, dummykey );
   END;


PROCEDURE ACTUALIZA_DIVISA(    wr_divisa: r_divisa;
                           var status_io  : integer);
   { Update con llave 1 }
   VAR
      llave1_control : llave1_div;
   BEGIN
   status_io := BTRV (actualiza, bloque_pos_div, wr_divisa,
                      long_reg_div, llave1_control, llave1 );
   END;


PROCEDURE ESCRIBE_DIVISA(    wr_divisa: r_divisa;
                         var status_io  : integer);
   VAR
      llave1_control : llave1_div;
   BEGIN
   status_io := BTRV (escribe, bloque_pos_div, wr_divisa,
                      long_reg_div, llave1_control, llave1);
   END;


PROCEDURE LEE_PRIMER_DIVISA(var wr_divisa: r_divisa;
                            var  llave_div : llave1_div;
                            var status_io : integer);
   { Pone en wr_divisa el primer registro de divisas con llave 1 indicada.}
   BEGIN
   status_io := BTRV (get_lowest, bloque_pos_div, wr_divisa,
                      long_reg_div, llave_div, llave1         );
   END;

PROCEDURE LEE_SIGUIENTE_DIVISA(var wr_divisa: r_divisa;
                               var llave_div : llave1_div;
                               var status_io : integer);
   { Pone en wr_divisa el siguiente registro de divisas con
     llave 1 indicada. }
   BEGIN
   status_io := BTRV (get_next, bloque_pos_div, wr_divisa,
                      long_reg_div, llave_div, llave1         );
   END;

PROCEDURE LEE_DIVISA(var wr_divisa : r_divisa;
                         llave_div : llave1_div;
                       var status_io : integer);
   { Pone en wr_divisa el registro de divisas con llave 1  indicada.}
   BEGIN
   status_io := BTRV (get_equal, bloque_pos_div, wr_divisa,
                      long_reg_div, llave_div, llave1         );
   END;


PROCEDURE LEE_ULT_DIVISA(var wr_divisa: r_divisa;
                         var status_io : integer);
   { Pone en wr_divisa el siguiente registro del archivo, usando la llave
1.}
   VAR
      llave1_control : llave1_div;
   BEGIN
   status_io := BTRV (get_highest, bloque_pos_div, wr_divisa,
                      long_reg_div, llave1_control, llave1);
```

```
  END;

PROCEDURE BORRA_DIVISA(var wr_divisa: r_divisa;
                       llave_div : llavel_div;
                   var status_io : integer);
  { Borra el registro  wr_divisa de divisas con  llave 1 indicada. }
  BEGIN
  status_io := BTRV (borra, bloque_pos_div, wr_divisa,
                     long_reg_div, llave_div, llavel        );
  END;


BEGIN
  long_reg_div := sizeof(r_divisa);
END. { UNIT UDIVISA}
```

Programa para manejo del archivo de cancelaciones:

```
UNIT UERRORES;

{$I-,V-,R-,N+,E+,D+}

INTERFACE

   USES
        UDCLGIRO;

   CONST

    { Nombre del archivo en MS-DOS: }
      archivo_errores= 'F:\GIROS\DATOS\ERRORES.DAT';

   TYPE
      LLAVE1_ERRORES = RECORD
         corresponsal_kerr : tpo_corresponsal;
         divisa_kerr   : tpo_divisa;
         folio_kerr    : tpo_folio;
      END; {LLAVE1_ERRORES}

      R_ERRORES  = RECORD
         folio_err         : tpo_folio;              {08}
         corresponsal_err  : tpo_corresponsal;       {10}
         divisa_err        : tpo_divisa;             {03}
         num_giro_err      : tpo_folio;              {08}
         causa_err         : tpo_beneficiario;       {100}
         fecha_err         : tpo_fecha;              {08}
      END; { R_ERRORES}                             {137}

   FUNCTION ABRE_ERRORES( modo : char ) : integer;
   FUNCTION CIERRA_ERRORES : integer;
   PROCEDURE ACTUALIZA1_ERRORES(    wr_errores  : r_errores;
                                var status_io : integer);
   PROCEDURE ESCRIBE_ERRORES(    wr_errores  : r_errores;
                             var status_io : integer          );
   PROCEDURE LEE_PRIMER_ERRORES(var wr_errores  : r_errores;
                                var llave_errores : llave1_errores;
                             var status_io : integer);
   PROCEDURE LEE_SIG_ERRORES(var wr_errores  : r_errores;
                             var llave_errores : llave1_errores;
                          var status_io : integer);
   PROCEDURE LEE1_ERRORES(var wr_errores  : r_errores;
                              llave_errores : llave1_errores;
                       var status_io : integer          );
   PROCEDURE LEE1_GE_ERRORES (var wr_errores : r_errores;
                           var  llave_errores: llave1_errores;
                           var status_io : integer );

   PROCEDURE LEE1_ULT_GIR(var wr_errores  : r_errores;
                          var status_io : integer          );
   PROCEDURE BORRA_ERRORES(var wr_errores  : r_errores;
                           llave_errores : llave1_errores;
                        var status_io : integer);

IMPLEMENTATION

   USES
      UBTRIEVE;

   VAR

      long_reg_err   : integer;
      bloque_pos_err : array [1..128] of byte;
```

```
FUNCTION ABRE_ERRORES( modo : char ): integer;
   { Abre el archivo de giros. }
   VAR
      dueno_archivo  : string[10];
      nombre_archivo : string[80];
      modo_int       : integer;
   BEGIN
   dueno_archivo := #0;
   nombre_archivo := archivo_errores + #0;
   case modo of
      modo_lectura       : modo_int := solo_lectura;
      modo_recuperacion : modo_int := con_recuperacion;
      end; {case}
   ABRE_ERRORES:= BTRV (abre_archivo, bloque_pos_err,dueno_archivo[1],
                        long_reg_err,nombre_archivo[1], modo_int      );
   END; { ABRE_ERRORES}


FUNCTION CIERRA_ERRORES: integer;
   BEGIN
   CIERRA_ERRORES:= BTRV (cierra_archivo, bloque_pos_err, dummy,
                                dummy, dummy, dummykey );
   END;


PROCEDURE ACTUALIZA1_ERRORES(    wr_errores   : r_errores;
                             var status_io  : integer);
   { Update con llave 1 }
   VAR
      llave1_control : llave1_errores;
   BEGIN
   status_io := BTRV (actualiza, bloque_pos_err, wr_errores,
                      long_reg_err, llave1_control, llave1 );
   END;


PROCEDURE ESCRIBE_ERRORES(    wr_errores   : r_errores;
                          var status_io  : integer);
   VAR
      llave1_control : llave1_errores;
   BEGIN
   status_io := BTRV (escribe, bloque_pos_err, wr_errores,
                      long_reg_err, llave1_control, llave1);
   END;


PROCEDURE LEE_PRIMER_ERRORES(var wr_errores  : r_errores;
                             var  llave_errores : llave1_errores;
                             var status_io : integer);
   { Pone en wr_errores el primer registro de giros con llave 1 indicada. }
   BEGIN
   status_io := BTRV (get_lowest, bloque_pos_err, wr_errores,
                      long_reg_err, llave_errores, llave1        );
   END;

PROCEDURE LEE_SIG_ERRORES(var wr_errores  : r_errores;
                          var  llave_errores : llave1_errores;
                          var status_io : integer);
   { Pone en wr_errores el siguiente registro de giros con llave 1
      indicada. }
   BEGIN
   status_io := BTRV (get_next, bloque_pos_err, wr_errores,
                      long_reg_err, llave_errores, llave1        );
   END;

PROCEDURE LEE1_ERRORES(var wr_errores  : r_errores;
                       llave_errores : llave1_errores;
                      var status_io : integer);
   { Pone en wr_errores el registro de giros con llave 1  indicada. }
   BEGIN
   status_io := BTRV (get_equal, bloque_pos_err, wr_errores,
                      long_reg_err, llave_errores, llave1        );
   END;
```

```
PROCEDURE LEE1_GE_ERRORES (var wr_errores : r_errores;
                           var llave_errores: llavel_errores;
                           var status_io : integer );
  BEGIN
  status_io := BTRV (get_greater_equal,bloque_pos_err, wr_errores,
                     long_reg_err, llave_errores,      llavel );
  END;

PROCEDURE LEE1_ULT_GIR(var wr_errores  : r_errores;
                       var status_io : integer);
  { Pone en wr_errores el siguiente registro del archivo, usando la llave
    1. }
  VAR
    llavel_control : llavel_errores;
  BEGIN
  status_io := BTRV (get_highest, bloque_pos_err, wr_errores,
                     long_reg_err, llavel_control, llavel);
  END;


PROCEDURE BORRA_ERRORES(var wr_errores  : r_errores;
                        llave_errores : llavel_errores;
                      var status_io : integer);
  { Borra el registro  wr_errores de giros con  llave 1 indicada. }
  BEGIN
  status_io := BTRV (borra, bloque_pos_err, wr_errores,
                     long_reg_err, llave_errores, llavel        );
  END;


BEGIN
  long_reg_err := sizeof(r_errores);
END. { UNIT UERRORES}
```

Programa para manejo del archivo de giros:

```
UNIT UGIROS;

{$I-,V-,R-,N+,E+,D+}

INTERFACE

  USES
      UDCLGIRO;

  CONST

    { Nombre del archivo en MS-DOS: }
     archivo_giros= 'F:\GIROS\DATOS\GIROS.DAT';

  TYPE
     LLAVE1_GIROS = record
                    folio_girk        : tpo_folio;
                    fecha_girk        : tpo_fecha;
                    corresponsal_girk : tpo_corresponsal;
                    divisa_girk       : tpo_divisa;
                    end;

     LLAVE2_GIROS = record
                    cuentahabte_girk  : tpo_corresponsal;
                    divisa_girk       : tpo_divisa;
                    folio_girk        : tpo_folio;
                    end;


     LLAVE3_GIROS = record
                    corresponsal_girk : tpo_corresponsal;
                    fecha_girk        : tpo_fecha;
                    fecha_cap_girk    : tpo_fecha;
                    divisa_girk       : tpo_divisa;
                    folio_girk        : tpo_folio;
                    end;

     R_GIROS = RECORD
        folio_gir          : tpo_folio;
        fecha_gir          : tpo_fecha;
        fecha_cap_gir      : tpo_fecha;
        divisa_gir         : tpo_divisa;
        corresponsal_gir   : tpo_corresponsal;
        monto_gir          : extended;
        beneficiario_gir   : tpo_beneficiario150;
        numero_cheque_gir  : tpo_folio;
        cuentahabte_gir    : tpo_corresponsal;
        cuenta_cargo_gir   : tpo_cuenta;
        estado_gir         : char;
     END; { R_GIROS}

  var
     bloque_pos_gir : tpo_bloque_pos;

  FUNCTION ABRE_GIROS( modo : char ) : integer;
  FUNCTION CIERRA_GIROS : integer;
  PROCEDURE ACTUALIZA1_GIROS(    wr_giros  : r_giros;
                             var status_io : integer);
  PROCEDURE ESCRIBE_GIROS(    wr_giros  : r_giros;
                          var status_io : integer          );
  PROCEDURE LEE_PRIMER_GIROS(var wr_giros  : r_giros;
                             var llave_gir : llave1_giros;
                             var status_io : integer);
  PROCEDURE LEE_SIG_GIROS(var wr_giros  : r_giros;
                          var llave_gir : llave1_giros;
                          var status_io : integer);
  PROCEDURE LEE1_GIROS(var wr_giros  : r_giros;
                           llave_gir : llave1_giros;
```

```
                        var status_io : integer                 );
    PROCEDURE LEE1_GE_GIROS (var wr_giros : r_giros;
                             var llave_gir: llave1_giros;
                             var status_io : integer );

    PROCEDURE LEE1_ULT_GIROS(var wr_giros  : r_giros;
                           var llave_gir : llave1_giros;
                           var status_io : integer);

    PROCEDURE BORRA_GIROS(var wr_giros  : r_giros;
                              llave_gir : llave1_giros;
                          var status_io : integer);
    PROCEDURE LEE2_GIROS(var wr_giros   : r_giros;
                             llave_gir  : llave2_giros;
                         var status_io  : integer);

    PROCEDURE LEE2_PRIMER_GIROS(var wr_giros  : r_giros;
                                var llave_gir : llave2_giros;
                                var status_io : integer);

    PROCEDURE LEE2_SIG_GIROS(var wr_giros  : r_giros;
                           var llave_gir : llave2_giros;
                           var status_io : integer);

    PROCEDURE LEE2_ULT_GIROS(var wr_giros  : r_giros;
                           var llave_gir : llave2_giros;
                           var status_io : integer          );

    PROCEDURE LEE3_PRIMER_GIROS(var wr_giros  : r_giros;
                                var llave_gir : llave3_giros;
                                var status_io : integer);

    PROCEDURE LEE3_SIG_GIROS(var wr_giros  : r_giros;
                             var llave_gir : llave3_giros;
                             var status_io : integer);

    PROCEDURE LEE3_GIROS(var wr_giros  : r_giros;
                             llave_gir : llave3_giros;
                         var status_io : integer                );

    PROCEDURE LEE3_GE_GIROS (var wr_giros : r_giros;
                             var llave_gir: llave3_giros;
                             var status_io : integer );

    PROCEDURE LEE3_ULT_GIROS(var wr_giros  : r_giros;
                           var llave_gir : llave3_giros;
                           var status_io : integer);


IMPLEMENTATION

    USES
       UBTRIEVE;

    VAR

       long_reg_gir   : integer;


FUNCTION ABRE_GIROS( modo : char ): integer;
   { Abre el archivo de giros. }
   VAR
      dueno_archivo  : string[10];
      nombre_archivo : string[80];
      modo_int       : integer;
   BEGIN
   dueno_archivo := #0;
   nombre_archivo := archivo_giros + #0;
   case modo of
      modo_lectura       : modo_int := solo_lectura;
      modo_recuperacion : modo_int := con_recuperacion;
      end; {case}
   ABRE_GIROS:= BTRV (abre_archivo, bloque_pos_gir,dueno_archivo[1],
                      long_reg_gir,nombre_archivo[1], modo_int      );
```

```
  END; { ABRE_GIROS}

FUNCTION CIERRA_GIROS: integer;
  BEGIN
  CIERRA_GIROS:= BTRV (cierra_archivo, bloque_pos_gir, dummy,
                       dummy, dummy, dummykey );
  END;

PROCEDURE ACTUALIZA1_GIROS(   wr_giros   : r_giros;
                          var status_io  : integer);
  { Update con llave 1 }
  VAR
     llave1_control : llave1_giros;
  BEGIN
  status_io := BTRV (actualiza, bloque_pos_gir, wr_giros,
                     long_reg_gir, llave1_control, llave1 );
  END;

PROCEDURE ESCRIBE_GIROS(   wr_giros   : r_giros;
                       var status_io  : integer);
  VAR
     llave1_control : llave1_giros;
  BEGIN
  status_io := BTRV (escribe, bloque_pos_gir, wr_giros,
                     long_reg_gir, llave1_control, llave1);
  END;

PROCEDURE LEE_PRIMER_GIROS(var wr_giros  : r_giros;
                           var llave_gir : llave1_giros;
                           var status_io : integer);
  { Pone en wr_giros el primer registro de giros con llave 1 indicada. }
  BEGIN
  status_io := BTRV (get_lowest, bloque_pos_gir, wr_giros,
                     long_reg_gir, llave_gir, llave1        );
  END;

PROCEDURE LEE_SIG_GIROS(var wr_giros  : r_giros;
                        var llave_gir : llave1_giros;
                        var status_io : integer);
  { Pone en wr_giros el siguiente registro de giros con llave 1 indicada.}
  BEGIN
  status_io := BTRV (get_next, bloque_pos_gir, wr_giros,
                     long_reg_gir, llave_gir, llave1        );
  END;

PROCEDURE LEE1_GIROS(var wr_giros  : r_giros;
                         llave_gir : llave1_giros;
                     var status_io : integer);
  { Pone en wr_giros el registro de giros con llave 1  indicada. }
  BEGIN
  status_io := BTRV (get_equal, bloque_pos_gir, wr_giros,
                     long_reg_gir, llave_gir, llave1        );
  END;

PROCEDURE LEE1_GE_GIROS (var wr_giros : r_giros;
                         var llave_gir: llave1_giros;
                         var status_io : integer );
  BEGIN
  status_io := BTRV (get_greater_equal,bloque_pos_gir, wr_giros,
                     long_reg_gir, llave_gir,       llave1 );
  END;

PROCEDURE LEE1_ULT_GIROS(var wr_giros  : r_giros;
                         var llave_gir : llave1_giros;
                         var status_io : integer);
  {Pone en wr_giros el siguiente registro del archivo, usando la llave 1.}
  BEGIN
  status_io :  BTRV (get_highest, bloque_pos_gir, wr_giros,
                     long_reg_gir, llave_gir, llave1);
  END;
```

```
PROCEDURE BORRA_GIROS(var wr_giros  : r_giros;
                          llave_gir : llave1_giros;
                      var status_io : integer);
   { Borra el registro  wr_giros de giros con  llave 1 indicada. }
   BEGIN
   status_io := BTRV (borra, bloque_pos_gir, wr_giros,
                      long_reg_gir, llave_gir, llave1        );
   END;

PROCEDURE LEE2_GIROS(var wr_giros  : r_giros;
                         llave_gir : llave2_giros;
                     var status_io : integer);
   { Pone en wr_giros el registro del archivo con llave 2 indicada. }
   BEGIN
   status_io := BTRV (get_equal, bloque_pos_gir, wr_giros,
                      long_reg_gir, llave_gir, llave2);
   END;

PROCEDURE LEE2_PRIMER_GIROS(var wr_giros  : r_giros;
                            var llave_gir : llave2_giros;
                            var status_io : integer);
   { Pone en wr_giros el registro del archivo con llave 2 indicada. }
   BEGIN
   status_io := BTRV (get_lowest, bloque_pos_gir, wr_giros,
                      long_reg_gir, llave_gir, llave2);
   END;

PROCEDURE LEE2_SIG_GIROS(var wr_giros  : r_giros;
                         var llave_gir : llave2_giros;
                         var status_io : integer);
   { Pone en wr_giros el registro del archivo con llave 2 indicada. }
   BEGIN
   status_io := BTRV (get_next, bloque_pos_gir, wr_giros,
                      long_reg_gir, llave_gir, llave2);
   END;

PROCEDURE LEE2_ULT_GIROS(var wr_giros  : r_giros;
                         var llave_gir : llave2_giros;
                         var status_io : integer);
   { Pone en wr_giros el siguiente registro del archivo, usando la llave 2.
}
   BEGIN
   status_io := BTRV (get_highest, bloque_pos_gir, wr_giros,
                      long_reg_gir, llave_gir, llave2);
   END;

PROCEDURE LEE3_PRIMER_GIROS(var wr_giros  : r_giros;
                            var llave_gir : llave3_giros;
                            var status_io : integer);
   { Pone en wr_giros el primer registro de giros con llave 3 indicada. }
   BEGIN
   status_io := BTRV (get_lowest, bloque_pos_gir, wr_giros,
                      long_reg_gir, llave_gir, llave3        );
   END;

PROCEDURE LEE3_SIG_GIROS(var wr_giros  : r_giros;
                         var llave_gir : llave3_giros;
                         var status_io : integer);
   { Pone en wr_giros el siguiente registro de giros con llave 3 indicada.}
   BEGIN
   status_io := BTRV (get_next, bloque_pos_gir, wr_giros,
                      long_reg_gir, llave_gir, llave3        );
   END;

PROCEDURE LEE3_GIROS(var wr_giros  : r_giros;
                         llave_gir : llave3_giros;
                     var status_io : integer);
   { Pone en wr_giros el registro de giros con llave 3  indicada. }
   BEGIN
   status_io := BTRV (get_equal, bloque_pos_gir, wr_giros,
                      long_reg_gir, llave_gir, llave3        );
   END;
```

```
PROCEDURE LEE3_GE_GIROS (var wr_giros : r_giros;
                         var llave_gir: llave3_giros;
                         var status_io : integer );
  BEGIN
  status_io := BTRV (get_greater_equal,bloque_pos_gir, wr_giros,
                     long_reg_gir, llave_gir,      llave3 );
  END;

PROCEDURE LEE3_ULT_GIROS(var wr_giros  : r_giros;
                         var llave_gir : llave3_giros;
                         var status_io : integer);
  {Pone en wr_giros el siguiente registro del archivo, usando la llave 3.}
  BEGIN
  status_io := BTRV (get_highest, bloque_pos_gir, wr_giros,
                     long_reg_gir, llave_gir, llave3);
  END;

BEGIN
  long_reg_gir := sizeof(r_giros);
END. { UNIT UGIROS}
```

A continuación se tienen los programas con funciones y procedimientos generales para el sistema:

Programa con funciones para procesos de captura:

```
{$E+,N+,V-}
UNIT UCAPTURA;
{ procedimientos para procesos de captura }
INTERFACE

   PROCEDURE CAPTURA_GIROS(movimiento: char);

   PROCEDURE CAPTURA_BENEFICIARIOS(movimiento : char);

   PROCEDURE CAPTURA_CORRESPONSAL(movimiento : char);

   PROCEDURE CAPTURA_CTAHABTE(movimiento : char);

   PROCEDURE CAPTURA_DIVISA(movimiento : char);

IMPLEMENTATION
   USES
     COLORDEF,
     TPCRT,
     TPDATE,
     TPENTRY,
     TPSTRING,
     UBEN_GIR,
     UCONTROL,
     UCORRESP,
     UCTAHAB,
     UDCLGIRO,
     UDIVISA,
     UGRALGIR,
     UMSJGIRO,
     URUTGIRO,
     UVALIDA,
     UGIROS;


FUNCTION TIPO_MOVIMIENTO(movimiento : char): string;
   VAR
     texto : string;
   begin
  texto := '';
  case movimiento of
      altas :
            texto:= 'ALTA ';
      bajas :
            texto:= 'BAJA ';
      cambios:
            texto:= 'CAMBIO ';
      end; {end case}
  TIPO_MOVIMIENTO := texto;
  end; {END TIPO_MOVIMIENTO}


PROCEDURE CAPTURA_GIROS (movimiento: char);
   VAR
     giros_scr          : ESRecord;
     wr_giros           : r_giros;
     llave_giros        : llave1_giros;
     comando_salida     : EStype;
     proceso_txt,
     string_aux         : string;
     salir              : boolean;
     status_io          : integer;
     rutina             : pointer;
     ceros              : string[8];
     fecha_aux          : tpo_fecha;

begin
```

```
ceros:= '00000000';
salir := off;
ABRE_ARCHIVOS(Giros, movimiento);
proceso_txt := TIPO_MOVIMIENTO(movimiento) + 'GIROS';
ENCABEZADO_PANTALLA(proceso_txt);
InitESRecord(giros_scr);
DEFINE_VENTANA_CAPTURA(giros_scr, 1, 6, 80, 24);
if movimiento = altas then
   begin
   rutina := @EXISTE_FOLIO;
   SetPreEditPtr(giros_scr, @SIG_FOLIO);
   end
else
   begin
   rutina := @VALIDA_FOLIO;
   SetPostEditPtr(giros_scr, @OBTIENE_FOLIO);
   end;
SetErrorPtr(giros_scr, @ErrorHandler);
SetWrapMode(giros_scr, ExitAtEdges);
SetRequired(On);
FECHA_SIGUIENTE(fecha_aux, status_io);
with wr_giros_cap do
   begin
   SetNumeric(on);
   AddStringField(giros_scr, 'FOLIO ', 2, 5,
                             '99999999', 2, 11, 8, 0,
                              rutina,
                              folio_gir_cap);
   SetNumeric(off);
   if movimiento = bajas then
      SetProtection(on);
   AddDateStField(giros_scr, 'FECHA EXPEDICION ', 2, 49,
                             'dd/mm/yy', 2, 66, 0,
                              fecha_gir_cap);

   AddStringField(giros_scr, 'DIVISA ', 5, 5,
                             CharStr('A', 3), 5,12, 3, 0,
                             @VALIDA_DIVISA,
                             divisa_gir_cap);

   AddExtField(giros_scr, 'MONTO ', 5, 46,
                          '$###,###,###,###,###.##',
                           5, 52, 0,
                           1, 999999999999999.99,
                           2, monto_gir_cap);

   AddStringField(giros_scr, 'CORRESPONSAL ', 8, 5,
                             CharStr('!', 10),
                             8, 18, 10, 0,
                             @valida_corresponsal,
                             corresponsal_gir_cap);

   AddStringField(giros_scr, 'BENEFICIARIO ', 11, 5,
                             CharStr('!', sizeof(tpo_beneficiario150)),
                             11, 19, 50, 0,
                             @valida_beneficiario,
                             beneficiario_gir_cap);
   SetRequired(off);
   SetNumeric(on);
   AddStringField(giros_scr, 'CUENTA CARGO ', 14, 5,
                             CharStr('!', 19),
                             14, 18, 19, 0,
                             @valida_cuenta,
                             cuenta_cargo_gir_cap);
   SetNumeric(off);
   AddStringField(giros_scr, 'CUENTA HABIENTE ', 14, 46,
                             CharStr('!', 10),
                             14, 62, 10, 0,
                             @valida_ctahabte,
                             cuentahabte_gir_cap);
   if movimiento = bajas then
      SetProtection(off);
   end; {end with}
repeat
  LINEA_GUIA('^ENTER - Aceptar        ESC - Salir');
```

```
move(blancos[1], wr_giros_cap, sizeof(r_giros_cap));
wr_giros_cap.fecha_gir_cap := TPOFECHA_FECHA(fecha_aux);
comando_salida := EditScreen(giros_scr, giros_scr.CurrentID, off);
case comando_salida of
   ESquit : begin
            salir := CONFIRMA('DESEA SALIR DE EDICION (S/N)?');
            if (not salir) and (movimiento <> altas) then
               ChangeProtection(giros_scr, 0, off);
            end;
   ESdone,
   ESnextrec,
   ESprevrec :
      begin
      move(blancos[1], wr_giros, sizeof(r_giros));
      with wr_giros, wr_giros_cap do
         begin
         FECHA_HOY(fecha_cap_gir, status_io);
         if status_io <> ok_io then
            DESPLIEGA_ERROR('PREPARA GIRO.FECHA_HOY', status_io);
         move(ceros[1], folio_gir[1], sizeof(tpo_folio));
         move(folio_gir_cap[1],
              folio_gir[sizeof(tpo_folio) - length(folio_gir_cap) +1],
              length(folio_gir_cap));
         string_aux :=FECHA_TPOFECHAST(fecha_gir_cap);
         move(string_aux[1], fecha_gir[1], sizeof(tpo_fecha));
         move(divisa_gir_cap[1], divisa_gir[1], sizeof(tpo_divisa));
         monto_gir := monto_gir_cap;
         move(corresponsal_gir_cap[1], corresponsal_gir[1],
              length(corresponsal_gir_cap));
         move(beneficiario_gir_cap[1], beneficiario_gir[1],
              length(beneficiario_gir_cap));
         move(cuenta_cargo_gir_cap[1], cuenta_cargo_gir[1],
               length(cuenta_cargo_gir_cap));
         move(cuentahabte_gir_cap[1], cuentahabte_gir[1],
               length(cuentahabte_gir_cap));
         wr_giros.estado_gir := negativo;
         with llave_giros do
            begin
            folio_girk := folio_gir;
            fecha_girk := fecha_gir;
            divisa_girk := divisa_girk;
            end;
         end; {end with}
      case movimiento of
        altas:
             begin
             ESCRIBE_GIROS(wr_giros, status_io);
             if status_io <> ok_io then
                DESPLIEGA_ERROR('CAPTURA_GIROS.ESCRIBE_GIROS',
                                status_io);
             end;
        bajas:
           begin
           if CONFIRMA('SE DA DE BAJA EL REGISTRO (S/N)?') then
              begin
              BORRA_GIROS(wr_giros, llave_giros, status_io);
              if status_io <> ok_io then
                 DESPLIEGA_ERROR('CAPTURA_GIROS.BORRA_GIROS',
                                 status_io);
              end;
           ChangeProtection(giros_scr, 0, off);
           end;
        cambios :
           begin
           if CONFIRMA('SE ACTUALIZA EL REGISTRO (S/N)?') then
              begin
              ACTUALIZA1_GIROS(wr_giros, status_io);
              if status_io <> ok_io then
                 DESPLIEGA_ERROR('CAPTURA_GIROS.ACTUALLIZA_GIROS',
                                 status_io);
              end;
           ChangeProtection(giros_scr, 0, off);
           end;
        end; {end case movimiento}
      end;
```

```
      end;{end case comando}
    giros_scr.CurrentID := 0;
  until salir;
  DisposeEditScreen(giros_scr);
  status_io := CIERRAS(BASE);
  if status_io <> ok_io then
     DESPLIEGA_ERROR('CAPTURA_GIROS.CIERRAS ', status_io);
end;   {END PROCEDURE CAPTURA_GIROS}


PROCEDURE CAPTURA_BENEFICIARIOS(movimiento : char);
  VAR
     benef_scr         : ESRecord;
     wr_benef          : r_benefgir;
     llave_benef       : llave1_ben;
     comando_salida    : EStype;
     salir             : boolean;
     proceso_txt       : string;
     rutina            : pointer;
     status_io         : integer;

begin
  salir := off;
  ABRE_ARCHIVOS(Beneficiarios, movimiento);
  proceso_txt := TIPO_MOVIMIENTO(movimiento) + 'BENEFICIARIOS';
  ENCABEZADO_PANTALLA(proceso_txt);
  InitESRecord(benef_scr);
  DEFINE_VENTANA_CAPTURA(benef_scr, 1, 6, 80, 24);
  if movimiento = altas then
     rutina := @EXISTE_BENEFICIARIO
  else
     begin
     rutina := @VALIDA_BENEFICIARIO;
     SetPostEditPtr(benef_scr, @OBTIENE_BENEF);
     end;
  SetErrorPtr(benef_scr, @ErrorHandler);
  SetWrapMode(benef_scr, ExitAtEdges);
  with wr_benef_cap do
     begin
     SetRequired(on);
     AddStringField(benef_scr, 'CLAVE BENEFICIARIO ', 3, 25,
                                '!!!!!!!!!!',
                                3, 44, 10, 0, rutina,
                                clave_benef_cap);
     if movimiento = bajas then
        SetProtection(on);
     AddStringField(benef_scr, 'NOMBRE BENEFICIARIO', 6, 15,
                                CharStr('A', sizeof(tpo_beneficiario150)),
                                7,15, 50, 0, nil,
                                nombre_benef_cap);
     SetRequired(off);

     AddExtField(benef_scr, 'MONTO ', 10, 24,
                            '$###,###,###,###,###.##',
                            10, 30, 0,
                            0, 999999999999999.99,
                            2, monto_benef_cap);

     AddStringField(benef_scr, 'EXTRA ', 13, 24,
                                CharStr('!', 25),
                                13, 30, 25, 0, nil,
                                extra_benef_cap);
     if movimiento = bajas then
        SetProtection(off);
     end; {END WITH}
  repeat
    LINEA_GUIA('ENTER - Aceptar        ESC - Salir');
    move(blancos[1], wr_benef_cap, sizeof(r_benef_cap));
    comando_salida := EditScreen(benef_scr, 0, off);
    case comando_salida of
       ESquit :begin
               salir := CONFIRMA('DESEA SALIR DE EDICION (S/N)?');
               if (not salir) and (movimiento <> altas) then
                  ChangeProtection(benef_scr, 0, off);
               end;
```

```
        ESdone,
        ESnextrec,
        ESprevrec :
           begin
           move(blancos[1], wr_benef, sizeof(r_benefgir));
           with wr_benef, wr_benef_cap do
              begin
              move(clave_benef_cap[1], clave_ben[1],
                   length(clave_benef_cap));
              move(nombre_benef_cap[1], nombre_ben[1],
                   length(nombre_benef_cap));
              move(extra_benef_cap[1], extra_ben[1],
                   length(extra_benef_cap));
              monto_ben := monto_benef_cap;
              llave_benef := clave_ben;
              end; {end with}
           case movimiento of
             altas:
                  begin
                  ESCRIBE_BENEF(wr_benef, status_io);
                  if status_io <> ok_io then
                     DESPLIEGA_ERROR('CAPTURA_BENEF.ESCRIBE_BENEF',
                                     status_io);
                  end;
             bajas:
                begin
                if CONFIRMA('SE DA DE BAJA EL REGISTRO (S/N)?')  then
                   begin
                   BORRA_BENEF(wr_benef, llave_benef, status_io);
                   if status_io <> ok_io then
                      DESPLIEGA_ERROR('CAPTURA_BENEF.BORRA_BENEF',
                                      status_io);
                   end;
                ChangeProtection(benef_scr, 0, off);
                end;
             cambios :
                begin
                if CONFIRMA('SE ACTUALIZA EL REGISTRO (S/N)?') then
                   begin
                   ACTUALIZA_BENEF(wr_benef, status_io);
                   if status_io <> ok_io then
                      DESPLIEGA_ERROR('CAPTURA_BENEF.ACTUALLIZA_BENEF',
                                      status_io);
                   end;
                ChangeProtection(benef_scr, 0, off);
                end;
             end; {end case movimiento}
           end;
        end;{end case comando}
     until salir;
     DisposeEditScreen(benef_scr);
     status_io := CIERRAS(BASE);
     if status_io <> ok_io then
        DESPLIEGA_ERROR('CAPTURA_BENEFICIARIO.CIERRAS', status_io);
  end;   {END PROCEDURE CAPTURA_BENEF}

  PROCEDURE CAPTURA_CORRESPONSAL(movimiento : char);
     VAR
        corresp_scr     : ESRecord;
        wr_corresp      : r_corresp;
        llave_corresp   : llave1_corr;
        comando_salida  : EStype;
        solo_lectura,
        salir           : boolean;
        proceso_txt     : string;
        rutina          : pointer;
        status_io       : integer;

  begin
     salir := off;
     ABRE_ARCHIVOS(Corresponsales, movimiento);
     proceso_txt :=  TIPO_MOVIMIENTO(movimiento)+ 'CORRESPONSALES';
     ENCABEZADO_PANTALLA(proceso_txt);
     InitESRecord(corresp_scr);
     DEFINE_VENTANA_CAPTURA(corresp_scr, 1, 6, 80, 24);
```

```
if movimiento = altas then
   rutina := @EXISTE_CORRESPONSAL
else
   begin
   rutina := @VALIDA_CORRESPONSAL;
   SetPostEditPtr(corresp_scr, @OBTIENE_CORRESP);
   end;
SetErrorPtr(corresp_scr, @ErrorHandler);
SetWrapMode(corresp_scr, ExitAtEdges);
with wr_corr_cap do
   begin
   SetRequired(on);
   AddStringField(corresp_scr, 'CLAVE CORRESPONSAL ', 3, 25,
                               '!!!!!!!!!!',
                               3, 44, 10, 0, rutina,
                               clave_corr_cap);
   if movimiento = bajas then
      SetProtection(on);
   AddStringField(corresp_scr, 'NOMBRE CORRESPONSAL', 5, 15,
                               CharStr('!', 45),
                               6,15, 45, 0, nil,
                               nombre_corr_cap);
   AddStringField(corresp_scr, 'DIRECCION CORRESPONSAL', 8, 15,
                               CharStr('!', 45),
                               9,15, 45, 0, nil,
                               direccion_corr_cap);

   AddStringField(corresp_scr, 'CUENTA CORRESPONSAL ', 12, 20,
                               CharStr('!', 19),
                               12, 40, 19, 0, nil,
                               cuenta_corr_cap);
   if movimiento = bajas then
      SetProtection(off);
   SetRequired(off);
   end; {end with}
repeat
  LINEA_GUIA('^ENTER - Aceptar        ESC - Salir');
  move(blancos[1], wr_corr_cap, sizeof(r_corr_cap));
  comando_salida := EditScreen(corresp_scr, corresp_scr.CurrentID, off);
  case comando_salida of
     ESquit : begin
              salir := CONFIRMA('DESEA SALIR DE EDICION (S/N)?');
              if (not salir) and (movimiento <> altas) then
                 ChangeProtection(corresp_scr, 0, off);
              end;
     ESdown,
     ESnextrec,
     ESprevrec :
        begin
        move(blancos[1], wr_corresp, sizeof(r_corresp));
        with wr_corresp, wr_corr_cap do
           begin
           move(clave_corr_cap[1], clave_corr[1],
                length(clave_corr_cap));
           move(nombre_corr_cap[1], nombre_corr[1],
                length(nombre_corr_cap));
           move(direccion_corr_cap[1], direccion_corr[1],
                length(direccion_corr_cap));
           move(cuenta_corr_cap[1], cuenta_corr[1],
                length(cuenta_corr_cap));
           llave_corresp := clave_corr;
           end; {end with}
        case movimiento of
          altas:
               begin
               ESCRIBE_CORRESP(wr_corresp, status_io);
               if status_io <> ok_io then
                  DESPLIEGA_ERROR('CAPTURA_CORRESPONSAL.ESCRIBE_CORR',
                                  status_io);
               end;
          bajas:
            begin
            if CONFIRMA('SE DA DE BAJA EL REGISTRO (S/N)?')  then
               begin
               BORRA_CORRESP(wr_corresp, llave_corresp, status_io);
```

```
                if status_io <> ok_io then
                   DESPLIEGA_ERROR('CAPTURA_CORRESP.BORRA_CORRESP',
                                   status_io);
                end;
             ChangeProtection(corresp_scr, 0, off);
             end;
          cambios :
             begin
             if CONFIRMA('SE ACTUALIZA EL REGISTRO (S/N)?') then
                begin
                ACTUALIZA_CORRESP(wr_corresp, status_io);
                if status_io <> ok_io then
                   DESPLIEGA_ERROR('CAPTURA_CORRESP.ACTUALLIZA_CORRESP',
                                   status_io);
                end;
             ChangeProtection(corresp_scr, 0, off);
             end;
          end; {end case movimiento}
       end;
     end;{end case comando}
  corresp_scr.CurrentID := 0;
until salir;
DisposeEditScreen(corresp_scr);
status_io := CIERRAS(BASE);
if status_io <> ok_io then
   DESPLIEGA_ERROR('CAPTURA_CORRESPONSAL.CIERRAS ', status_io);
end;   {END PROCEDURE CAPTURA_CORRESPSONSAL}


PROCEDURE CAPTURA_CTAHABTE(movimiento : char);
  VAR
    ctahabte_scr     : ESRecord;
    wr_ctahabte      : r_ctahabte;
    llave_ctahabte   : llave1_ctahabte;
    comando_salida   : EStype;
    salir            : boolean;
    proceso_txt      : string;
    status_io        : integer;
    rutina           : pointer;

begin
  salir := off;
  ABRE_ARCHIVOS(Cuentahabtes, movimiento);
  proceso_txt :=  TIPO_MOVIMIENTO(movimiento) + 'CUENTAHABIENTES';
  ENCABEZADO_PANTALLA(proceso_txt);
  InitESRecord(ctahabte_scr);
  DEFINE_VENTANA_CAPTURA(ctahabte_scr, 1, 6, 80, 24);
  if movimiento = altas then
     rutina := @EXISTE_CTAHABTE
  else
     begin
     rutina := @VALIDA_CTAHABTE;
     SetPostEditPtr(ctahabte_scr, @OBTIENE_CTAHABTE);
     end;
  SetErrorPtr(ctahabte_scr, @ErrorHandler);
  SetWrapMode(ctahabte_scr, ExitAtEdges);
  with wr_ctahabte_cap do
     begin
     SetRequired(on);
     AddStringField(ctahabte_scr, 'CLAVE CUENTAHABIENTE ', 5, 24,
                                  '!!!!!!!!!!',
                                   5, 45, 10, 0, rutina,
                                   clave_ctahabte_cap);

     if movimiento = bajas then
        SetProtection(on);
     AddStringField(ctahabte_scr, 'NOMBRE CUENTAHABIENTE', 8, 15,
                                   CharStr('!', 100),
                                   9,15, 50, 0, nil,
                                   nombre_ctahabte_cap);

     AddStringField(ctahabte_scr, 'CUENTA TESOFE ', 12, 23,
                                   CharStr('!', 19),
                                   12, 37, 19, 0, nil,
                                   cta_tesofe_ctahabte_cap);
```

```
   SetRequired(off);

   AddStringField(ctahabte_scr, 'CUENTA ', 14, 27,
                                CharStr('!', 19),
                                14, 34, 19, 0, nil,
                                cta_ctahabte_cap);

   AddStringField(ctahabte_scr, 'DIVISA ', 16, 27,
                                CharStr('!', 3),
                                16, 34, 3, 0, @valida_divisa,
                                divisa_ctahabte_cap);

   if movimiento = bajas then
      SetProtection(off);
  end; {end with}
repeat
  LINEA_GUIA('^ENTER - Aceptar       ESC - Salir');
  move(blancos[1], wr_ctahabte_cap, sizeof(r_ctahabte_cap));
  comando_salida := EditScreen(ctahabte_scr, ctahabte_scr.CurrentID,off);
  case comando_salida of
     ESquit : begin
              salir := CONFIRMA('DESEA SALIR DE EDICION (S/N)?');
              if (not salir) and (movimiento <> altas) then
                 ChangeProtection(ctahabte_scr, 0, off);
              end;
     ESdone,
     ESnextrec,
     ESprevrec :
        begin
        move(blancos[1], wr_ctahabte, sizeof(r_ctahabte));
        with wr_ctahabte, wr_ctahabte_cap do
           begin
           move(clave_ctahabte_cap[1], clave_ctahabte[1],
                length(clave_ctahabte_cap));
           move(nombre_ctahabte_cap[1], nombre_ctahabte[1],
                length(nombre_ctahabte_cap));
           move(cta_tesofe_ctahabte_cap[1], cta_tesofe_ctahabte[1],
                length(cta_tesofe_ctahabte_cap));
           move(cta_ctahabte_cap[1], cuenta_ctahabte[1],
                length(cta_ctahabte_cap));
           move(divisa_ctahabte_cap[1], divisa_monto_ctahabte[1],
                length(divisa_ctahabte_cap));
           llave_ctahabte:= clave_ctahabte;
           end; {end with}
        case movimiento of
          altas:
               begin
               ESCRIBE_CTAHABTE(wr_ctahabte, status_io);
               if status_io <> ok_io then
                DESPLIEGA_ERROR('CAPTURA_CUENTAHABIENTE.ESCRIBE_CTAHABTE'
                                , status_io);
               end;
          bajas:
             begin
             if CONFIRMA('SE DA DE BAJA EL REGISTRO (S/N)?')  then
               begin
               BORRA_CTAHABTE(wr_ctahabte, llave_ctahabte, status_io);
               if status_io <> ok_io then
                  DESPLIEGA_ERROR('CAPTURA_CUENTAHABTE.BORRA_CTAHABTE',
                                  status_io);
               end;
             ChangeProtection(ctahabte_scr, 0, off);
             end;
          cambios :
             begin
             if CONFIRMA('SE ACTUALIZA EL REGISTRO (S/N)?') then
               begin
               ACTUALIZA_CTAHABTE(wr_ctahabte, status_io);
               if status_io <> ok_io then
                 DESPLIEGA_ERROR('CAPTURA_CTAHABTE.ACTUALLIZA_CTAHABTE',
                                 status_io);
               end;
             ChangeProtection(ctahabte_scr, 0, off);
             end;
          end; {end case movimiento}
```

```
            end;
          end;{end case comando}
        ctahabte_scr.CurrentID := 0;
      until salir;
      DisposeEditScreen(ctahabte_scr);
      status_io := CIERRAS(BASE);
      if status_io <> ok_io then
         DESPLIEGA_ERROR('CAPTURA_CUENTAHABTIE.CIERRAS ', status_io);
    end;   {END PROCEDURE CAPTURA_CTAHABTE}

    PROCEDURE CAPTURA_DIVISA(movimiento : char);
      VAR
        divisa_scr       : ESRecord;
        wr_divisa        : r_divisa;
        llave_divisa     : llavel_div;
        comando_salida   : EStype;
        solo_lectura,
        salir            : boolean;
        proceso_txt      : string;
        rutina           : pointer;
        status_io        : integer;

    begin
      salir := off;
      ABRE_ARCHIVOS(Divisas, movimiento);
      proceso_txt :=  TIPO_MOVIMIENTO(movimiento)+ 'DIVISAS';
      ENCABEZADO_PANTALLA(proceso_txt);
      InitESRecord(divisa_scr);
      DEFINE_VENTANA_CAPTURA(divisa_scr, 1, 6, 80, 24);
      if movimiento = altas then
         rutina := @EXISTE_DIVISA
      else
         begin
         rutina := @VALIDA_DIVISA;
         SetPostEditPtr(divisa_scr, @OBTIENE_DIVISA);
         end;
      SetErrorPtr(divisa_scr, @ErrorHandler);
      SetWrapMode(divisa_scr, ExitAtEdges);
      with wr_divisa_cap do
         begin
         SetRequired(on);
         AddStringField(divisa_scr, 'DIVISA ', 5, 35,
                                    'AAA',
                                    5, 42, 3, 0, rutina,
                                    clave_div_cap);
         if movimiento = baja then
            SetProtection(on);
         AddStringField(divisa_scr, 'NOMBRE DIVISA (ESPAÑOL)', 8, 15,
                                    CharStr('!', 100),
                                    9,15, 50, 0, nil,
                                    nombre_esp_div_cap);

         AddStringField(divisa_scr, 'NOMBRE DIVISA (INGLES)', 12, 15,
                                        CharStr('!', 100),
                                        13, 15, 50, 0, nil,
                                        nombre_ing_div_cap);
         if movimiento = baja then
            SetProtection(off);
         SetRequired(off);
         end; {end with}
      repeat
        LINEA_GUIA('^ENTER - Aceptar       ESC - Salir');
        move(blancos[1], wr_divisa_cap, sizeof(r_divisa_cap));
        comando_salida := EditScreen(divisa_scr, divisa_scr.CurrentID, off);
        case comando_salida of
           ESquit : begin
                    salir := CONFIRMA('DESEA SALIR DE EDICION (S/N)?');
                    if (not salir) and (movimiento <> altas) then
                       ChangeProtection(divisa_scr, 0, off);
                    end;
           ESdone,
           ESnextrec,
           ESprevrec :
              begin
              move(blancos[1], wr_divisa, sizeof(r_divisa));
```

```
        with wr_divisa, wr_divisa_cap do
           begin
           move(clave_div_cap[1], clave_div[1], sizeof(tpo_divisa));
           move(nombre_esp_div_cap[1], nombre_esp_div[1],
                length(nombre_esp_div_cap));
           move(nombre_ing_div_cap[1], nombre_ing_div[1],
                length(nombre_esp_div_cap));
           llave_divisa := clave_div;
           end; {end with}
        case movimiento of
          altas:
               begin
               ESCRIBE_DIVISA(wr_divisa, status_io);
               if status_io <> ok_io then
                  DESPLIEGA_ERROR('CAPTURA_DIVISA.ESCRIBE_DIVISA',
                                  status_io);
               end;
          bajas:
             begin
             if CONFIRMA('SE DA DE BAJA EL REGISTRO (S/N)?') then
                begin
                BORRA_DIVISA(wr_divisa, llave_divisa, status_io);
                if status_io <> ok_io then
                   DESPLIEGA_ERROR('CAPTURA_DIVISA.BORRA_DIVISA',
                                   status_io);
                end;
             ChangeProtection(divisa_scr, 0, off);
             end;
          cambios :
             begin
             if CONFIRMA('SE ACTUALIZA EL REGISTRO (S/N)?') then
                begin
                ACTUALIZA_DIVISA(wr_divisa, status_io);
                if status_io <> ok_io then
                   DESPLIEGA_ERROR('CAPTURA_DIVISA.ACTUALLIZA_DIVISA',
                                   status_io);
                end;
             ChangeProtection(divisa_scr, 0, off);
             end;
          end; {end case movimiento}
        end;
      end;{end case comando}
    divisa_scr.CurrentID := 0;
  until salir;
  DisposeEditScreen(divisa_scr);
  status_io := CIERRAS(BASE);
  if status_io <> ok_io then
     DESPLIEGA_ERROR('CAPTURA_DIVISA.CIERRAS ', status_io);
end;   {END PROCEDURE CAPTURA_DIVISA}
begin
  TextAttr := WhiteonBlack;
end.
```

Programa con funciones para leer el tipo de usuario en sesión:

```
{$I-,V-,R-,N+,E+,D+}
UNIT UCON_USU;
{ procedimiento para leer claves de acceso y actualizaciones de usuarios }

INTERFACE
   USES
       UDCLGIRO;

   PROCEDURE LEE_CLAVE_USUARIO (var clave : tpo_operador;
                                var grupo : tpo_grupo_usuarios);

   PROCEDURE ACTUALIZA_CLAVE_USUARIO(clave : tpo_operador;
                                     grupo : tpo_grupo_usuarios);


IMPLEMENTATION

  USES UGRALGIR;


   PROCEDURE LEE_CLAVE_USUARIO (var clave : tpo_operador;
                                var grupo : tpo_grupo_usuarios);

      VAR
         arch_usu : text;
         clave_str : string[6];
         status_io : integer;
         grupo_str : string[2];

      BEGIN
      base := nil;
      assign(arch_usu, arch_control_usuario);
      reset(arch_usu);
      status_io := IOresult;
      if status_io <> ok_io then
         DESPLIEGA_ERROR('RUTINA LEE_STATUS_USUARIO (reset) ',status_io);

      clave_str := '      ';
      grupo_str := '  ';
      readln(arch_usu,clave_str);
      status_io := IOresult;
      if status_io <> ok_io then
         DESPLIEGA_ERROR('RUTINA LEE_STATUS_USUARIO (readln 1) ',
                         status_io);

      readln(arch_usu,grupo_str);
      status_io := IOresult;
      if status_io <> ok_io then
         DESPLIEGA_ERROR('RUTINA LEE_STATUS_USUARIO (readln 2) ',
                         status_io);
      close(arch_usu);
      status_io := IOresult;
      if status_io <> ok_io then
         DESPLIEGA_ERROR('RUTINA LEE_STATUS_USUARIO (close) ',status_io);

      move(clave_str[1], clave[1], 6);
      move(grupo_str[1], grupo[1], 2);

      END;


    PROCEDURE ACTUALIZA_CLAVE_USUARIO (clave : tpo_operador;
                                       grupo : tpo_grupo_usuarios);

      VAR
         arch_usu : text;
         clave_str : string[6];
         status_io : integer;

      BEGIN
      assign(arch_usu, arch_control_usuario);
```

```
      rewrite(arch_usu);
      status_io := IOresult;
      if status_io <> ok_io then
         DESPLIEGA_ERROR('RUTINA ACTUALIZA_STATUS_USUARIO (rewrite) ',
                         status_io);

      writeln(arch_usu,clave);
      status_io := IOresult;
      if status_io <> ok_io then
         DESPLIEGA_ERROR('RUTINA ACTUALIZA_STATUS_USUARIO (writeln 1) ',
                         status_io);
      writeln(arch_usu,grupo);
      status_io := IOresult;
      if status_io <> ok_io then
         DESPLIEGA_ERROR('RUTINA ACTUALIZA_STATUS_USUARIO (writeln 2) ',
                         status_io);
      close(arch_usu);
      status_io := IOresult;
      if status_io <> ok_io then
         DESPLIEGA_ERROR('RUTINA LEE_STATUS_USUARIO (close) ',status_io);
      END;
END.
```

Programa con funciones para el control de los giros:

```
unit UCTRLGIR;

{$I-,V-,R-,N+,E+,D+}

interface

USES
  UDCLGIRO;

CONST

  { Nombre del archivo en MS-DOS: }
  archivo_ctrl_gir= 'f:\giros\datos\ctrl_gir.dat';

  { Posibles valores del campo estado_gir: }
  giro_generado    = 'G';      { Estado asignado al generar el giro.     }
  giro_cancelado   = 'C';      { giro cancelado por la oficina de trámite de
                                 cambios}

TYPE

  tpo_monto_gir= array[1..15] of Char;


  LLAVE1_CTRL_GIR = tpo_folio; { Sobre el campo num_giro}

  LLAVE2_CTRL_GIR = tpo_folio; { Sobre el campo folio_giro}


  R_CTRL_GIR = record
                    {-Información básica del mensaje}
                    num_giro_ctrgir     : tpo_folio; { Número del giro}{8}
                    folio_ctrgir        : tpo_folio; { folio del giro} {8}
                    estado_giro_ctrgir : Char; { Gen | Cancel     }     {1}
                    fecha_gen_ctrgir    : tpo_fecha;                    {8}
                    hora_gen_ctrgir     : tpo_hora;                     {4}
                    corresponsal_ctrgir: tpo_corresponsal;             {10}
                    divisa_ctrgir       : tpo_divisa;                   { 3}
                    monto_ctrgir        : tpo_monto_gir;               {15}
                    beneficiario_ctrgir: tpo_beneficiario50;           {50}
                    cuentahabte_ctrgir : tpo_corresponsal              {10}
                  end;           { R_CTRL_GIR }                       {217}
```

```
function ABRE_CTRL_GIR(modo : Char) : Integer;
function CIERRA_CTRL_GIR : Integer;
procedure ACTUALIZA1_CTRL_GIR(wr_ctrl_gir : R_CTRL_GIR;
                              var status_io : Integer);
procedure ACTUALIZA2_CTRL_GIR(wr_ctrl_gir : R_CTRL_GIR;
                              var status_io : Integer);
procedure ESCRIBE_CTRL_GIR(wr_ctrl_gir : R_CTRL_GIR;
                           var status_io : Integer);
procedure ESCRIBE2_CTRL_GIR(wr_ctrl_gir : R_CTRL_GIR;
                            var status_io : Integer);
procedure LEE_PRIMER_CTRL_GIR(var wr_ctrl_gir : R_CTRL_GIR;
                              var llave_ctrgir : LLAVE1_CTRL_GIR;
                              var status_io : Integer);
procedure LEE_SIGUIENTE_CTRL_GIR(var wr_ctrl_gir : R_CTRL_GIR;
                              var llave_ctrgir : LLAVE1_CTRL_GIR;
                                 var status_io : Integer);
procedure LEE1_CTRL_GIR(var wr_ctrl_gir : R_CTRL_GIR;
                            llave_ctrgir : LLAVE1_CTRL_GIR;
                            var status_io : Integer);
procedure LEE1_ULT_CTRL_GIR(var wr_ctrl_gir : R_CTRL_GIR;
                              var status_io : Integer);
procedure BORRA_CTRL_GIR(var wr_ctrl_gir : R_CTRL_GIR;
                             llave_ctrgir : LLAVE1_CTRL_GIR;
                             var status_io : Integer);
procedure LEE2_CTRL_GIR(var wr_ctrl_gir : R_CTRL_GIR;
                            llave_ctrgir : LLAVE2_CTRL_GIR;
                            var status_io : Integer);

implementation


USES
  UBTRIEVE;

VAR

  long_reg_ctrgir : Integer;
  bloque_pos_ctrgir : array[1..128] of Byte;


  function ABRE_CTRL_GIR(modo : Char) : Integer;
    { Abre el archivo de control de giros. }
  VAR
    dueno_archivo : String[10];
    nombre_archivo : String[80];
    modo_int : Integer;
  begin
    dueno_archivo := #0;
    nombre_archivo := archivo_ctrl_gir + #0;
    case modo of
      modo_lectura : modo_int := solo_lectura;
      modo_recuperacion : modo_int := con_recuperacion;
    end;                                {case}
    ABRE_CTRL_GIR := BTRV(abre_archivo, bloque_pos_ctrgir,
                          dueno_archivo[1], long_reg_ctrgir,
                          nombre_archivo[1], modo_int);
  end;                                  { ABRE_CTRL_GIR }


  function CIERRA_CTRL_GIR : Integer;
  begin
    CIERRA_CTRL_GIR := BTRV(cierra_archivo, bloque_pos_ctrgir, dummy,
                            dummy, dummy, dummykey);
  end;


  procedure ACTUALIZA1_CTRL_GIR(wr_ctrl_gir : R_CTRL_GIR;
                                VAR status_io : Integer);
    { Update con llave 1 }
  VAR
    llave1_control : LLAVE1_CTRL_GIR;
  begin
    status_io :  BTRV(actualiza, bloque_pos_ctrgir, wr_ctrl_gir,
                      long_reg_ctrgir, llave1_control, llave1);
  end;
```

```
procedure ACTUALIZA2_CTRL_GIR(wr_ctrl_gir : R_CTRL_GIR;
                              var status_io : Integer);
  { Update con llave 2 }
VAR
  llave2_control : LLAVE2_CTRL_GIR;
begin
  status_io := BTRV(actualiza, bloque_pos_ctrgir, wr_ctrl_gir,
                    long_reg_ctrgir, llave2_control, llave2);
end;

procedure ESCRIBE_CTRL_GIR(wr_ctrl_gir : R_CTRL_GIR;
                           var status_io : Integer);
VAR
  llave1_control : LLAVE1_CTRL_GIR;
begin
  status_io := BTRV(escribe, bloque_pos_ctrgir, wr_ctrl_gir,
                    long_reg_ctrgir, llave1_control, llave1);
end;


procedure ESCRIBE2_CTRL_GIR(wr_ctrl_gir : R_CTRL_GIR;
                            var status_io : Integer);
VAR
  llave2_control : LLAVE2_CTRL_GIR;
begin
  status_io := BTRV(escribe, bloque_pos_ctrgir, wr_ctrl_gir,
                    long_reg_ctrgir, llave2_control, llave2);
end;

procedure LEE_PRIMER_CTRL_GIR(var wr_ctrl_gir : R_CTRL_GIR;
                              var llave_ctrgir : LLAVE1_CTRL_GIR;
                              var status_io : Integer);
  { Pone en wr_ctrl_gir el primer registro de ctrl_gir. }
begin
  status_io := BTRV(get_lowest, bloque_pos_ctrgir, wr_ctrl_gir,
                    long_reg_ctrgir, llave_ctrgir, llave1);
end;

procedure LEE_SIGUIENTE_CTRL_GIR(var wr_ctrl_gir : R_CTRL_GIR;
                                 var llave_ctrgir : LLAVE1_CTRL_GIR;
                                  var status_io : Integer);
 { Pone en wr_ctrl_gir el siguiente registro de control_giros con
   llave 1 indicada. }
begin
  status_io := BTRV(get_next, bloque_pos_ctrgir, wr_ctrl_gir,
                    long_reg_ctrgir, llave_ctrgir, llave1);
end;

procedure LEE1_CTRL_GIR(var wr_ctrl_gir : R_CTRL_GIR;
                            llave_ctrgir : LLAVE1_CTRL_GIR;
                            var status_io : Integer);
 { Pone en wr_ctrl_gir el registro de control_giros con llave 1
   indicada. }
begin
  status_io := BTRV(get_equal, bloque_pos_ctrgir, wr_ctrl_gir,
                    long_reg_ctrgir, llave_ctrgir, llave1);
end;


procedure LEE1_ULT_CTRL_GIR(var wr_ctrl_gir : R_CTRL_GIR;
                            var status_io : Integer);
 { Pone en wr_ctrl_gir el siguiente registro del archivo,
   usando la llave 1. }
VAR
  llave1_control : LLAVE1_CTRL_GIR;
begin
  status_io := BTRV(get_highest, bloque_pos_ctrgir, wr_ctrl_gir,
                    long_reg_ctrgir, llave1_control, llave1);
end;


procedure BORRA_CTRL_GIR(var wr_ctrl_gir : R_CTRL_GIR;
                            llave_ctrgir : LLAVE1_CTRL_GIR;
                            var status_io : Integer);
```

```
    { Borra el registro  wr_ctrl_gir de control_giros con
      llave 1 indicada. }
  begin
    status_io := BTRV(borra, bloque_pos_ctrgir, wr_ctrl_gir,
                      long_reg_ctrgir, llave_ctrgir, llave1);
  end;

  procedure LEE2_CTRL_GIR(var wr_ctrl_gir : R_CTRL_GIR;
                              llave_ctrgir : LLAVE2_CTRL_GIR;
                              var status_io : Integer);
    { Pone en wr_ctrl_gir el registro del archivo con llave 2 indicada. }
  begin
    status_io := BTRV(get_equal, bloque_pos_ctrgir, wr_ctrl_gir,
                      long_reg_ctrgir, llave_ctrgir, llave2);
  end;

begin
  long_reg_ctrgir := SizeOf(R_CTRL_GIR);
end.                                { UNIT UCTLGIR}
```

Programa con declaraciónes generales del SISGIRO:

```
{$I-,V+,R+,N+,E+,D+}
UNIT UGRALGIR;
  { Unidad que contiene las rutinas generales del sistema }

INTERFACE

   USES
        UDCLGIRO;

 CONST
   path_ejecutable = 'f:\giros\sistema\';

 TYPE
   FuncAK                  = function ( x : char;
                                        y : string08 ):integer;


   PROCEDURE MUESTRA_ERROR (  Texto     : string;
                              Estado_Io : integer );


   FUNCTION ABRES ( var Ba   : Enlace;
                        Ide  : Cadena8;
                        A    : FuncA;
                        modo : char;
                        C    : FuncC):integer;

   FUNCTION ABRESK ( var Ba   : Enlace;
                         Ide  : Cadena8;
                         AK   : FuncAK;
                         modo : char;
                         llave: string08;
                         C    : FuncC):integer;

   FUNCTION CIERRAS ( var Ba : Enlace):integer;

   PROCEDURE DESPLIEGA_ERROR (texto : string ;
                              codigo: integer);

   PROCEDURE ERROR_SISTEMA (rutina,
                            status: string_alfa);


   FUNCTION REDONDEA (eq_monto  : extended;
                      decimales : byte ) : extended;


   PROCEDURE DA_FECHA_Y_HORA_MSDOS (var fecha_msdos : tpo_fecha;
                                    var hora_msdos  : tpo_hora);


      PROCEDURE EJECUTA (programa      : string;
                         comandos      : string;
                         nombre        : string;
                         proceso       : longint;
                         descarga_prog : boolean;
                         usa_command   : boolean);


IMPLEMENTATION

   USES  CHAIN,
         TPSTRING,
         TPDATE,
         UBTRIEVE,
         TPCRT,
         DOS,
         TPDOS,
         UCONTROL,
         UMSJGIRO,
```

```
    UPROGIR;

PROCEDURE MUESTRA_ERROR ( Texto      : string;
                          Estado_Io  : integer );
  BEGIN
  FastWrite ( texto + form('###',Estado_Io), wherey, 15, 15);
  writeln;
  END;



FUNCTION ABRES ( var Ba   : Enlace;
                     Ide  : Cadena8;
                     A    : FuncA;
                     modo : char;
                     C    : FuncC):integer;
  VAR
     Estado: integer;
     Ar    : Enlace;
  BEGIN
  if Ba = nil then
     RESETEA_BTRIEVE;
  estado := A(Modo);
  if estado= 0 then
     begin
     new(Ar);
     Ar^.Ident:=Ide;
     Ar^.Funcion:=C;
     Ar^.Sigue:=Ba;
     Ba:=Ar;
     end
  else
     MUESTRA_ERROR (  'Error en el archivo '+Ide+' ',Estado);
  ABRES:=Estado;
  END;

FUNCTION ABRESK ( var Ba    : Enlace;
                      Ide   : Cadena8;
                      AK    : FuncAk;
                      modo  : char;
                      llave: string08;
                      C     : FuncC):integer;
  VAR
     Estado: integer;
     Ar    : Enlace;
  BEGIN
  if Ba = nil then
     RESETEA_BTRIEVE;
  estado := AK (Modo,llave);
  if estado= 0 then
     begin
     new(Ar);
     Ar^.Ident:=Ide;
     Ar^.Funcion:=C;
     Ar^.Sigue:=Ba;
     Ba:=Ar;
     end
  else
     MUESTRA_ERROR (  'Error en el archivo '+Ide+' ',Estado);
  ABRESK:=Estado;
  END;

FUNCTION CIERRA_ARCHIVO ( var Ba       : Enlace;
                              Archivo : Cadena8):integer;
  VAR
     Estado,Cont : integer;
     Base_Original,
     Anterior,
     Paso,
     Pa          : Enlace;
  BEGIN
  Cont:=0;
  if Ba^.Ident = Archivo then
     begin
```

```
      Estado:=Ba^.Funcion;
      if Estado <> 0 then
         begin
         MUESTRA_ERROR (  'Error en el archivo '+Ba^.ident+' ',Estado);
         inc (Cont);
         end;
      Pa:=Ba;
      Ba:=Ba^.Sigue;
      dispose (Pa);
      end
   else
      begin
      Base_Original:=Ba;
      while Ba <> nil do
         begin
         Anterior:=Ba;
         Ba:=Ba^.Sigue;
         if Ba^.Ident = Archivo then
            begin
            Estado:=Ba^.Funcion;
            if Estado <> 0 then
               begin
               MUESTRA_ERROR (  'Error en el archivo '+Ba^.ident+'
                                ',Estado);
               inc (Cont);
               end;
            Pa:=Ba;
            Paso:=Ba^.sigue;
            Ba:=Anterior;
            Anterior^.sigue:=Paso;
            dispose (Pa);
            end;
         end;
      end;
   CIERRA_ARCHIVO:=Cont;
   END;


FUNCTION CIERRAS ( var Ba : Enlace):integer;
   VAR
      Estado,Cont : integer;
      Pa          : Enlace;
   BEGIN
   Cont:=0;
   while Ba <> nil do
      begin
      Estado:=Ba^.Funcion;
      if (Estado <> 0) then
         begin
         MUESTRA_ERROR (  'Error en el archivo '+Ba^.ident+' ',Estado);
         inc (Cont);
         end;
      Pa:=Ba;
      Ba:=Ba^.Sigue;
      dispose (Pa);
      end;
   CIERRAS:=Cont;
   RESETEA_BTRIEVE;
   END;


PROCEDURE DESPLIEGA_ERROR(texto : string ;
                          codigo: integer);
   VAR
      codigo_str : string[3];
      status_io  : integer;
   BEGIN
   str(codigo:3 , codigo_str);
   MENSAJE (' ERROR SISTEMA: ',
            texto + '. Codigo: ' + codigo_str, 21, falla_sistema);
   status_io := CIERRAS(Base);
   halt(codigo);
   END;
```

```
PROCEDURE ERROR_SISTEMA ( rutina,
                          status: string_alfa);
   BEGIN
   DESPLIEGA_ERROR ( 'Rutina: ' + rutina + ' Status: ' + status, 1);
  END;


FUNCTION REDONDEA (eq_monto  : extended;
                   decimales : byte ) : extended;
   VAR
      fraccion  : extended;
      base_10   : extended;
      indice    : byte;

   BEGIN
   base_10 := 1;
   if decimales > 0 then
      for indice := 1 to decimales do base_10 := base_10 * 10;
   fraccion := frac(eq_monto) * base_10;
   if (fraccion>2147483647) or (fraccion<-2147483647) then
      DESPLIEGA_ERROR('RUTINA REDONDEA: FRACCION RUNTIME ',207);

   fraccion := round(fraccion) / base_10;
   REDONDEA := int(eq_monto) + fraccion;
   END; { REDONDEA }

   PROCEDURE DA_FECHA_Y_HORA_MSDOS (var fecha_msdos : tpo_fecha;
                                    var hora_msdos  : tpo_hora);
      VAR
         fecha_msdos_str  : DateString;
         hora_msdos_str   : DateString;
      BEGIN
      fecha_msdos_str := TodayString('yyyymmdd');
      move(fecha_msdos_str[1],fecha_msdos[1],8);
      hora_msdos_str := CurrentTimeString('hhmm:ss');
      move(hora_msdos_str[1], hora_msdos[1], 4);
      END;        { DA_FECHA_Y_HORA_MSDOS }


   PROCEDURE EJECUTA (programa      : string;
                      comandos      : string;
                      nombre        : string;
                      proceso       : longint;
                      descarga_prog : boolean;
                      usa_command   : boolean);

      VAR
       status_io     : integer;
       letrero_error : string;
       stat_aux      : integer;

      BEGIN
         if trim(nombre) <> '' then
            MENSAJE('ESPERAR','REALIZANDO : '+ nombre,21,aviso);

         if descarga_prog then
            status_io := chain4 (programa, comandos)
         else
            status_io := execdos (programa+' '+comandos, usa_command,
                                  nil);
         stat_aux := DosExitCode;
         if (status_io <> ok_io) then
            begin
            case status_io of
                2: letrero_error := 'ARCHIVO NO ENCONTRADO';
                3: letrero_error := 'PATH NO ENCONTRADO';
                5: letrero_error := 'ACCESO NO PERMITIDO';
                6: letrero_error := 'HANDLE INVALIDO';
                8: letrero_error := 'MEMORIA INSUFICIENTE';
               10: letrero_error := 'MEDIO AMBIENTE INVALIDO';
               11: letrero_error := 'FORMATO INVALIDO';
               18: letrero_error := 'NO MAS ARCHIVOS';
               else
                   letrero_error := 'NO TERMINO BIEN EL PROCESO1';
               end; {case}
```

```
                DESPLIEGA_ERROR (letrero_error + ' ',status_io);
                end
              else {termino bien}
                begin
                if (proceso <> proceso_dummy) then
                  begin
                  ACTUALIZA_STATUS_PROCESO(proceso, realizado_ctr,
                                           status_io);
                  if status_io <> ok_io then
                    DESPLIEGA_ERROR('RUTINA EJECUTA:
                                     ACTUALIZA_STATUS_PROCESO ',status_io);
                  end;
                end;
            END;    {end procedure ejecuta}
END.
```

Programa con funciones generales para impresión:

```
{$I-,V-,R-,N+,E+}
UNIT UIMPRESO;
  { Unidad que contiene las rutinas para controlar la impresora }

INTERFACE
  USES UDCLGIRO;

  CONST
  Tamanio_15_cpp             = 15;
  Tamanio_10_cpp             = 10;
  Laser_carta_16_cpp         = 16;
  laser_carta_hor_16_cpp     = 17;
  laser_carta_hor_16_cpp_sm  = 18; { sin_margen }
  laser_vertical_10_cpp      = 9;
  laser_vertical_16_cpp      = 11;
  Ren1='
';
  Ren2='
';
  laser_red    = 'LR';
  laser_local  = 'LL';
  ati_red      = 'AR';
  ati_local    = 'AL';
  TInt='###,###,###,###,###,###.';

  VAR
  TamanioRenglon : integer;
  TamanioHoja    : integer;
  Titulo1,
  Titulo2,
  Titulo3,
  Titulo4          : string;
  control_imprime : string;
  ContRenglon,
  Pagina          : integer;
  Renglon         : string;

  FUNCTION CHECA_PRT : boolean;

  PROCEDURE PONE_LOCAL;

  PROCEDURE ACTUALIZA_STATUS_IMPRESORA(imp : tpo_impresora);

  PROCEDURE LEE_STATUS_IMPRESORA(var imp : tpo_impresora);

  PROCEDURE ELIGE_IMPRESORA;

  PROCEDURE CONTROL_DE_IMPRESORAS (comando_control: byte;
                                   P              : integer;
                                   C              : integer;
                                   T              : integer);

  PROCEDURE ENCABEZADO;

  PROCEDURE ENCABEZADO_MISMA_HOJA;

  PROCEDURE OTRA_HOJA;

  PROCEDURE SALTA_HOJA;

  PROCEDURE OTRA_HOJA_CON_ESPERA;

  PROCEDURE SALTA_HOJA_CON_ESPERA;

  PROCEDURE IMPRIME_RENGLON;

  PROCEDURE FECHA_REPORTES (     fecha     : tpo_fecha ;
                            var fecha_rep : tpo_fecha_reporte);
```

```
IMPLEMENTATION

   USES PRINTER,
        DOS,
        TPDOS,
        TPSTRING,
        UGRALGIR,
        UMSJGIRO,
        TPCRT,
        TPMENU;

   VAR status_io : integer;

   {-------------------------------------------------------------------------
-----}
   FUNCTION CHECA_PRT : boolean;
         { Regresa el valor boleano que indica el estado de la impresora   }
         { TRUE    --> Impresora Lista                                     }
         { FALSE   --> Impresora con problemas                             }
       CONST
          int17 = $17;
       VAR
          regs       : registers;

       BEGIN
       CHECA_PRT := false;
       regs.ah := 02; regs.dx := 0;
       intr (int17,regs);
       if regs.ah = 144 then CHECA_PRT := true;
       END;


     PROCEDURE PONE_LOCAL;

        BEGIN
        status_io := execdos('f:\public\endcap.exe', false, nil);
        if status_io <> ok_io then
           DESPLIEGA_ERROR('RUTINA PONE_LOCAL ',status_io);
        END;


     PROCEDURE ACTUALIZA_STATUS_IMPRESORA(imp : tpo_impresora);
        VAR
        imp_txt   : text;
        status_io : integer;

        BEGIN
        assign(imp_txt,arch_control_impresora);
        rewrite(imp_txt);
        status_io := IOresult;
        if status_io <> 0 then
           DESPLIEGA_ERROR('RUTINA ACTUALIZA STAT IMPRESORA
                            (rewrite)',status_io);
        write(imp_txt,imp);
        status_io := Ioresult;
        if status_io <> ok_io then
           DESPLIEGA_ERROR('RUTINA ACTUALIZA STAT IMPRESORA
                            (write)',status_io);

        close(imp_txt);
        status_io := Ioresult;
        if status_io <> ok_io then
           DESPLIEGA_ERROR('RUTINA ACTUALIZA STAT IMPRESORA
                            (close)',status_io);

        XYwrite(20,22,' ');
        if imp = laser_red then
           begin
           status_io := execdos('f:\public\stoi_lr.bat', true, nil);
           if status_io <> ok_io then
              DESPLIEGA_ERROR('RUTINA ACTUALIZA STAT IMPRESORA',status_io);
           end
        else
           if imp = laser_local then
              begin
```

```
            PONE_LOCAL;
            end
         else
            if imp = ati_red then
               begin
               status_io := execdos('f:\public\stoi_ar.bat', true, nil);
               if status_io <> ok_io then
                  DESPLIEGA_ERROR('RUTINA ACTUALIZA STAT
                                   IMPRESORA',status_io);
               end
            else
               if imp = ati_local then
                  begin
                  PONE_LOCAL;
                  end
               else
                  DESPLIEGA_ERROR('RUTINA ACTUALIZA STAT IMPR;  PARAMETRO
                                   DESCONOCIDO ',4);
   END;


PROCEDURE LEE_STATUS_IMPRESORA(var imp : tpo_impresora);

   VAR
   imp_txt   : text;
   status_io : integer;

   BEGIN
   assign(imp_txt,arch_control_impresora);
   reset(imp_txt);
   status_io := IOresult;

   if status_io = 2 then
      begin
      rewrite(imp_txt);
      status_io := IOresult;
      if status_io <> 0 then
         DESPLIEGA_ERROR('RUTINA LEE STATUS IMP (IOresult
                          rewrite)',status_io);
      write(imp_txt,ati_red);
      imp := ati_red;
      end
   else
      if status_io = 0 then
         readln(imp_txt,imp)
      else
         DESPLIEGA_ERROR('RUTINA LEE STATUS IMP (IOresult
                          reset)',status_io);
   END;


PROCEDURE ELIGE_IMPRESORA;
{$I f:\giros\fuentes\incluye\ctr_impr.pas}

VAR
caracter_salida      : char;
menu_ctr_impr        : menu;
key_ctr_impr         : menukey;
stack_menu_ctr_impr  : menustackp;

BEGIN
ENCABEZADO_PANTALLA('ELECCION DE IMPRESORA');
MENU_CTR_IMPRESORA (menu_ctr_impr);
repeat
   key_ctr_impr := MenuChoice(menu_ctr_impr,caracter_salida);
   EraseMenuOntoStack(menu_ctr_impr,stack_menu_ctr_impr);

   if caracter_salida <> #27 then
      case key_ctr_impr of
         1000: ACTUALIZA_STATUS_IMPRESORA(laser_local);

         2000: ACTUALIZA_STATUS_IMPRESORA(laser_red);

         3000: ACTUALIZA_STATUS_IMPRESORA(ati_local);
```

```
            4000: ACTUALIZA_STATUS_IMPRESORA(ati_red);

            end;

      ENCABEZADO_PANTALLA('ELECCION DE IMPRESORA');
      DrawMenuFromStack(menu_ctr_impr,stack_menu_ctr_impr);
   until (caracter_salida = #27);
   ENCABEZADO_PANTALLA('MENU PRINCIPAL');
   END;


   PROCEDURE CONTROL_DE_IMPRESORAS (comando_control: byte;
                                    P              : integer;
                                    C              : integer;
                                    T              : integer);

{ PROPOSITO : Da los comandos de control a la impresora para que opere
              en la forma que se le diga. }
CONST       { Caracteres ASCII }
   c_0   = #0  ;        c_1   = #1  ;
   c_15  = #15 ;        c_18  = #18 ;
   c_esc = #27 ;        c_33  = #33 ;
   c_48  = #48 ;        c_64  = #64 ;

VAR
   impresora       : tpo_impresora;

BEGIN  { CONTROL_DE_IMPRESORA }
LEE_STATUS_IMPRESORA(impresora);
Pagina:=P;
ContRenglon:=C;
TamanioRenglon:=T;
Renglon:=Ren1+Ren2;
control_imprime := '';

case comando_control of
     tamanio_15_cpp  : begin
                       if (impresora = ati_red) or
                          (impresora = ati_local) then
                          begin
                          control_imprime := c_15;
                          TamanioHoja:=88;
                          end
                       else
                          begin
                          control_imprime := #27 + 'E'          + {reset}
                                             #27 + '&l3A'       + {long
                                                                   pag 84}
                                             #27 + '(10U'       + {IBM-
                                                                   pc(usa)}
                                             #27 + '&l1O'       +
                                                          {landscape}
                                             #27 + '(s16.66H'   + {16.66
                                                                   cpp}
                                             #27 + '&l8D'       + {8 lpp}
                                             #27 + '9'          + {borra
                                                                   margenes}
                                             #27 + '&a14L'      ; {margen
                                                                   izq 14}
                          TamanioHoja := 131;
                          end;
                       end;

     tamanio_10_cpp  : begin
                       if (impresora = ati_red) or
                          (impresora = ati_local) then
                          begin
                          control_imprime := c_18;
                          TamanioHoja:=66;
                          end
                       else
                          begin
                          control_imprime := #27 + 'E'          + {reset}
```

```
                                        #27 + '&l3A'      + {long
                                                            pag 84}
                                        #27 + '(10U'      + {IBM-
                                                            pc(usa)}
                                        #27 + '&l1O'      +
                                                {landscape}
                                        #27 + '(s10H'     + {10 cpp}
                                        #27 + '&l8D'      + {8 lpp}
                                        #27 + '9'         + {borra
                                                            margenes}
                                        #27 + '&a9L'      ; {margen
                                                            izq 9}
                        TamanioHoja := 88;
                        end;
                    end;

laser_vertical_10_cpp : begin
                        if (impresora = ati_red) or
                           (impresora = ati_local) then
                           begin
                           control_imprime := c_18;
                           TamanioHoja:=66;
                           end
                        else
                           begin
                           control_imprime := #27 + 'E'         +
                                                        {reset}
                                              #27 + '&l3A'      +
                                                  {long pag 84}
                                              #27 + '(10U'      +
                                                  {IBM-pc(usa)}
                                              #27 + '&l0O'      +
                                            {portrait vertical}
                                              #27 + '(s10H'     +
                                                        {10 cpp}
                                              #27 + '&l8D'      + {8
                                                            lpp}
                                              #27 + '9'         +
                                                  {borra margenes}
                                              #27 + '&a5L'      ;
                                                  {margen izq 5}
                           TamanioHoja := 88;
                           end;
                        end;

laser_vertical_16_cpp : begin
                        if (impresora = ati_red) or
                           (impresora = ati_local) then
                           begin
                           control_imprime := c_15;
                           TamanioHoja:=66;
                           end
                        else
                           begin
                           control_imprime := #27 + 'E'         +
                                                        {reset}
                                              #27 + '&l3A'      +
                                                  {long pag 84}
                                              #27 + '(10U'      +
                                                  {IBM-pc(usa)}
                                              #27 + '&l0O'      +
                                            {portrait vertical}
                                              #27 + '(s16.66H'  +
                                                  {16.66 cpp}
                                              #27 + '&l8D'      + {8
                                                            lpp}
                                              #27 + '9'         +
                                                  {borra margenes}
                                              #27 + '&a13L'     ;
                                                  {margen izq 13}
                           TamanioHoja := 88;
                           end;
                        end;
```

```
laser_carta_16_cpp : begin
                     if (impresora = laser_local) or
                        (impresora = laser_red) then
                        control_imprime := #27 + 'E'          + {reset}
                                           #27 + '&l2A'       + {letter}
                                           #27 + '(10U'       + {IBM-
                                                                pc(usa)}
                                           #27 + '&l0O'       +
                                             {portrait vertical}
                                           #27 + '(s16.66H' + {16.66
                                                                cpp}
                                           #27 + '&l8D'       + {8 lpp}
                                           #27 + '9'          + {borra
                                                                margenes}
                                           #27 + '&a5L'       + {margen
                                                                izq 5}
                                           #27 + '&l8E'         {margen
                                                                top 8}
                     else
                     control_imprime := c_15;
                     end;

laser_carta_hor_16_cpp : begin
                     if (impresora = laser_local) or
                        (impresora = laser_red) then
                        control_imprime := #27 + 'E'          + {reset}
                                           #27 + '&l2A'       +
                                                              {letter}
                                           #27 + '(10U'       + {IBM-
                                                                pc(usa)}
                                           #27 + '&l1O'       +
                                                 {landscape}
                                           #27 + '(s16.66H' + {16.66
                                                                cpp}
                                           #27 + '&l8D'       + {8 lpp}
                                           #27 + '9'          + {borra
                                                                margenes}
                                           #27 + '&a17L'      + {margen
                                                                izq 17}
                                           #27 + '&8E'          {margen
                                                                top 8}
                     else
                     control_imprime := c_15;
                     end;

laser_carta_hor_16_cpp_sm : begin
                     if (impresora = laser_local) or
                        (impresora = laser_red) then
                        control_imprime := #27 + 'E'          + {reset}
                                           #27 + '&l2A'       +{letter}
                                           #27 + '(10U'       + {IBM-
                                                                pc(usa)}
                                           #27 + '&l1O'       +
                                                 {landscape}
                                           #27 + '(s16.67H' + {16.66
                                                   cpp}
                                           #27 + '&l8D'        +{8 lpp}
                                           #27 + '9'          + {borra
                                                                margenes}
                                           #27 + '&a0L'       + {margen
                                                                izq 17}
                                           #27 + '&2E'          {margen
                                                                top 8}
                     else
                     control_imprime := c_15;
                     end;
   end;  { case }

while (not CHECA_PRT) do
   MENSAJE ('AVISO','No esta lista la impresora.Favor de revisarla', 15,
            aviso);

writeln(lst, control_imprime)
END ; { CONTROL_DE_IMPRESORAS }
```

```
PROCEDURE ENCABEZADO;

VAR
SPagina     : string[3];
Renglon_Tit : String;
Con_Tam_Ren : byte absolute Renglon;
Con_Tam_Enc : byte absolute Renglon_Tit;
Inicio,
a           : integer;
BdeM,
Hoja,
Fecha,
THora,
stoi        : string[15];
Hora,
Minuto,
Segundo,
CSegundo,
Anio,
Mes,
Dia,
DiaS        : word;
Sanio       : string[4];
SHora,
SMinuto,
SSegundo,
Smes,
SDia        : string[2];

BEGIN
inicio:=0;
Renglon_Tit:=Ren1+Ren2;
Con_Tam_Ren:=TamanioRenglon;
Con_Tam_Enc:=TamanioRenglon;
BdeM:='';
SPagina:= form ('###',Pagina);
Hoja :='HOJA:  ';
Fecha:='FECHA:    /  /  ';
THora:='HORA:     :  :  ';
Stoi:='STOI  ';
move (BdeM[1],Renglon_Tit[1],15);
move (Hoja[1],Renglon_Tit[(TamanioRenglon-14)],6);
move (SPagina[1],Renglon_Tit[(TamanioRenglon-7)],3);
inicio:=length(titulo1);
inicio:=round(TamanioRenglon/2)-round(length(Titulo1)/2);
move (Titulo1[1],Renglon_Tit[Inicio],length(Titulo1));
writeln(lst,Renglon_Tit);
Renglon_Tit:=Ren1+Ren2;
Con_Tam_Enc:=TamanioRenglon;
move (stoi[1],Renglon_Tit[1],6);
move (Fecha[1],Renglon_Tit[TamanioRenglon-14],15);
getdate (Anio,Mes,Dia,DiaS);
Sanio:=form ('####',anio);
SMes:= form ('@#',mes);
SDia:= form ('@#',dia);
move (Sanio[3],Renglon_Tit[tamanioRenglon-1],2);
move (Smes[1],Renglon_Tit[tamanioRenglon-4],2);
move (Sdia[1],Renglon_Tit[tamanioRenglon-7],2);
inicio:=round(TamanioRenglon/2)-round(length(Titulo2)/2);
move (Titulo2[1],Renglon_Tit[Inicio],length(Titulo2));
writeln(lst,Renglon_Tit);
Renglon_Tit:=Ren1+Ren2;
Con_Tam_Enc:=TamanioRenglon;
move (THora[1],Renglon_Tit[TamanioRenglon-14],15);
gettime (Hora,Minuto,Segundo,CSegundo);
SHora:=form ('##',Hora);
Sminuto:= form ('@#',minuto);
SSegundo:= form ('@#',Segundo);
move (SHora[1],Renglon_Tit[tamanioRenglon-7],2);
move (Sminuto[1],Renglon_Tit[tamanioRenglon-4],2);
move (SSegundo[1],Renglon_Tit[tamanioRenglon-1],2);
inicio:=round(TamanioRenglon/2)-round(length(Titulo3)/2);
move (Titulo3[1],Renglon_Tit[Inicio],length(Titulo3));
writeln(lst,Renglon_Tit);
```

```
for a:=1 to TamanioRenglon do
   Renglon_Tit[a]:='-';
Con_Tam_Enc:=TamanioRenglon;
writeln(lst,Renglon_Tit);
Renglon_Tit:=Ren1+Ren2;
Con_Tam_Enc:=TamanioRenglon;
move (Titulo4[1],Renglon_Tit[1],length(Titulo4));
writeln(lst,Renglon_Tit);
ContRenglon:=5;
END;

PROCEDURE ENCABEZADO_MISMA_HOJA;

VAR
SPagina     : string[3];
Renglon_Tit : String;
Con_Tam_Ren : byte absolute Renglon;
Con_Tam_Enc : byte absolute Renglon_Tit;
Inicio,
a           : integer;
BdeM,
Hoja,
Fecha,
THora,
stoi        : string[15];
Hora,
Minuto,
Segundo,
CSegundo,
Anio,
Mes,
Dia,
DiaS        : word;
Sanio       : string[4];
SHora,
SMinuto,
SSegundo,
Smes,
SDia        : string[2];

BEGIN
inicio:=0;
Renglon_Tit:=Ren1+Ren2;
Con_Tam_Ren:=TamanioRenglon;
Con_Tam_Enc:=TamanioRenglon;
BdeM:='';
SPagina:= form ('###',Pagina);
Hoja :='HOJA:  ';
Fecha:='FECHA:    /  /  ';
THora:='HORA:     :  :  ';
Stoi:='STOI  ';
move (BdeM[1],Renglon_Tit[1],15);
move (Hoja[1],Renglon_Tit[(TamanioRenglon-14)],6);
move (SPagina[1],Renglon_Tit[(TamanioRenglon-7)],3);
inicio:=length(titulo1);
inicio:=round(TamanioRenglon/2)-round(length(Titulo1)/2);
move (Titulo1[1],Renglon_Tit[Inicio],length(Titulo1));
writeln(lst,Renglon_Tit);
Renglon_Tit:=Ren1+Ren2;
Con_Tam_Enc:=TamanioRenglon;
move (stoi[1],Renglon_Tit[1],6);
move (Fecha[1],Renglon_Tit[TamanioRenglon-14],15);
getdate (Anio,Mes,Dia,DiaS);
Sanio:=form ('####',anio);
SMes:= form ('@#',mes);
SDia:= form ('@#',dia);
move (Sanio[3],Renglon_Tit[tamanioRenglon-1],2);
move (Smes[1],Renglon_Tit[tamanioRenglon-4],2);
move (Sdia[1],Renglon_Tit[tamanioRenglon-7],2);
inicio:=round(TamanioRenglon/2)-round(length(Titulo2)/2);
move (Titulo2[1],Renglon_Tit[Inicio],length(Titulo2));
writeln(lst,Renglon_Tit);
Renglon_Tit:=Ren1+Ren2;
Con_Tam_Enc:=TamanioRenglon;
move (THora[1],Renglon_Tit[TamanioRenglon-14],15);
```

```
gettime (Hora,Minuto,Segundo,CSegundo);
SHora:=form ('##',Hora);
Sminuto:= form ('@#',minuto);
SSegundo:= form ('@#',Segundo);
move (SHora[1],Renglon_Tit[tamanioRenglon-7],2);
move (Sminuto[1],Renglon_Tit[tamanioRenglon-4],2);
move (SSegundo[1],Renglon_Tit[tamanioRenglon-1],2);
inicio:=round(TamanioRenglon/2)-round(length(Titulo3)/2);
move (Titulo3[1],Renglon_Tit[Inicio],length(Titulo3));
writeln(lst,Renglon_Tit);
for a:=1 to TamanioRenglon do
   Renglon_Tit[a]:='-';
Con_Tam_Enc:=TamanioRenglon;
writeln(lst,Renglon_Tit);
Renglon_Tit:=Ren1+Ren2;
Con_Tam_Enc:=TamanioRenglon;
move (Titulo4[1],Renglon_Tit[1],length(Titulo4));
writeln(lst,Renglon_Tit);
writeln(lst);
END;


PROCEDURE OTRA_HOJA;

BEGIN
Pagina:=Pagina+1;
write (lst,chr(12));{RMA hector no le cambies por favor}
ContRenglon := 1;
END;

PROCEDURE SALTA_HOJA;

BEGIN
Pagina:=Pagina+1;
write (lst,chr(12));
ENCABEZADO;
END;

PROCEDURE OTRA_HOJA_CON_ESPERA;

BEGIN
Pagina:=Pagina+1;
write (lst,chr(12));
delay(1000);
ContRenglon := 1;
END;

PROCEDURE SALTA_HOJA_CON_ESPERA;

BEGIN
Pagina:=Pagina+1;
write (lst,chr(12));
delay(1000);
ENCABEZADO;
END;


PROCEDURE IMPRIME_RENGLON;

VAR
Con_Tam_Ren : byte absolute Renglon;

BEGIN
if ContRenglon>= (TamanioHoja-5) then
   begin
   Pagina:=Pagina+1;
   ENCABEZADO;
   end;
writeln(lst,Renglon);
Renglon:=Ren1+Ren2;
Con_Tam_Ren:=TamanioRenglon;
ContRenglon:=ContRenglon+1;
END;

PROCEDURE FECHA_REPORTES (      fecha      : tpo_fecha ;
```

```
                                var fecha_rep : tpo_fecha_reporte);
   VAR
   aux : string[2];
   mes : string[3];

   BEGIN
   fecha_rep := '          ';
   aux := '  ';
   mes := '   ';
   move(fecha[3],fecha_rep[8],2);
   move(fecha[7],fecha_rep[1],2);
   move(fecha[5],aux[1],2);
   if aux[1] = '0' then
      begin
      case aux[2] of
      '1': mes := 'ENE';
      '2': mes := 'FEB';
      '3': mes := 'MAR';
      '4': mes := 'ABR';
      '5': mes := 'MAY';
      '6': mes := 'JUN';
      '7': mes := 'JUL';
      '8': mes := 'AGO';
      '9': mes := 'SEP';
      end
      end
   else
      begin
      case aux[2] of
      '0': mes := 'OCT';
      '1': mes := 'NOV';
      '2': mes := 'DIC';
      end
      end;
   move(mes[1],fecha_rep[4],3);
   END;


END.
```

Programa con funciones generales para impresión:

```
{$I-,V-,R-,N+,E+}
UNIT UIMPRIME;

{  Unidad que contiene las rutinas necesarias para el respaldo  y la
   reimpresión de reportes. }


INTERFACE

   USES
      UDCLGIRO;


   CONST

         {Tipo Impresora}
      laser_red    = 'LR';
      laser_local = 'LL';
      ati_red      = 'AR';
      ati_local    = 'AL';

         {Margenes default}
      margen_superior  = '8';
      margen_izquierdo = '5';
      margen_nulo      = '';
         {Tipo caracter, cpp o lpp}
      cpp_16           = '16.66';
      cpp_10           = '10';
      lpp_8            = '8';
         {Tipo hoja}
      carta            = '2A';
      oficio           = '3A';
         {Orientación}
      o_horizontal     = '1O';
      o_vertical       = '0O';

         { Caracteres ASCII }
      c_15  = #27#33#5 ;
      c_18  = #18 ;
      c_esc = #27 ;
      c_esc_laser = #027;
   VAR
      Titulo1,
      Titulo2,
      Titulo3,
      Titulo4           : string;
      TamanioRenglon    : integer;
      ContRenglon       : integer;
      renglones_x_hoja : integer;
      pagina            : integer;


   PROCEDURE ELIGE_IMPRESORA;

   PROCEDURE CONTROL_DE_IMPRESORAS (margen_sup      : string;
                                    margen_izq      : string;
                                    tipo_hoja       : string;
                                    orientacion     : string;
                                    cpp             : string;
                                    lpp             : string;
                                    tamanio_renglon : integer);
   PROCEDURE ENCABEZADO;

   PROCEDURE ENCABEZADO_MISMA_PAGINA;

   PROCEDURE SALTA_HOJA;

   PROCEDURE OTRA_HOJA;

   PROCEDURE OTRA_HOJA_CON_ESPERA;
```

```
    PROCEDURE INICIA_REPORTE (nombre_archivo : string);

    PROCEDURE IMPRIME_LINEA    (    salto_previo   : integer;
                                    linea_rep      : string);

    PROCEDURE TERMINA_REPORTE;

    PROCEDURE FECHA_REPORTES (     fecha      : tpo_fecha ;
                               var fecha_rep  : tpo_fecha_reporte);

    PROCEDURE LEE_STATUS_IMPRESORA (var imp : tpo_impresora);

    PROCEDURE ACTUALIZA_STATUS_IMPRESORA (imp : tpo_impresora);

IMPLEMENTATION

    USES
        UGRALGIR,
        UMSJGIRO,
        PRINTER,
        DOS,
        TPCRT,
        TPMENU,
        TPDOS,
        TPSTRING;


    VAR
       f              : text;
       status_io      : integer;
       rayas          : array[1..250] of char;

    FUNCTION CHECA_PRT : boolean;

          { Regresa el valor boleano que indica el estado de la impresora  }
          { TRUE    --> Impresora Lista                                    }
          { FALSE   --> Impresora con problemas                            }

        CONST
           int17 = $17;
        VAR
           regs      : registers;

        BEGIN
        CHECA_PRT := false;
        regs.ah := 02;
        regs.dx := 0;
        intr (int17,regs);
        if (regs.ah = 144) then CHECA_PRT := true;
        END;


    PROCEDURE PONE_LOCAL;

         BEGIN
         status_io := execdos('f:\public\endcap.exe', false, nil);
         if status_io <> ok_io then
            DESPLIEGA_ERROR('RUTINA PONE_LOCAL ',status_io);
         END;


    PROCEDURE ACTUALIZA_STATUS_IMPRESORA (imp : tpo_impresora);

         VAR
            imp_txt    : text;
            status_io : integer;

         BEGIN
         assign(imp_txt,arch_control_impresora);
         rewrite(imp_txt);
         status_io := IOresult;
         if (status_io <> 0) then
            DESPLIEGA_ERROR('actualiza_stat_impresora (rewrite)',status_io);
         write(imp_txt,imp);
```

```
      status_io := Ioresult;
      if (status_io <> ok_io) then
         DESPLIEGA_ERROR('actualiza_stat_impresora (write)',status_io);

      close(imp_txt);
      status_io := Ioresult;
      if (status_io <> ok_io) then
         DESPLIEGA_ERROR('actualiza_stat_impresora (close)',status_io);

      XYwrite(20,22,' ');
      if (imp = laser_red) then
         begin
         status_io := execdos('f:\public\stoi_lr.bat', true, nil);
         if (status_io <> ok_io) then
            DESPLIEGA_ERROR('actualiza_stat_impresora',status_io);
         end
      else
         if (imp = laser_local) then PONE_LOCAL
         else
         if (imp = ati_red) then
            begin
            status_io := execdos('f:\public\stoi_ar.bat', true, nil);
            if (status_io <> ok_io) then
               DESPLIEGA_ERROR('actualiza_stat_impresora',status_io);
            end
         else
         if (imp = ati_local) then PONE_LOCAL
         else
            DESPLIEGA_ERROR('actualiza_stat_impresora: param unknown',4);
   END; {ACTUALIZA_STATUS_IMPRESORA}


PROCEDURE LEE_STATUS_IMPRESORA (var imp : tpo_impresora);

   VAR
      imp_txt   : text;
      status_io : integer;

   BEGIN
   assign(imp_txt,arch_control_impresora);
   reset(imp_txt);
   status_io := IOresult;
   if (status_io = 2) then
      begin
      rewrite(imp_txt);
      status_io := IOresult;
      if (status_io <> 0) then
         DESPLIEGA_ERROR('lee_status_impresora.rewrite',status_io);
      write(imp_txt,ati_red);
      imp := ati_red;
      end
   else
      if (status_io = 0) then readln(imp_txt,imp)
      else
         DESPLIEGA_ERROR('lee_status_impresora.reset',status_io);
   END; {LEE_STATUS_IMPRESORA}


PROCEDURE ELIGE_IMPRESORA;

   PROCEDURE MENU_CTR_IMPRESORA (var M : Menu);
      CONST
        Color1 : MenuColorArray = ($79,$79,$7F,$1F,$74,$1E,$7F,$00);
        Frame1 : FrameArray = '++++-¦';

      BEGIN
        {Customize this call for special exit characters and
         custom item displays}
      M := NewMenu([], nil);
      SubMenu(29,9,ScreenHeight,Vertical,Frame1,Color1,'');
        MenuMode(True, True, False);
        MenuItem('   LASER LOCAL ',1,4,1000,'');
        MenuItem('   LASER RED',2,6,2000,'');
```

```
        MenuItem('   ATI LOCAL',3,4,3000,'');
        MenuItem('   ATI RED',4,5,4000,'');
        PopSublevel;

      ResetMenu(M);
      SetMenuDelay(M, 20);
      END; {MENU_CTR_IMPRESORA}


   VAR
      caracter_salida      : char;
      menu_ctr_impr        : menu;
      key_ctr_impr         : menukey;
      stack_menu_ctr_impr : menustackp;

   BEGIN
   ENCABEZADO_PANTALLA('ELECCION DE IMPRESORA');
   MENU_CTR_IMPRESORA (menu_ctr_impr);
   repeat
      key_ctr_impr := MenuChoice(menu_ctr_impr,caracter_salida);
      EraseMenuOntoStack(menu_ctr_impr,stack_menu_ctr_impr);

      if (caracter_salida <> c_esc) then
         case key_ctr_impr of
            1000: ACTUALIZA_STATUS_IMPRESORA(laser_local);

            2000: ACTUALIZA_STATUS_IMPRESORA(laser_red);

            3000: ACTUALIZA_STATUS_IMPRESORA(ati_local);

            4000: ACTUALIZA_STATUS_IMPRESORA(ati_red);

            end;

      ENCABEZADO_PANTALLA('ELECCION DE IMPRESORA');
      DrawMenuFromStack(menu_ctr_impr,stack_menu_ctr_impr);
   until (caracter_salida = c_esc);
   ENCABEZADO_PANTALLA('MENU PRINCIPAL');
   END; {ELIGE_IMPRESORA}


PROCEDURE GUARDA_LINEA (    salto_previo : integer;
                            linea        : string);
   VAR
      salto_previo_s : string[2];
      stat_aux       : integer;

   BEGIN
   str(salto_previo,salto_previo_s);
   if (salto_previo < 10) then
      salto_previo_s := '0' + salto_previo_s;
   writeln (f,concat(salto_previo_s,linea));
   stat_aux := IOResult;
   if (stat_aux = 160) then
      begin
      while (not CHECA_PRT) do
            MENSAJE ('AVISO',
                     'No esta lista la impresora.Favor de revisarla',
                     15,
                     aviso);
      end
   else
   if (stat_aux <> 0) then
      DESPLIEGA_ERROR ('guarda_linea:',stat_aux);
   END;


PROCEDURE CONTROL_DE_IMPRESORAS (margen_sup      : string;
                                 margen_izq      : string;
                                 tipo_hoja       : string;
                                 orientacion     : string;
                                 cpp             : string;
                                 lpp             : string;
```

```
                                        tamanio_renglon : integer);

  { Da los comandos de control a la impresora para que opere
    en la forma que se le diga. }


VAR
   impresora       : tpo_impresora;
   control_imprime : string;

BEGIN
LEE_STATUS_IMPRESORA (impresora);
pagina          := 1;
ContRenglon     := 200;
TamanioRenglon  := tamanio_renglon;
control_imprime := '';

        {Determinar número de renglones_x_hoja}
if (impresora = laser_local) or (impresora = laser_red)
then
   begin       {laser's}
   if (lpp = lpp_8)
   then
      begin     {8 lpp}
      if (orientacion = o_vertical)
      then
         begin   {vertical}
         if (tipo_hoja = carta)
         then     {carta}
            renglones_x_hoja := 80
         else     {oficio}
            renglones_x_hoja := 96;
         end
      else       {horizontal}
         renglones_x_hoja := 60;
      end;
   end
else
   begin       {ati's}
   if (lpp = lpp_8)
   then
      begin     {8 lpp}
      if (cpp = cpp_16)
      then       {16.66 cpp}
         renglones_x_hoja := 88
      else       {10 cpp}
         renglones_x_hoja := 66;
      end;
   end;

        {Determinar los caracteres de control de la impresora}
if (impresora = laser_local) or (impresora = laser_red)
then
    begin       {laser's}
    control_imprime := c_esc_laser + 'E'                          +
                                                                  {reset}
                       c_esc_laser + '&l' + tipo_hoja             +
                       c_esc_laser + '(10U'                       +  {IBM-
                                                                  pc(usa)}
                       c_esc_laser + '&l' + orientacion           +
                       c_esc_laser + '(s' + cpp + 'H'             +
                       c_esc_laser + '&l' + lpp + 'D'             +
                       c_esc_laser + '9'                          +
                                                    {Borra margenes}
                       c_esc_laser + '&a' + margen_izq + 'L';
    if (margen_sup <> margen_nulo) then
       control_imprime := control_imprime +
                          c_esc + '&l' + margen_sup + 'E';
    end
else
   begin        {ati's}
   if (cpp = cpp_16)
   then          {16.66 cpp}
      control_imprime := c_15
   else          {10 cpp}
```

```
        control_imprime := c_18 ;
    end;

  while (not CHECA_PRT) do
        MENSAJE ('AVISO',
                 'No esta lista la impresora.Favor de revisarla',
                 15,
                 aviso);

  writeln(lst, control_imprime);
  GUARDA_LINEA (0, control_imprime);
  END ; { CONTROL_DE_IMPRESORAS }


PROCEDURE ENCABEZADO;

  VAR
     Spagina      : string[3];
     renglon_tit  : String;
     inicio       : integer;
     a            : integer;
     BdeM,
     hoja,
     fecha,
     Thora,
     siegse           : string[15];
     hora,
     minuto,
     segundo,
     Csegundo,
     anio,
     mes,
     dia,
     diaS             : word;
     Sanio            : string[4];
     Shora,
     Sminuto,
     Ssegundo,
     Smes,
     Sdia             : string[2];
     salto_de_hoja    : string;

  BEGIN
  if (pagina > 0) then
     begin
     salto_de_hoja := chr(12);
     write (lst,salto_de_hoja);
     GUARDA_LINEA (0,salto_de_hoja);
     end;

  pagina := pagina+1;
  ContRenglon := 5;

  inicio      := 0;
  renglon_tit := copy(blancos,1,255);
  BdeM        := '';
  Spagina     := form ('###',pagina);
  hoja        := 'HOJA:  ';
  fecha       := 'FECHA:   /  /  ';
  Thora       := 'HORA:    :  :  ';
  siegse        := 'SISGIROS  ';
  move (BdeM[1],renglon_tit[1],15);
  move (hoja[1],renglon_tit[(TamanioRenglon-14)],6);
  move (Spagina[1],renglon_tit[(TamanioRenglon-7)],3);
  inicio      := length(titulo1);
  inicio      := round(TamanioRenglon/2)-round(length(Titulo1)/2);
  move (Titulo1[1],renglon_tit[inicio],length(Titulo1));
  writeln(lst,copy(renglon_tit,1,TamanioRenglon));
  GUARDA_LINEA (0,copy(renglon_tit,1,TamanioRenglon));

  renglon_tit := copy(blancos,1,255);
  move (siegse[1],renglon_tit[1],6);
  move (fecha[1],renglon_tit[TamanioRenglon-14],15);
  getdate (anio,mes,dia,diaS);
```

```
    Sanio       := form ('####',anio);
    Smes        := form ('@#',mes);
    Sdia        := form ('@#',dia);
    move (Sanio[3],renglon_tit[tamanioRenglon-11],2);
    move (Smes[1],renglon_tit[tamanioRenglon-4],2);
    move (Sdia[1],renglon_tit[tamanioRenglon-7],2);
    inicio := round(TamanioRenglon/2)-round(length(Titulo2)/2);
    move (Titulo2[1],renglon_tit[inicio],length(Titulo2));
    writeln(lst,copy(renglon_tit,1,TamanioRenglon));
    GUARDA_LINEA (0,copy(renglon_tit,1,TamanioRenglon));

    renglon_tit := copy(blancos,1,255);
    move (Thora[1],renglon_tit[TamanioRenglon-14],15);
    gettime (hora,minuto,segundo,Csegundo);
    Shora       := form ('##',hora);
    Sminuto     := form ('@#',minuto);
    Ssegundo    := form ('@#',segundo);
    move (Shora[1],renglon_tit[tamanioRenglon-7],2);
    move (Sminuto[1],renglon_tit[tamanioRenglon-4],2);
    move (Ssegundo[1],renglon_tit[tamanioRenglon-1],2);
    inicio      := round(TamanioRenglon/2)-round(length(Titulo3)/2);
    move (Titulo3[1],renglon_tit[inicio],length(Titulo3));
    writeln(lst,copy(renglon_tit,1,TamanioRenglon));
    GUARDA_LINEA (0,copy(renglon_tit,1,TamanioRenglon));

    renglon_tit := copy(rayas,1,TamanioRenglon);
    writeln(lst,copy(renglon_tit,1,TamanioRenglon));
    GUARDA_LINEA (0,copy(renglon_tit,1,TamanioRenglon));

    renglon_tit := copy(blancos,1,255);
    move (Titulo4[1],renglon_tit[1],length(Titulo4));
    writeln(lst,copy(renglon_tit,1,TamanioRenglon));
    GUARDA_LINEA (0,copy(renglon_tit,1,TamanioRenglon));
    END; {ENCABEZADO}

PROCEDURE ENCABEZADO_MISMA_PAGINA;

    VAR
       Spagina      : string[3];
       renglon_tit  : String;
       inicio       : integer;
       a            : integer;
       BdeM,
       hoja,
       fecha,
       Thora,
       siegse            : string[15];
       hora,
       minuto,
       segundo,
       Csegundo,
       anio,
       mes,
       dia,
       diaS              : word;
       Sanio             : string[4];
       Shora,
       Sminuto,
       Ssegundo,
       Smes,
       Sdia              : string[2];
       salto_de_hoja     : string;

    BEGIN
    inicio      := 0;
    renglon_tit := copy(blancos,1,255);
    BdeM        := '';
    Spagina     := form ('###',pagina);
    hoja        := 'HOJA:  ';
    fecha       := 'FECHA:    /  /  ';
    Thora       := 'HORA:     :  :  ';
    siegse        := 'SIGGIROS';
    move (BdeM[1],renglon_tit[1],15);
    move (hoja[1],renglon_tit[(TamanioRenglon-14)],6);
    move (Spagina[1],renglon_tit[(TamanioRenglon-7)],3);
```

```
   inicio      := length(titulo1);
   inicio      := round(TamanioRenglon/2)-round(length(Titulo1)/2);
   move (Titulo1[1],renglon_tit[inicio],length(Titulo1));
   writeln(lst,copy(renglon_tit,1,TamanioRenglon));
   GUARDA_LINEA (0,copy(renglon_tit,1,TamanioRenglon));

   renglon_tit := copy(blancos,1,255);
   move (slegse[1],renglon_tit[1],6);
   move (fecha[1],renglon_tit[TamanioRenglon-14],15);
   getdate (anio,mes,dia,diaS);
   Sanio       := form ('####',anio);
   Smes        := form ('@#',mes);
   Sdia        := form ('@#',dia);
   move (Sanio[3],renglon_tit[tamanioRenglon-1],2);
   move (Smes[1],renglon_tit[tamanioRenglon-4],2);
   move (Sdia[1],renglon_tit[tamanioRenglon-7],2);
   inicio := round(TamanioRenglon/2)-round(length(Titulo2)/2);
   move (Titulo2[1],renglon_tit[inicio],length(Titulo2));
   writeln(lst,copy(renglon_tit,1,TamanioRenglon));
   GUARDA_LINEA (0,copy(renglon_tit,1,TamanioRenglon));

   renglon_tit := copy(blancos,1,255);
   move (Thora[1],renglon_tit[TamanioRenglon-14],15);
   gettime (hora,minuto,segundo,Csegundo);
   Shora       := form ('##',hora);
   Sminuto     := form ('@#',minuto);
   Ssegundo    := form ('@#',segundo);
   move (Shora[1],renglon_tit[tamanioRenglon-7],2);
   move (Sminuto[1],renglon_tit[tamanioRenglon-4],2);
   move (Ssegundo[1],renglon_tit[tamanioRenglon-1],2);
   inicio      := round(TamanioRenglon/2)-round(length(Titulo3)/2);
   move (Titulo3[1],renglon_tit[inicio],length(Titulo3));
   writeln(lst,copy(renglon_tit,1,TamanioRenglon));
   GUARDA_LINEA (0,copy(renglon_tit,1,TamanioRenglon));

   renglon_tit := copy(rayas,1,TamanioRenglon);
   writeln(lst,copy(renglon_tit,1,TamanioRenglon));
   GUARDA_LINEA (0,copy(renglon_tit,1,TamanioRenglon));

   renglon_tit := copy(blancos,1,255);
   move (Titulo4[1],renglon_tit[1],length(Titulo4));
   writeln(lst,copy(renglon_tit,1,TamanioRenglon));
   GUARDA_LINEA (0,copy(renglon_tit,1,TamanioRenglon));
   END; {ENCABEZADO}


PROCEDURE SALTA_HOJA;

   BEGIN
   ContRenglon := 200;
   END;

PROCEDURE OTRA_HOJA;
   BEGIN
   write(lst,chr(12)); {EAL}
   ContRenglon := 200;
   END;


PROCEDURE OTRA_HOJA_CON_ESPERA;
   BEGIN
   write (lst,chr(12)); {EAL}
   delay(1000);
   ContRenglon := 1;
   END;


PROCEDURE FECHA_REPORTES (     fecha     : tpo_fecha ;
                           var fecha_rep : tpo_fecha_reporte);

   VAR
      aux : string[2];
      mes : string[3];

   BEGIN
```

```
      fecha_rep := '          ';
      aux       := '  ';
      mes       := '    ';
      move(fecha[3],fecha_rep[8],2);
      move(fecha[7],fecha_rep[1],2);
      move(fecha[5],aux[1],2);
      if (aux[1] = '0') then
         begin
         case aux[2] of
         '1': mes := 'ENE';
         '2': mes := 'FEB';
         '3': mes := 'MAR';
         '4': mes := 'ABR';
         '5': mes := 'MAY';
         '6': mes := 'JUN';
         '7': mes := 'JUL';
         '8': mes := 'AGO';
         '9': mes := 'SEP';
         end {case}
         end
      else
         begin
         case aux[2] of
         '0': mes := 'OCT';
         '1': mes := 'NOV';
         '2': mes := 'DIC';
         end {case}
         end;
      move(mes[1],fecha_rep[4],3);
      END; {FECHA_REPORTES}


   PROCEDURE INICIA_REPORTE (nombre_archivo : string);
{
               Prepara el archivo que servirá para la impresión:
                     1. Si ya existe lo borra.
                     2. Lo crea y abre para salida.
}
      VAR
         inf_archivo  : SearchRec;
         i            : integer;
         stat         : integer;

      BEGIN
      for i := 1 to sizeof(rayas) do
         rayas[i] := '-';
      assign(f,nombre_archivo);
      stat := IOResult;
      if (stat <> 0) then
         DESPLIEGA_ERROR ('inicia_reporte, '+nombre_archivo+'
                           ASSIGN:',stat);
      FindFirst (nombre_archivo,archive,inf_archivo);
      if (DosError = 0) then
         begin
         Erase (f);
         stat := IOResult;
         if (stat <> 0) then
            DESPLIEGA_ERROR ('inicia_reporte, '+nombre_archivo+'
                              ERASE:',stat);
         end
      else
         if (DosError = 2) then
            DESPLIEGA_ERROR ('inicia_reporte, PATH_INVALIDO',0);
      ReWrite (f);
      stat := IOResult;
      if (stat <> 0) then
         DESPLIEGA_ERROR ('inicia_reporte, REWRITE:',stat);
      END; {INICIA_REPORTE}


   PROCEDURE IMPRIME_LINEA   (   salto_previo   : integer;
                                 linea_rep      : string);
{
         Imprime y guarda la línea en turno en el archivo de impresión.
```

```
}
      VAR
         stat_aux,
         saltos          : integer;

      BEGIN
               {Imprime la linea}
      if ((ContRenglon+salto_previo) >= renglones_x_hoja)
      then
         ENCABEZADO
      else
         begin
         saltos := 0;
         while (saltos < salto_previo) do
            begin
            while (not CHECA_PRT) do
            MENSAJE ('AVISO',
                     'No esta lista la impresora.Favor de revisarla',
                     15,
                     aviso);
            writeln (lst);
            saltos := saltos+1;
            ContRenglon := ContRenglon+1;
            end;
         end;

      while (not CHECA_PRT) do
         MENSAJE ('AVISO','No esta lista la impresora.Favor de revisarla',
                  15,aviso);
      writeln(lst,linea_rep);

      GUARDA_LINEA (salto_previo, linea_rep);
      ContRenglon := ContRenglon+1;
      END; {IMPRIME_LINEA}


   PROCEDURE TERMINA_REPORTE;
{
            Cierra el archivo de respaldo para las impresiones.
}
      VAR
         stat : integer;

      BEGIN
{      writeln (lst, c_esc+'E');  resetea impresora}

      Close (f);
      stat := IOResult;
      if (stat <> 0) then
         DESPLIEGA_ERROR ('cierra_reporte, CLOSE:',stat);
      END;

END.
```

Programa con funciones para procesos de lectura de datos capturados:

```
{$I-,V-,R-,N+,E+,D+}
UNIT ULECTURA;
  { Unidad que contiene las rutinas generales del sistema }

INTERFACE
   USES UDCLGIRO;

   PROCEDURE LEE_FECHA(  columna,renglon : byte;
                         texto_escrito   : string ;
                     var fecha           : tpo_fecha);


   PROCEDURE LEE_TEXTO (  columna,renglon : byte;
                          texto_escrito   : string;
                      var texto_leer      : string;
                          visible         : boolean;
                      var press_ESC       : boolean);

   PROCEDURE LEE_NUMEROS(columna,renglon : byte;
                         texto_escrito   : string;
                     var texto_leer      : string);

IMPLEMENTATION
   USES
       UCONTROL,
       UMSJGIRO,
       UGRALGIR,
       TPCRT;


   PROCEDURE LEE_FECHA(  columna,renglon : byte;
                         texto_escrito   : string ;
                     var fecha           : tpo_fecha);

   VAR
      str_lee : string[10];
      c:char;
      i,aux,
      status_io : integer;
      indica_salto : boolean;
      whoy      : tpo_fecha;

   BEGIN
   xywrite(columna,renglon,texto_escrito);
   FECHA_HOY(whoy, status_io);
   if status_io <> ok_io then
      DESPLIEGA_ERROR('RUTINA LEE_FECHA; FECHA_HOY ',status_io);
   str_lee := '          ';
   str_lee[3] := '/';
   str_lee[6] := '/';
   move(whoy[1], str_lee[7], 4);
   move(whoy[5], str_lee[4], 2);
   move(whoy[7], str_lee[1], 2);
   COLORES(blue, lightgray);
   aux := length(texto_escrito) + columna + 1;
   xywrite(aux,renglon,str_lee);
   gotoxy(aux,renglon);
   i := 1;
   c := ' ';
   while (i < (length(str_lee) + 1)) and (ord(c) <> 13) do
      begin {while}
      indica_salto := false;
      if (i=3) or (i=6)then
         begin
         i:= i+1;
         indica_salto := true;
         gotoxy(aux+i-1,renglon);
         end;
      c:= readkey;
```

```
      if (ord(c)>=48) and (ord(c)<=57) then
         begin
         if not indica_salto then
            gotoxy(aux+i-1,renglon);
         write(c);
         str_lee[i] := c;
         i := i+1;
         end
      else
         if (ord(c) = 8) and (i>1) then
            begin
            if (i=4) or (i=7) then
               i:= i-2
            else
               i:= i-1;
            str_lee[i] := ' ';
            gotoxy(aux+i-1,renglon);
            write(' ');
            gotoxy(aux+i-1,renglon);
            end;
      end;{end}
   move(str_lee[7],fecha[1],4);
   move(str_lee[4],fecha[5],2);
   move(str_lee[1],fecha[7],2);
   COLORES(lightgray,blue);
   END; { LEE_FECHA }


PROCEDURE LEE_TEXTO (  columna,renglon : byte;
                       texto_escrito   : string;
                   var texto_leer      : string;
                       visible         : boolean;
                   var press_ESC       : boolean);
   VAR
      c:char;
      i,aux : integer;
   BEGIN
(*    FastText(texto_escrito, renglon, columna); *)
   xywrite(columna,renglon,texto_escrito);
   if visible then
      COLORES(blue,lightgray)
   else
      COLORES(lightgray, lightgray);
   aux := length(texto_escrito) + columna + 1;
   xywrite(aux,renglon,texto_leer);
   gotoxy(aux,renglon);
   i := 1;
   c := ' ';
   while (i<length(texto_leer) + 1) and (ord(c)<>13) and (ord(c) <> 27) do
      begin {while}
      c:= readkey;
      c:= upcase(c);
      if ord(c) = 27 then
         press_ESC := true;
      if ((ord(c)>=48) and (ord(c)<=57)) or
         ((ord(c)>=65) and (ord(c)<=90)) or
         (ord(c) = 32) then
         begin
         write(c);
         texto_leer[i] := c;
         i := i+1;
         end
      else
         if (ord(c) = 8) and (i>1) then
            begin
            i:= i-1;
            texto_leer[i] := ' ';
            gotoxy(aux+i-1,renglon);
            write(' ');
            gotoxy(aux+i-1,renglon);
            end;
      end;{end}
   COLORES(lightgray,blue);
   END; { LEE_TEXTO }
```

```
PROCEDURE LEE_NUMEROS(columna,renglon : byte;
                      texto_escrito   : string;
                  var texto_leer      : string);

   CONST
      punto = '.';
      vacio = ' ';
      coma  = ',';

   VAR
      caracter                : char;
      numero_char,
      columna_aux,
      columna_ult,
      long_cadena,
      long_aux,
      ind_poner_coma          : byte;
      texto_leer_comas,
      texto_leer_comas_aux    : string;
      tiene_punto             : boolean;
      indica                  : integer;

   BEGIN
   xywrite(columna,renglon,texto_escrito);
   COLORES(blue,lightgray);
   columna_aux := length(texto_escrito) + columna + 1;
   columna_ult := length(texto_escrito) + columna + length(texto_leer);
   long_cadena := length(texto_leer);
   xywrite(columna_aux,renglon,texto_leer);

   numero_char  := 1;
   caracter     := ' ';

   while (ord(caracter) <> 13) and (numero_char <= long_cadena) do
      begin {while}
      gotoxy(columna_ult,renglon);
      caracter:= readkey;
      if ((ord(caracter)>=48) and (ord(caracter)<=57)) or (caracter=punto)
      then
         begin
         long_aux := long_cadena + 1 - numero_char;
         for columna_aux := (columna_ult - numero_char + 1) to
              (columna_ult-1) do
            begin
            texto_leer[long_aux] := texto_leer[long_aux + 1];
            xywrite(columna_aux , renglon, texto_leer[long_aux]);
            long_aux := long_aux + 1;
            end;
         texto_leer[long_cadena] := caracter;
         xywrite(columna_ult , renglon, caracter);
         numero_char             := numero_char + 1;
         end
      else
         if (ord(caracter) = 8) and (numero_char>1) then
            begin
            long_aux := long_cadena;
            for columna_aux := (columna_ult) downto (columna_ult -
                 numero_char + 2) do
               begin
               texto_leer[long_aux] := texto_leer[long_aux-1];
               xywrite(columna_aux, renglon, texto_leer[long_aux]);
               long_aux := long_aux - 1;
               end;
            texto_leer[long_cadena - numero_char + 1] := vacio;
            xywrite(columna_ult - numero_char + 1, renglon, vacio);
            numero_char := numero_char - 1;
            end;
      end;{while}
   texto_leer_comas := texto_leer;
   tiene_punto := false;
   for indica := 1 to length(texto_leer_comas) do
      if texto_leer_comas[indica] = punto then
         tiene_punto := true;
   if not tiene_punto then
```

```
      texto_leer_comas := texto_leer_comas + '.00';
   indica              := length(texto_leer_comas);
   ind_poner_coma      := 0;
   texto_leer_comas_aux := '';
   while (indica > 0) and (texto_leer_comas[indica] <> vacio)  do
      begin
      if indica = (length(texto_leer_comas) - 2) then
         begin
         texto_leer_comas_aux := texto_leer_comas[indica] +
                                 texto_leer_comas_aux;
         ind_poner_coma       := 0;
         indica               := indica - 1;
         end;

      if ind_poner_coma = 3 then
         begin
         texto_leer_comas_aux := coma + texto_leer_comas_aux;
         ind_poner_coma       := 0;
         end;

      texto_leer_comas_aux := texto_leer_comas[indica] +
                              texto_leer_comas_aux;
      ind_poner_coma       := ind_poner_coma + 1;
      indica               := indica - 1;
      end;
   COLORES(lightgray,red);
   for indica := 1 to long_cadena do
      xywrite(length(texto_escrito)+columna+indica,renglon,' ');
   COLORES(blue,lightgray);
   xywrite(length(texto_escrito)+columna+1,renglon,texto_leer_comas_aux);
   COLORES(lightgray, blue);
   END;
END.
```

Programa con funciones para mensajes en pantalla:

```
{$I-,V-,R-,N+,E+,D+}
{$B-}
UNIT UMSJGIRO;
  { Unidad que contiene las rutinas de mensajes del sistema }

INTERFACE

   TYPE
      tipo_mensaje_error = (falla_sistema,
                            error_usuario,
                            aviso,
                            aviso_leer    );


   PROCEDURE COLORES(color_fore, color_back : byte);

   PROCEDURE XYWrite (x, y: byte; s: string);

   PROCEDURE ENCABEZADO_PANTALLA (proceso: string);

   PROCEDURE MENSAJE (titulo   : string;
                      texto    : string;
                      ren_sup  : byte;
                      tipo_msj : tipo_mensaje_error);

   PROCEDURE _DESPLIEGA_ERROR ( texto      : string; var salida :boolean );


IMPLEMENTATION

   USES TPCRT,
        TPSTRING,
        TPWINDOW,
        TPEDIT,
        COLORDEF,
        UDCLGIRO,
        UCONTROL,
        UGRALGIR,
        URUTGIRO;


   PROCEDURE COLORES (color_fore, color_back : byte);
      BEGIN
      textcolor(color_fore);
      textbackground(color_back);
      END;


   PROCEDURE XYWrite (x, y: byte; s: string);
      BEGIN
      GoToXY(x,y);
      write(s);
      END;


   PROCEDURE ENCABEZADO_PANTALLA (proceso: string);
      VAR
        fecha : tpo_fecha;
        status_io : integer;
        fecha_str : string;

      BEGIN
      FECHA_HOY(fecha, status_io);
      if status_io <> ok_io then
         DESPLIEGA_ERROR('ENCABEZADO_PANTALLA.FECHA_HOY', status_io);
      fecha_str := FECHA_TEXTO(fecha);
      window(1,1,80,25);
      COLORES(lightgray,blue);
      ClrScr;
      window(1,1,80,4);
```

```
    COLORES(yellow,blue);
    SetFrameChars(#186,#205,#188,#187,#200,#201);  { doble }
    Framewindow(1,1,80,4, $1E, $70, '');
    XYWrite(3, 2,'');
    XYWrite(3, 3,'SISGIROS');
    XYWrite(80-length(proceso)-1,2,proceso);
    XYWrite(80 - length(fecha_str)-1,3,fecha_str);
    COLORES(lightgray, black);
    gotoxyabs(1,5);
    clreol;
    Framewindow(1,6,80,24, $1E, $70,'');
    window(1,1,80,25);
    gotoxyabs(1,25);
    clreol;
    COLORES(lightgray,blue)
    END; { ENCABEZADO_PANTALLA }


PROCEDURE MENSAJE (titulo,
                   texto    : string;
                   ren_sup  : byte;
                   tipo_msj : tipo_mensaje_error);
  { Versión original de RMA }

  VAR
     c       : char;
     col_izq,
     col_der,
     ren_inf : byte;

  BEGIN
  col_izq := (80 - length(texto)) div 2;
  col_der := col_izq + length(texto) + 3;
  ren_inf := ren_sup + 2;

  if (tipo_msj = error_usuario) or
     (tipo_msj = falla_sistema) then
     begin
     COLORES(lightgray, blue);
     window(3, ren_sup, 77, ren_inf);
     clrscr;
     end;
  window(col_izq,ren_sup,col_der,ren_inf);
  SetFrameChars(#186,#205,#188,#187,#200,#201);

  case tipo_msj of
    error_usuario :
       begin
       COLORES(yellow,red);
       Framewindow(col_izq,ren_sup,col_der,ren_inf,$4E,$4E,titulo);
       end;
    falla_sistema :
       begin
       COLORES(yellow,black);
       Framewindow(col_izq,ren_sup,col_der,ren_inf,$0E,$0E,titulo);
       end;
    aviso         :
       begin
       COLORES(red,lightgray);
       Framewindow(col_izq,ren_sup,col_der,ren_inf,$74,$74,titulo);
       end;
    aviso_leer    :
       begin
       COLORES(white,red);
       Framewindow(col_izq,ren_sup,col_der,ren_inf,$4F,$4F,titulo);
       end;
    end;


  XYwrite(2,2,' ' + texto + ' ');

  COLORES(lightgray,blue);

  if tipo_msj = error_usuario then
     begin
```

```
            sound(440);
            delay(800);
            nosound;
            end;

         if (tipo_msj = error_usuario) or (tipo_msj = falla_sistema) then
            begin
            repeat until keypressed;
            c:=readkey;
            clrscr;
            end;

         window(1,1,80,25);
         END; { MENSAJE }

      PROCEDURE _DESPLIEGA_ERROR ( texto      : string; var salida: boolean );
         { Crea una ventana para desplegar un texto de error.
           Es llamado cuando ocurren errores no asociados con Btrieve. }
         CONST
            enter = ^M;
            esc   = ^[;
         VAR
            win         : WindowPtr;
            encabezado  : string[15];
            car         :  char;

         BEGIN
         encabezado := '';
         FrameChars := '++++-|';
         Shadow := true;
         Explode := true;
         SoundFlagW := false;
         if MakeWindow (win,5,20,75,22,true,true,false,
                        BlackOnLtGray, BlackOnLtGray, WhiteOnLtGray,
                        encabezado)
            AND DisplayWindow(win)
         then
            FastCenter(texto,1,BlackOnLtGray)
         else
            begin
            FastWrite(Pad(texto,80), 23,1,WhiteOnBlue);
            FastWrite(Pad(' ERROR FATAL: Memoria insuficiente para la ventana'
                        + ' de error',80), 24,1, WhiteOnBlue);
            end;
         ReadCharacter (Pad(' ENTER-Continuar  ESC-Interrumpir: ', 80), 25,1,
                        BlackOnLtGray, [ENTER, ESC], car   );
         salida := (car = ENTER);
         DisposeWindow(EraseTopWindow);
         FastWrite(Pad(' ', 80), 25,1, WhiteonBlack);
         END; { _DESPLIEGA_ERROR }
END.
```

Programa con funciones para convertir montos a letras en idioma ingles:

```
{
  PROGRAMA: UNUMBLET.PAS
  OBJETIVO: Unidad con procedimiento para trasnformar un número en letras
            en ingles.
  }

UNIT UNUMBLET;

{SI-,V-,R-,N+,E+,F+}
INTERFACE

   USES
      TPCRT,
      TPDATE,
      TPDOS,
      TPSTRING;
   CONST
      {nombre de números que son únicos}
      numero_00 = 'ZERO';
      numero_01 = 'ONE';
      numero_02 = 'TWO';
      numero_03 = 'THREE';
      numero_04 = 'FOUR';
      numero_05 = 'FIVE';
      numero_06 = 'SIX';
      numero_07 = 'SEVEN';
      numero_08 = 'EIGHT';
      numero_09 = 'NINE';
      numero_10 = 'TEN';
      numero_11 = 'ELEVEN';
      numero_12 = 'TWELVE';
      numero_13 = 'THIRTEEN';
      numero_14 = 'FOURTEEN';
      numero_15 = 'FIFTEEN';
      numero_16 = 'SIXTEEN';
      numero_17 = 'SEVENTEEN';
      numero_18 = 'EIGHTEEN';
      numero_19 = 'NINETEEN';
      numero_20 = 'TWENTY';
      numero_30 = 'THIRTY';
      numero_40 = 'FORTY';
      numero_50 = 'FIFTY';
      numero_60 = 'SIXTY';
      numero_70 = 'SEVENTY';
      numero_80 = 'EIGHTY';
      numero_90 = 'NINETY';

      {nombre de palabras que forman números}
      prefijo_mil     = 'THOUSAND';
      prefijo_cientos= 'HUNDRED';
      prefijo_millon = 'MILLION';
      prefijo_miles_millon = 'MILLIARD';
      prefijo_billon = 'BILLION';


      {palabras de unión que forman números}
      plural_es = 'S';
      sufijo_y  = 'AND';
      sufijo_con = 'WITH';
      sufijo_centavos = 'HUNDRETHS';

      {limites numéricos máximos y minimos}
      limite_maximo_billones = 9999999999999999.00;
      limite_minimo_billones =      1000000000000.00;
      limite_maximo_millones =       999999999999.00;
      limite_minimo_millones =            1000000.00;
      limite_maximo_miles    =             999999.00;
      limite_minimo_miles    =               1000.00;
      limite_maximo_cientos  =                999.00;
      limite_minimo_cientos  =                100.00;
      limite_maximo_decenas  =                 99.00;
```

```
    limite_minimo_decenas  =                20.00;
    unidad                 =                 1.00;

    {número de decenas}
    diez       = 10;
    veinte     = 20;
    treinta    = 30;
    cuarenta   = 40;
    cincuenta  = 50;
    sesenta = 60;
    setenta = 70;
    ochenta = 80;
    noventa = 90;
    cien    = 100;

    {otras constantes}
    un_blanco = ' ';

  VAR
    monto_en_letra  : string;          {resultado del monto en letras}
    arreglo_1_a_19  : array[1..19] of string;
    arreglo_decenas : array[1..9] of string;
    numero_decenas  : array[1..9] of byte;
    letrero         : string;          {parcial del monto en letras}

  PROCEDURE SEPARA_CANTIDAD(     monto_a_separar   : extended;
                                 numero_centavos   : byte;
                             var valor_centavos    : extended;
                             var valor_parteentera : extended;
                             var error_unit        : integer);
  PROCEDURE NUMBER_TO_LETERS(var monto_a_convertir : extended;
                                 numero_centavos   : byte;
                                 indicador_moneda  : string;
                                 indicador_final   : string;
                                 convierte_ctvs    : boolean;
                             var monto_en_letra    : string;
                             var error_unit        : integer );

IMPLEMENTATION

  USES
   DOS;

{=============================================================}
PROCEDURE LLENA_1_A_19;
BEGIN  {LLENA_1_A_19}
   arreglo_1_a_19[1] := numero_01;
   arreglo_1_a_19[2] := numero_02;
   arreglo_1_a_19[3] := numero_03;
   arreglo_1_a_19[4] := numero_04;
   arreglo_1_a_19[5] := numero_05;
   arreglo_1_a_19[6] := numero_06;
   arreglo_1_a_19[7] := numero_07;
   arreglo_1_a_19[8] := numero_08;
   arreglo_1_a_19[9] := numero_09;
   arreglo_1_a_19[10] := numero_10;
   arreglo_1_a_19[11] := numero_11;
   arreglo_1_a_19[12] := numero_12;
   arreglo_1_a_19[13] := numero_13;
   arreglo_1_a_19[14] := numero_14;
   arreglo_1_a_19[15] := numero_15;
   arreglo_1_a_19[16] := numero_16;
   arreglo_1_a_19[17] := numero_17;
   arreglo_1_a_19[18] := numero_18;
   arreglo_1_a_19[19] := numero_19;
END;   {LLENA_1_A_19}

{=============================================================}
PROCEDURE LLENA_ARREGLO_DECENAS;
BEGIN  {LLENA_ARREGLO_DECENAS}
   arreglo_decenas[01] := numero_10;
   arreglo_decenas[02] := numero_20;
   arreglo_decenas[03] := numero_30;
   arreglo_decenas[04] := numero_40;
   arreglo_decenas[05] := numero_50;
```

```
   arreglo_decenas[06] := numero_60;
   arreglo_decenas[07] := numero_70;
   arreglo_decenas[08] := numero_80;
   arreglo_decenas[09] := numero_90;
   numero_decenas[01]  := diez;
   numero_decenas[02]  := veinte;
   numero_decenas[03]  := treinta;
   numero_decenas[04]  := cuarenta;
   numero_decenas[05]  := cincuenta;
   numero_decenas[06]  := sesenta;
   numero_decenas[07]  := setenta;
   numero_decenas[08]  := ochenta;
   numero_decenas[09]  := noventa;
END;   {LLENA_ARREGLO_DECENAS}

{=============================================================}
PROCEDURE SEPARA_CANTIDAD(      monto_a_separar   : extended;
                                numero_centavos   : byte;
                           var valor_centavos     : extended;
                           var valor_parteentera  : extended;
                           var error_unit         : integer);
CONST
   long_max_numero = 19;
   base_centavos   = 10;
VAR
   monto_str      : string;              {es monto_a_separar en string}
   contador, factor_division : integer;

BEGIN   {SEPARA_CANTIDAD}
   valor_centavos    := 0;         {inicializa valor de centavos a extraer}
   valor_parteentera := 0;         {inicializa valor de parte entera a leer}
   error_unit        := 0;         {inicializa error. 0 = no existe error}
   If (numero_centavos = 0) Then  {monto sin centavos}
      Begin
      valor_centavos    := 0;
      valor_parteentera := monto_a_separar;
      End
   Else   {el monto se forma de parte entera y centavos}
      Begin
      Str(monto_a_separar:long_max_numero:numero_centavos, monto_str);
      Val(copy(monto_str,long_max_numero-(numero_centavos-1),
               numero_centavos), valor_centavos, error_unit);
      If (error_unit = 0) Then
         Begin
         factor_division := 1;
         For contador := 1 to numero_centavos Do
            factor_division := factor_division * base_centavos;
         valor_parteentera := monto_a_separar -
                              (valor_centavos/factor_division);
         End;  {endif error_unit <> 0}
      End;  {endelse el monto se formó de parte entera y centavos}
   If ((valor_parteentera < 1) and (valor_parteentera <> 0)) or
      ((valor_centavos < 1) and (valor_centavos <> 0))
   Then    {se interpretó algo como decimales}
      error_unit := 4;
END;    {SEPARA_CANTIDAD}

{=============================================================}
PROCEDURE NUMBER_TO_LETERS(var monto_a_convertir : extended;
                               numero_centavos   : byte;
                               indicador_moneda  : string;
                               indicador_final   : string;
                               convierte_ctvs    : boolean;
                           var monto_en_letra    : string;
                           var error_unit        : integer );

VAR
   centavos_extraidos   : extended;
   parteentera_extraida : extended;

   {=============================================================}
   PROCEDURE INTERPRETA_CANTIDAD(    cantidad : extended;
                                 var letrero  : string;
                                 var error_unit : integer);
   VAR
```

```
    indice          : longint;
    factor_division : extended;
    aux_num         : longint;
    aux_str         : string;

BEGIN  {INTERPRETA_CANTIDAD}
   if (cantidad >= 0) and (cantidad <= 19) then
      begin
      if cantidad = 0 then
         letrero := letrero + numero_00
      else
         begin
         aux_str := Real2Str(cantidad, 19, 2);
         aux_str := Trim(aux_str);
         aux_str :=copy(aux_str,1,pos('.',aux_str) - 1);
         if Str2Long(aux_str, aux_num) then
            begin
            letrero := letrero + arreglo_1_a_19[aux_num];
            end;
         end;
      end
   else   {hubo algún error}
     if (cantidad <> 0) and (cantidad < 1) Then
        error_unit := 4
     else
        if (cantidad >= limite_minimo_decenas) and
           (cantidad <= limite_maximo_decenas) then
           Begin
           indice := Trunc(cantidad / diez);
           cantidad := cantidad - (indice * diez);
           If (cantidad <> 0) Then          {fué un número que no es
                                             cerrado}
              Begin   {pone prefijos dieci ó veinti porque no llevan el
                       'y'}
              If (indice = 1) Then
                 letrero := letrero + arreglo_1_a_19[Trunc(cantidad)]
              Else
                 letrero := letrero + arreglo_decenas[indice] +
                                      un_blanco;
              INTERPRETA_CANTIDAD(cantidad, letrero, error_unit);
              End    {endif cantidad <> 0}
           Else  {fué una decena cerrada}
              letrero := letrero + arreglo_decenas[indice];
           End
        else
           if (cantidad >= limite_minimo_cientos) and
              (cantidad <= limite_maximo_cientos) then
              Begin
              indice := Trunc(cantidad / cien);          {cantidad div
                                                          cien}
              cantidad := cantidad - (indice * cien);    {cantidad mod
                                                          cien}
              letrero := letrero + arreglo_1_a_19[indice] + un_blanco +
                         prefijo_cientos + un_blanco;
              If (cantidad <> 0) Then
                 INTERPRETA_CANTIDAD(cantidad, letrero, error_unit);
              End

           else
              Begin
              If (cantidad < limite_minimo_millones) Then   {1,000 <=cant<=
                                                             999,999}
                 factor_division := limite_minimo_miles
              Else       {prueba que 1,000,000 <= cantidad <=
                          999,999,999,999}
              If (cantidad < limite_minimo_billones) Then
                 factor_division := limite_minimo_millones
              Else       {prueba que 1,000,000,000,000 <= cantidad (o sea,
                          billones)}
                 factor_division := limite_minimo_billones;
              indice   := Trunc(cantidad / factor_division);
              cantidad := cantidad - (indice * factor_division);
              INTERPRETA_CANTIDAD(indice, letrero, error_unit);
              If (factor_division = limite_minimo_miles) Then
                 letrero := letrero + un_blanco + prefijo_mil + un_blanco
```

```
            Else
            If (factor_division = limite_minimo_millones) Then
               If (indice = unidad) Then      {debe decir un millón algo}
                  letrero := letrero + un_blanco + prefijo_millon +
                             un_blanco
               Else                           {debe decir x millones
                                                algo}
                  letrero := letrero + un_blanco+ prefijo_millon +
                             plural_es + un_blanco
            Else
            If (factor_division = limite_minimo_billones) Then
               If (indice = unidad) Then      {debe decir un billón algo}
                  letrero := letrero + un_blanco + prefijo_billon +
                             un_blanco
               Else
                  letrero := letrero + un_blanco + prefijo_billon +
                             plural_es + un_blanco;
            If (cantidad <> 0) Then       {no fué una cifra cerrada}
               INTERPRETA_CANTIDAD(cantidad, letrero, error_unit);
            End;    {endif cantidad fué una cantidad > 999}
   END;   {INTERPRETA_CANTIDAD}

BEGIN  {CANTIDAD_A_LETRAS}
  letrero := '';
  SEPARA_CANTIDAD(monto_a_convertir, numero_centavos,
                  centavos_extraidos, parteentera_extraida, error_unit);
  If (error_unit = 0) Then
     Begin
     INTERPRETA_CANTIDAD(parteentera_extraida, letrero, error_unit);
     letrero := letrero + un_blanco + indicador_moneda;
     If (convierte_ctvs) and (numero_centavos <> 0) and
        (error_unit = 0) then
        Begin
        letrero := letrero + un_blanco + sufijo_con + un_blanco;
        INTERPRETA_CANTIDAD(centavos_extraidos, letrero, error_unit);
        letrero := letrero + un_blanco + sufijo_centavos + indicador_final;
        End  {endif numero_centavos <> 0}
     Else
        letrero := letrero + indicador_final;
     End;  {endif error_unit = 0}
  monto_en_letra := Trim('(' + Trim(letrero) + ')');
END;   {CANTIDAD_A_LETRAS}

{================================================================}
BEGIN
LLENA_1_A_19;
LLENA_ARREGLO_DECENAS;
END.   {UNIT UNUMBLET}
```

Programa con funciones para convertir montos a letras en idioma español:

```
{
  PROGRAMA: UNUMLETR.PAS
  OBJETIVO: Unidad con procedimiento para trasnformar un número en letras.
  }

UNIT UNUMLETR;

{$I-,V-,R-,N+,E+,O+,S-}

INTERFACE

   USES
      TPCRT,
      TPDATE,
      TPDOS,
      TPSTRING;
   CONST
      {nombre de números que son únicos}
      numero_00 = 'CERO';
      numero_01 = 'UN';           numero_20 = 'VEINTE';
      numero_02 = 'DOS';          numero_30 = 'TREINTA';
      numero_03 = 'TRES';         numero_40 = 'CUARENTA';
      numero_04 = 'CUATRO';       numero_50 = 'CINCUENTA';
      numero_05 = 'CINCO';        numero_60 = 'SESENTA';
      numero_06 = 'SEIS';         numero_70 = 'SETENTA';
      numero_07 = 'SIETE';        numero_80 = 'OCHENTA';
      numero_08 = 'OCHO';         numero_90 = 'NOVENTA';
      numero_09 = 'NUEVE';        numero_100 = 'CIEN';
      numero_10 = 'DIEZ';         numero_500 = 'QUINIENTOS';
      numero_11 = 'ONCE';         numero_700 = 'SETECIENTOS';
      numero_12 = 'DOCE';         numero_900 = 'NOVECIENTOS';
      numero_13 = 'TRECE';        numero_1000 = 'MIL';
      numero_14 = 'CATORCE';
      numero_15 = 'QUINCE';

      {nombre de palabras que forman números}
      prefijo_mil    = 'MIL';
      prefijo_dieci  = 'DIECI';
      prefijo_veinti = 'VEINTI';
      prefijo_ciento = 'CIENTO';
      prefijo_cientos= 'CIENTOS';
      prefijo_millon = 'MILLON';
      prefijo_billon = 'BILLON';

      {palabras de unión que forman números}
      plural_es = 'ES';
      sufijo_y  = 'Y';
      pesos_mn  = 'PESOS M.N.';
      sufijo_con = 'CON';
      sufijo_centavos = 'CENTAVOS';
      sufijo_dolares  = 'DOLARES AMERICANOS';

      {límites numéricos máximos y mínimos}
      limite_maximo_billones = 9999999999999999.99;
      limite_minimo_billones =      1000000000000.00;
      limite_maximo_millones =       999999999999.99;
      limite_minimo_millones =            1000000.00;
      limite_maximo_miles    =             999999.99;
      limite_minimo_miles    =               1000.00;
      limite_maximo_cientos  =                999.99;
      limite_minimo_cientos  =                100.00;
      unidad                 =                  1.00;

      {número de decenas}
      diez      = 10;             sesenta = 60;
      veinte    = 20;             setenta = 70;
      treinta   = 30;             ochenta = 80;
      cuarenta  = 40;             noventa = 90;
      cincuenta = 50;             cien    = 100;
```

```
    {otras constantes}
    un_blanco = ' ';

  VAR
    monto_en_letra  : string;          {resultado del monto en letras}
    arreglo_1_a_15  : array[1..15] of string;
    arreglo_decenas : array[1..9] of string;
    numero_decenas  : array[1..9] of byte;
    letrero         : string;          {parcial del monto en letras}

  PROCEDURE LLENA_1_A_15;
  PROCEDURE LLENA_ARREGLO_DECENAS;
  PROCEDURE SEPARA_CANTIDAD(      monto_a_separar   : extended;
                                  numero_centavos   : byte;
                            var valor_centavos      : extended;
                            var valor_parteentera   : extended;
                            var error_unit          : integer);
  PROCEDURE CANTIDAD_A_LETRAS(var monto_a_convertir : extended;
                                  numero_centavos   : byte;
                                  indicador_moneda  : string;
                                  indicador_final   : string;
                                  convierte_ctvs    : boolean;
                            var monto_en_letra      : string;
                            var error_unit          : integer );

IMPLEMENTATION

  USES
    DOS;

{=============================================================}
PROCEDURE LLENA_1_A_15;
BEGIN  {LLENA_1_A_15}
  arreglo_1_a_15[01] := numero_01;
  arreglo_1_a_15[02] := numero_02;
  arreglo_1_a_15[03] := numero_03;
  arreglo_1_a_15[04] := numero_04;
  arreglo_1_a_15[05] := numero_05;
  arreglo_1_a_15[06] := numero_06;
  arreglo_1_a_15[07] := numero_07;
  arreglo_1_a_15[08] := numero_08;
  arreglo_1_a_15[09] := numero_09;
  arreglo_1_a_15[10] := numero_10;
  arreglo_1_a_15[11] := numero_11;
  arreglo_1_a_15[12] := numero_12;
  arreglo_1_a_15[13] := numero_13;
  arreglo_1_a_15[14] := numero_14;
  arreglo_1_a_15[15] := numero_15;
END;   {LLENA_1_A_15}

{=============================================================}
PROCEDURE LLENA_ARREGLO_DECENAS;
BEGIN  {LLENA_ARREGLO_DECENAS}
  arreglo_decenas[01] := numero_10;
  arreglo_decenas[02] := numero_20;
  arreglo_decenas[03] := numero_30;
  arreglo_decenas[04] := numero_40;
  arreglo_decenas[05] := numero_50;
  arreglo_decenas[06] := numero_60;
  arreglo_decenas[07] := numero_70;
  arreglo_decenas[08] := numero_80;
  arreglo_decenas[09] := numero_90;
  numero_decenas[01]  := diez;
  numero_decenas[02]  := veinte;
  numero_decenas[03]  := treinta;
  numero_decenas[04]  := cuarenta;
  numero_decenas[05]  := cincuenta;
  numero_decenas[06]  := sesenta;
  numero_decenas[07]  := setenta;
  numero_decenas[08]  := ochenta;
  numero_decenas[09]  := noventa;
END;   {LLENA_ARREGLO_DECENAS}

{=============================================================}
PROCEDURE SEPARA_CANTIDAD(      monto_a_separar   : extended;
```

```
                                    numero_centavos    : byte;
                                var valor_centavos     : extended;
                                var valor_parteentera  : extended;
                                var error_unit         : integer);
CONST
   long_max_numero = 19;
   base_centavos   = 10;
VAR
   monto_str      : string;                (es monto_a_separar en string)
   contador, factor_division : integer;

BEGIN   {SEPARA_CANTIDAD}
   valor_centavos    := 0;        {inicializa valor de centavos a extraer}
   valor_parteentera := 0;        {inicializa valor de parte entera a leer}
   error_unit        := 0;        {inicializa error. 0 = no existe error}
   If (numero_centavos = 0) Then  {monto sin centavos}
      Begin
      valor_centavos    := 0;
      valor_parteentera := monto_a_separar;
      End
   Else   {el monto se forma de parte entera y centavos}
      Begin
      Str(monto_a_separar:long_max_numero:numero_centavos, monto_str);
      Val(copy(monto_str,long_max_numero-(numero_centavos-1),
           numero_centavos),
           valor_centavos, error_unit);
      If (error_unit = 0) Then
         Begin
         factor_division := 1;
         For contador := 1 to numero_centavos Do
            factor_division := factor_division * base_centavos;
         valor_parteentera := monto_a_separar -
                              (valor_centavos/factor_division);
         End;  {endif error_unit <> 0}
      End;  {endelse el monto se formó de parte entera y centavos}
   If ((valor_parteentera < 1) and (valor_parteentera <> 0)) or
      ((valor_centavos < 1) and (valor_centavos <> 0))
   Then    {se interpretó algo como decimales}
      error_unit := 4;
END;    {SEPARA_CANTIDAD}

{=============================================================}
PROCEDURE CANTIDAD_A_LETRAS(var monto_a_convertir : extended;
                                numero_centavos   : byte;
                                indicador_moneda  : string;
                                indicador_final   : string;
                                convierte_ctvs    : boolean;
                            var monto_en_letra    : string;
                            var error_unit        : integer );

VAR
   centavos_extraidos   : extended;
   parteentera_extraida : extended;

   {=============================================================}
   PROCEDURE INTERPRETA_CANTIDAD(    cantidad : extended;
                                 var letrero  : string;
                                 var error_unit : integer);
   VAR
      indice          : longint;
      factor_division : extended;
   BEGIN   {INTERPRETA_CANTIDAD}
      If (cantidad < 16) Then  {se trata de un número entre 1 y 15}
         Begin
         If (cantidad = 0) Then
            letrero := letrero + numero_00;
         If (cantidad = unidad) Then
            letrero := letrero + arreglo_1_a_15[1]
         Else
         If (cantidad = 2) Then
            letrero := letrero + arreglo_1_a_15[2]
         Else
         If (cantidad = 3) Then
            letrero := letrero + arreglo_1_a_15[3]
         Else
```

```
      If (cantidad = 4) Then
         letrero := letrero + arreglo_1_a_15[4]
      Else
      If (cantidad = 5) Then
         letrero := letrero + arreglo_1_a_15[5]
      Else
      If (cantidad = 6) Then
         letrero := letrero + arreglo_1_a_15[6]
      Else
      If (cantidad = 7) Then
         letrero := letrero + arreglo_1_a_15[7]
      Else
      If (cantidad = 8) Then
         letrero := letrero + arreglo_1_a_15[8]
      Else
      If (cantidad = 9) Then
         letrero := letrero + arreglo_1_a_15[9]
      Else
      If (cantidad = diez) Then
         letrero := letrero + arreglo_1_a_15[diez]
      Else
      If (cantidad = 11) Then
         letrero := letrero + arreglo_1_a_15[diez + 1]
      Else
      If (cantidad = 12) Then
         letrero := letrero + arreglo_1_a_15[12]
      Else
      If (cantidad = 13) Then
         letrero := letrero + arreglo_1_a_15[13]
      Else
      If (cantidad = 14) Then
         letrero := letrero + arreglo_1_a_15[14]
      Else
      If (cantidad = 15) Then
         letrero := letrero + arreglo_1_a_15[15]
      Else   {hubo algún error}
      If (cantidad <> 0) and (cantidad < 1) Then
         error_unit := 4;
      End   {endif entre 1 y 16}
   Else      {cantidad entre 16 y 100}

   If (cantidad < cien) Then
      Begin
      indice := Trunc(cantidad / diez);
      cantidad := cantidad - (indice * diez);
      If (cantidad <> 0) Then          {fué un número que no es cerrado}
         Begin   {pone prefijos dieci ó veinti porque no llevan el 'y'}
         If (indice = 1) Then
            letrero := letrero + prefijo_dieci
         Else
         If (indice = 2) Then
            letrero := letrero + prefijo_veinti
         Else
            letrero := letrero + arreglo_decenas[indice] + un_blanco +
                       sufijo_y + un_blanco;
         INTERPRETA_CANTIDAD(cantidad, letrero, error_unit);
         End    {endif cantidad <> 0}
      Else  {fué una decena cerrada}
         letrero := letrero + arreglo_decenas[indice];
      End  {endif cantidad < cien}
   Else {elseif cantidad < cien. Interpreta cantidad entre 100 y 999}

   If (cantidad < limite_minimo_miles) Then
      Begin
      indice := Trunc(cantidad / cien);          {cantidad div cien}
      cantidad := cantidad - (indice * cien);    {cantidad mod cien}
      If (indice = 1) Then
         Begin
         If (cantidad <> 0) Then      {no fué el número 100}
            letrero := letrero + prefijo_ciento + un_blanco
         Else   {fué el número 100}
            letrero := letrero + numero_100;
         End  {endif indice = 1}
      Else
         If (indice = 5) Then
```

```
            letrero := letrero + numero_500 + un_blanco
         Else
            If (indice = 7) Then
               letrero := letrero + numero_700 + un_blanco
            Else
               If (indice = 9) Then
                  letrero := letrero + numero_900 + un_blanco
               Else  {fué 200,300,400,600,800}
                  letrero := letrero + arreglo_1_a_15[indice] +
                             prefijo_cientos + un_blanco;

      If (cantidad <> 0) Then
         INTERPRETA_CANTIDAD(cantidad, letrero, error_unit);
      End   {cantidad < 1000}
   Else  {fué una cantidad mayor a 999}

      Begin
      If (cantidad < limite_minimo_millones) Then   {1,000 <=cant<=
                                                     999,999}
         factor_division := limite_minimo_miles
      Else       {prueba que 1,000,000 <= cantidad <=  999,999,999,999}
      If (cantidad < limite_minimo_billones) Then
         factor_division := limite_minimo_millones
      Else      {prueba que 1,000,000,000,000 <= cantidad (o sea,
                 billones)}
         factor_division := limite_minimo_billones;

      indice   := Trunc(cantidad / factor_division);
      cantidad := cantidad - (indice * factor_division);
      INTERPRETA_CANTIDAD(indice, letrero, error_unit);
      If (factor_division = limite_minimo_miles) Then
         letrero := letrero + un_blanco + prefijo_mil + un_blanco
      Else
      If (factor_division = limite_minimo_millones) Then
         If (indice = unidad) Then       {debe decir un millón algo}
            letrero := letrero + un_blanco + prefijo_millon + un_blanco
         Else                            {debe decir x millones algo}
            letrero := letrero + un_blanco+ prefijo_millon + plural_es +
                       un_blanco
      Else
      If (factor_division = limite_minimo_billones) Then
         If (indice = unidad) Then       {debe decir un billón algo}
            letrero := letrero + un_blanco + prefijo_billon + un_blanco
         Else
            letrero := letrero + un_blanco + prefijo_billon + plural_es
                       + un_blanco;
      If (cantidad <> 0) Then       {no fué una cifra cerrada}
         INTERPRETA_CANTIDAD(cantidad, letrero, error_unit);
      End;    {endif cantidad fué una cantidad > 999}
  END;   {INTERPRETA_CANTIDAD}

BEGIN  {CANTIDAD_A_LETRAS}
  letrero := '';
  SEPARA_CANTIDAD(monto_a_convertir, numero_centavos,
                  centavos_extraidos, parteentera_extraida, error_unit);
  If (error_unit = 0) Then
     Begin
     INTERPRETA_CANTIDAD(parteentera_extraida, letrero, error_unit);
     letrero := letrero + un_blanco + indicador_moneda;
     If (convierte_ctvs) and (numero_centavos <> 0) and
        (error_unit = 0) Then
        Begin
        letrero := letrero + un_blanco + sufijo_con + un_blanco;
        INTERPRETA_CANTIDAD(centavos_extraidos, letrero, error_unit);
        letrero := letrero + un_blanco + sufijo_centavos + indicador_final;
        End  {endif numero_centavos <> 0}
     Else
        letrero := letrero + indicador_final;
     End;  {endif error_unit = 0}
  monto_en_letra := Trim('(' + Trim(letrero) + ')');
END;   {CANTIDAD_A_LETRAS}

{=============================================================}
BEGIN
LLENA_1_A_15;
```

```
LLENA_ARREGLO_DECENAS;
END.   {UNIT UNUMLETR}
```

Programa con funciones para manejar el archivo de cifras de control:

```
{
*************************************************************************
   STOI: UNUMGIR
*************************************************************************
   Archivo  : NUM_GIR
   Objetivo : Guarda las cifras de control de los giros.
*************************************************************************
}

UNIT UNUMGIR;

{$I-,V-,R-,N+,E+,D+}

INTERFACE

   USES
        UDCLGIRO;

   CONST

     { Nombre del archivo en MS-DOS: }
      archivo_num_giros= 'F:\GIROS\DATOS\NUM_GIRO.DAT';

   TYPE
      LLAVE1_NUM_GIR = longint;


      R_NUM_GIR = RECORD
         ident_num_gir     : longint;
         descripcion       : array [1.. 30] of char;
         numero_gir        : tpo_folio;
         num_aux_num_gir   : integer;
      END; { R_NUM_GIR}


   FUNCTION ABRE_NUM_GIR( modo : char ) : integer;
   FUNCTION CIERRA_NUM_GIR : integer;
   PROCEDURE ACTUALIZA1_NUM_GIR(    wr_num_gir  : r_num_gir;
                                var status_io : integer);
   PROCEDURE ESCRIBE_NUM_GIR(    wr_num_gir  : r_num_gir;
                             var status_io : integer         );
   PROCEDURE LEE_PRIMER_NUM_GIR(var wr_num_gir  : r_num_gir;
                                var llave_num_gir : llave1_num_gir;
                                var status_io : integer);
   PROCEDURE LEE_SIG_NUM_GIR(var wr_num_gir  : r_num_gir;
                             var llave_num_gir : llave1_num_gir;
                             var status_io : integer);
   PROCEDURE LEE1_NUM_GIR(var wr_num_gir: r_num_gir;
                              llave_num_gir : llave1_num_gir;
                            var status_io : integer              );
   PROCEDURE LEE1_GE_NUM_GIR (var wr_num_gir: r_num_gir;
                              var llave_num_gir: llave1_num_gir;
                              var status_io : integer );

   PROCEDURE LEE1_ULT_NUM_GIR(var wr_num_gir  : r_num_gir;
                           var status_io : integer          );
   PROCEDURE BORRA_NUM_GIR(var wr_num_gir: r_num_gir;
                               llave_num_gir : llave1_num_gir;
                           var status_io : integer);

IMPLEMENTATION

   USES
      UBTRIEVE;
```

```
   VAR

      long_reg_num_gir   : integer;
      bloque_pos_num_gir : array [1..128] of byte;


FUNCTION ABRE_NUM_GIR( modo : char ): integer;
   { Abre el archivo num_gir. }
   VAR
      dueno_archivo  : string[10];
      nombre_archivo : string[80];
      modo_int       : integer;
   BEGIN
   dueno_archivo := #0;
   nombre_archivo := archivo_num_giros + #0;
   case modo of
      modo_lectura       : modo_int := solo_lectura;
      modo_recuperacion : modo_int := con_recuperacion;
      end; {case}
   ABRE_NUM_GIR:= BTRV (abre_archivo, bloque_pos_num_gir,dueno_archivo[1],
                        long_reg_num_gir,nombre_archivo[1], modo_int       );
   END; { ABRE_NUM_GIR}


FUNCTION CIERRA_NUM_GIR: integer;
   BEGIN
   CIERRA_NUM_GIR:= BTRV (cierra_archivo, bloque_pos_num_gir, dummy,
                                  dummy, dummy, dummykey );
   END;


PROCEDURE ACTUALIZA1_NUM_GIR(     wr_num_gir    : r_num_gir;
                              var status_io  : integer);
   { Update con llave 1 }
   VAR
      llave1_control : llave1_num_gir;
   BEGIN
   status_io := BTRV (actualiza, bloque_pos_num_gir, wr_num_gir,
                      long_reg_num_gir, llave1_control, llave1 );
   END;


PROCEDURE ESCRIBE_NUM_GIR(    wr_num_gir    : r_num_gir;
                          var status_io  : integer);
   VAR
      llave1_control : llave1_num_gir;
   BEGIN
   status_io := BTRV (escribe, bloque_pos_num_gir, wr_num_gir,
                      long_reg_num_gir, llave1_control, llave1);
   END;


PROCEDURE LEE_PRIMER_NUM_GIR(var wr_num_gir  : r_num_gir;
                             var llave_num_gir : llave1_num_gir;
                             var status_io : integer);
   { Pone en wr_num_gir el primer registro de num_gir con llave 1 indicada.
}
   BEGIN
   status_io := BTRV (get_lowest, bloque_pos_num_gir, wr_num_gir,
                      long_reg_num_gir, llave_num_gir, llave1        );
   END;

PROCEDURE LEE_SIG_NUM_GIR(var wr_num_gir  : r_num_gir;
                          var llave_num_gir : llave1_num_gir;
                          var status_io : integer);
   { Pone en wr_num_gir el siguiente registro de giros con llave 1
indicada. }
   BEGIN
   status_io := BTRV (get_next, bloque_pos_num_gir, wr_num_gir,
                      long_reg_num_gir, llave_num_gir, llave1        );
   END;

PROCEDURE LEE1_NUM_GIR(var wr_num_gir  : r_num_gir;
                       llave_num_gir : llave1_num_gir;
```

```
                    var status_io : integer);
  { Pone en wr_num_gir el registro de giros con llave 1  indicada. }
  BEGIN
  status_io := BTRV (get_equal, bloque_pos_num_gir, wr_num_gir,
                    long_reg_num_gir, llave_num_gir, llave1        );
  END;


PROCEDURE LEE1_GE_NUM_GIR (var wr_num_gir : r_num_gir;
                           var llave_num_gir: llave1_num_gir;
                           var status_io : integer );
  BEGIN
  status_io := BTRV (get_greater_equal,bloque_pos_num_gir, wr_num_gir,
                    long_reg_num_gir, llave_num_gir,      llave1 );
  END;

PROCEDURE LEE1_ULT_NUM_GIR(var wr_num_gir  : r_num_gir;
                           var status_io : integer);
  {Pone en wr_num_gir el sig. registro del archivo, usando la llave 1.}
  VAR
    llave1_control : llave1_num_gir;
  BEGIN
  status_io := BTRV (get_highest, bloque_pos_num_gir, wr_num_gir,
                    long_reg_num_gir, llave1_control, llave1);
  END;


PROCEDURE BORRA_NUM_GIR(var wr_num_gir  : r_num_gir;
                            llave_num_gir : llave1_num_gir;
                        var status_io : integer);
  { Borra el registro  wr_num_gir de giros con  llave 1 indicada. }
  BEGIN
  status_io := BTRV (borra, bloque_pos_num_gir, wr_num_gir,
                    long_reg_num_gir, llave_num_gir, llave1        );
  END;


BEGIN
   long_reg_num_gir := sizeof(r_num_gir);
END. { UNIT UNUMGIR }
```

Programa con rutinas para claves de acceso:

```
{$I-,V-,R-,N+,E+,D+}
UNIT UPASSWOR;
  { Unidad que contiene las rutinas de claves y passwords }

INTERFACE

   USES
        UDCLGIRO;

   PROCEDURE PIDE_CLAVE( var clave    : tpo_operador;
                         var grupo    : tpo_grupo_usuarios );

   PROCEDURE CAMBIO_PASSWORD_SIGSE;

   PROCEDURE ALTA_CLAVE(grupo : tpo_grupo_usuarios);

   PROCEDURE BAJA_CLAVE(grupo : tpo_grupo_usuarios);


IMPLEMENTATION

   USES
        TPWINDOW,
        TPCRT,
        COLORDEF,
        UUSU_GIR,
        UCON_USU,
        UMSJGIRO,
        UGRALGIR,
        ULECTURA;

   VAR
   wr_usuarios  : r_usuarios;
   clave        : tpo_operador;
   password     : tpo_password;
   clave_aux    : string;
   password_aux : string;
   status_io    : integer;
   presiono_ESC : boolean;


   PROCEDURE PIDE_CLAVE( var clave    : tpo_operador;
                         var grupo    : tpo_grupo_usuarios );

      CONST
      no_maximo_de_intentos = 3;

      VAR
      clave_ok     : boolean;
      wtu,
      no_intentos  : integer;
      archivo      : text;

      BEGIN
      clave_ok := false;
      base     := nil;

      for wtu:=1 to sizeof(blancos) do
         udclgiro.blancos[wtu] := ' ';

      status_io :=
ABRES(base,'USU_GIRO',abre_usuarios,modo_lectura,cierra_usuarios);
      if status_io <> ok_io then
         DESPLIEGA_ERROR('RUTINA PASSWORD:ABRE_USUARIOS ',status_io);

      no_intentos := 0;

      repeat
         presiono_ESC := false;
         ENCABEZADO_PANTALLA('CLAVES DE ACCESO');
```

```
        no_intentos := no_intentos + 1;
        clave_aux    := '      ';
        password_aux := '                ';
        MENSAJE('','   Teclear CLAVE :                ',10,aviso_leer);
        LEE_TEXTO(43,11,'',clave_aux,True,presiono_ESC);
        if presiono_ESC then
           begin
           status_io := CIERRAS(base);
           halt(0);
           end;
        move(clave_aux[1], clave[1], 6);
        LEE_USUARIOS(wr_usuarios,clave,status_io);
        if (status_io <> ok_io) and (status_io <> no_existe_llave) then
           DESPLIEGA_ERROR ('RUTINA PASSWORD : LEE_USUARIOS ',status_io);

        case status_io of
           ok_io :
              begin
              MENSAJE('','Teclear PASSWORD :                ',
                       12,aviso_leer);
              LEE_TEXTO(43,13,'',password_aux, False, presiono_ESC);
              if presiono_ESC then
                 begin
                 status_io := CIERRAS(base);
                 halt(0);
                 end;
              move(password_aux[1], password[1], 12);
              if password = wr_usuarios.password_usu then
                 begin
                 clave_ok := true;
                 grupo := wr_usuarios.grupo_usu;
                 ACTUALIZA_CLAVE_USUARIO(wr_usuarios.clave_usu,
                                         wr_usuarios.grupo_usu);
                 end
              else
                 MENSAJE(' ERROR ','PASSWORD INCORRECTO', 21,
                          error_usuario);
              end;

           no_existe_llave :
              begin
              MENSAJE ('ERROR','CLAVE INCORRECTA',21,error_usuario);
              end;

           else
              DESPLIEGA_ERROR ('RUTINA PASSWORD : LEE_USUARIOS ',
                               status_io);
           end {case}
     until (clave_ok) or (no_intentos = no_maximo_de_intentos) ;
     status_io := CIERRAS(base);
     if status_io <> ok_io then
        DESPLIEGA_ERROR('RUTINA PIDE CLAVE : CIERRAS ',status_io);
     if (not clave_ok) then
        begin
        MENSAJE(' ERROR ','INTENTOS EXCEDIDOS; EJECUCION DEL SISTEMA
                 SUSPENDIDA', 21,error_usuario);
        ENCABEZADO_PANTALLA('MENU PRINCIPAL');
        halt(3);
        end
     else
        ENCABEZADO_PANTALLA('MENU PRINCIPAL');
     END;{PIDE_CLAVE}

PROCEDURE CAMBIO_PASSWORD_SIGSE;

     VAR
     password_nuevo1,
     password_nuevo2  : string;


     BEGIN
     base := nil;
     status_io := ABRES(base,'USUARIOS',abre_usuarios, modo_recuperacion,
                        cierra_usuarios);
     if status_io <> ok_io then
```

```
   DESPLIEGA_ERROR('RUTINA PASSWORD:ABRE_USUARIOS ',status_io);
ENCABEZADO_PANTALLA('CAMBIO DE PASSWORD');
clave_aux     := '      ';
password_aux := '            ';
password_nuevo1 := '            ';
password_nuevo2 := '            ';
presiono_ESC    := false;
MENSAJE('','          Teclear CLAVE :                ',10,
        aviso_leer);
LEE_TEXTO(47,11,'',clave_aux,true, presiono_ESC);
if presiono_ESC then
   begin
   status_io := CIERRAS(base);
   exit;
   end;
move(clave_aux[1],clave[1],6);
LEE_USUARIOS(wr_usuarios,clave,status_io);
if (status_io <> ok_io) and (status_io <> no_existe_llave) then
   DESPLIEGA_ERROR ('RUTINA PASSWORD : LEE_USUARIOS ',status_io);
if status_io <> no_existe_llave then
   begin
   MENSAJE('','       Teclear PASSWORD :              ',
           12,aviso_leer);
   LEE_TEXTO(47,13,'',password_aux,false , presiono_ESC);
   if presiono_ESC then
      begin
      status_io := CIERRAS(base);
      exit;
      end;
   move(password_aux[1],password[1],12);
   if password = wr_usuarios.password_usu then
      begin
      MENSAJE('','  Teclear PASSWORD NUEVO :                ',14,
              aviso_leer);
      LEE_TEXTO(47,15,'',password_nuevo1,false,presiono_ESC);
      if presiono_ESC then
         begin
         status_io := CIERRAS(base);
         exit;
         end;
      MENSAJE('','Reteclear PASSWORD NUEVO :                ',16,
              aviso_leer);
      LEE_TEXTO(47,17,'',password_nuevo2,false,presiono_ESC);
      if presiono_ESC then
         begin
         status_io := CIERRAS(base);
         exit;
         end;
      if password_nuevo1 = password_nuevo2 then
         begin
         if password_nuevo1 <> password then
            begin
            move(password_nuevo1[1],wr_usuarios.password_usu[1],12);
            ACTUALIZA_USUARIOS(wr_usuarios,status_io);
            if status_io <> ok_io then
               DESPLIEGA_ERROR('RUTINA CAMBIO_PASSW
                               :ACTUALIZA_USUARIOS ', status_io);
            MENSAJE('','CAMBIO REALIZADO',21,error_usuario);
            end
         else
            MENSAJE(' ERROR ','PASSWORD NUEVO IGUAL AL ANTERIOR;
                    CAMBIO CANCELADO',21,error_usuario);
         end
      else
         MENSAJE(' ERROR ','PASSWORDS DIFERENTES; CAMBIO
                 CANCELADO',21,error_usuario);
      end
   else
      MENSAJE(' ERROR ','PASSWORD INCORRECTO; CAMBIO
              CANCELADO',21,error_usuario);
   end
else
   MENSAJE(' ERROR ','CLAVE INCORRECTA; CAMBIO
              CANCELADO',21,error_usuario);
status_io := CIERRAS(base);
```

```
   ENCABEZADO_PANTALLA('MENU PRINCIPAL');
   END;

PROCEDURE ALTA_CLAVE(grupo : tpo_grupo_usuarios);

   VAR
   clave_nuevo      : string;
   password_nuevo1,
   password_nuevo2 : string;
   nombre_nuevo     : string;
   grupo_nuevo      : string;
   area_correcta    : boolean;

   BEGIN
   ENCABEZADO_PANTALLA('ALTA DE CLAVES');
   base := nil;
   status_io := ABRES(base,'USUARIOS',abre_usuarios,
                      modo_recuperacion,cierra_usuarios);
   if status_io <> ok_io then
      DESPLIEGA_ERROR('RUTINA PASSWORD:ABRE_USUARIOS ',status_io);
   clave_nuevo     := '       ';
   password_nuevo1 := '              ';
   password_nuevo2 := '              ';
   nombre_nuevo    := '                         ';
   grupo_nuevo     := '  ';
   presiono_ESC    := false;
   MENSAJE('','      Teclear CLAVE NUEVA :                          ',
           8, aviso_leer);
   LEE_TEXTO(42,9,'',clave_nuevo,true,presiono_ESC);
   if presiono_ESC then
      begin
      status_io := CIERRAS(base);
      exit;
      end;
   move(clave_nuevo[1],clave[1],6);
   LEE_USUARIOS(wr_usuarios,clave,status_io);
   case status_io of
      ok_io :
         MENSAJE(' ERROR ','YA EXISTE CLAVE; ALTA CANCELADA',21,
                 error_usuario);
      no_existe_llave     : begin
                            MENSAJE('','   Teclear PASSWORD NUEVO :',10,
                                    aviso_leer);
                            LEE_TEXTO(42,11,'',password_nuevo1,false,
                                      presiono_ESC);
                            if presiono_ESC then
                               begin
                               status_io := CIERRAS(base);
                               exit;
                               end;
                            MENSAJE('',' Reteclear PASSWORD NUEVO :',12,
                                    aviso_leer);
                            LEE_TEXTO(42,13,'',password_nuevo2,false,
                                      presiono_ESC);
                            if presiono_ESC then
                               begin
                               status_io := CIERRAS(base);
                               exit;
                               end;
                            if password_nuevo1 <> password_nuevo2 then
                               MENSAJE (' ERROR ','PASSWORDS DIFERENTES;
                                 ALTA CANCELADA',21,error_usuario)
                            else
                               begin
                               MENSAJE('','      Teclear NOMBRE NUEVO :',
                                       14, aviso_leer);
                               LEE_TEXTO(42,15,'',nombre_nuevo,true,
                                         presiono_ESC);
                               if presiono_ESC then
                                  begin
                                  status_io := CIERRAS(base);
                                  exit;
                                  end;
                               area_correcta := false;
```

```
                                repeat
                                 MENSAJE('','Teclear AREA del USUARIO :',
                                          16 ,aviso_leer);
                                   LEE_TEXTO(42,17,'',grupo_nuevo,true,
                                                presiono_ESC);
                                   if presiono_ESC then
                                      begin
                                      status_io := CIERRAS(base);
                                      exit;
                                      end;
                                      area_correcta := (((grupo_nuevo =
                                                supervisor_giros)or
                                                (grupo_nuevo =
                                                operador_giros)     ) and
                                                (grupo =
                                                supervisor_giros) )  or
                                                ((grupo_nuevo =
                                                 supervisor_giros) or
                                                (grupo_nuevo =
                                                 operador_giros) or
                                                (grupo_nuevo =
                                                 interventor_giros) and
                                                (grupo =
                                                 supervisor_gral));
                                   if not area_correcta then
                                       MENSAJE('ERROR','SG=SUP GIROS;
                                                  OG=OP GIROS', 21,
                                                  error_usuario);
                               until area_correcta;

                               with wr_usuarios do
                                  begin
                                  clave_usu := clave;
                                  move(password_nuevo1[1],
                                       password_usu[1],
                                     length(password_nuevo1));
                                  move(grupo_nuevo[1], grupo_usu[1],
                                     length(grupo_nuevo));
                                  move(nombre_nuevo[1], nombre_usu[1],
                                     length(nombre_nuevo));
                                  end;
                               ESCRIBE_USUARIOS(wr_usuarios,status_io);
                               if status_io <> 0 then
                                  DESPLIEGA_ERROR('RUTINA ALTA_CLAVE :
                                                   ESCRIBE_USUARIOS
                                                   ', status_io);
                               MENSAJE('','USUARIO: ' + clave_nuevo + ';
                                          DADO DE ALTA',21,
                                          error_usuario);
                                end;
                             end;
      else
         DESPLIEGA_ERROR ('RUTINA PASSWORD : LEE_USUARIOS ',status_io);
      end;
   status_io := CIERRAS(base);
   if status_io <> ok_io then
      DESPLIEGA_ERROR ('RUTINA PASSWORD : CIERRAS USUARIOS ',status_io);
   ENCABEZADO_PANTALLA('MENU PRINCIPAL');
   END;


PROCEDURE BAJA_CLAVE (grupo : tpo_grupo_usuarios);

   VAR
   clave_baja      : string;
   clave_baja2     : string;
   grupo_baja      : string;
   area_correcta    : boolean;

   BEGIN
   ENCABEZADO_PANTALLA('BAJA DE USUARIOS');
   base := nil;
   status_io := ABRES(base,'USUARIOS',abre_usuarios,modo_recuperacion,
                      cierra_usuarios);
   if status_io <> ok_io then
```

```
      DESPLIEGA_ERROR('RUTINA BAJA_CLAVE:ABRE_USUARIOS ',status_io);
   clave_baja := '      ';
   grupo_baja := '  ';
   presiono_ESC    := false;
   MENSAJE('','     Teclear CLAVE DE BAJA:
           8, aviso_leer);
   LEE_TEXTO(42,9,'',clave_baja,true,presiono_ESC);
   if presiono_ESC then
      begin
      status_io := CIERRAS(base);
      exit;
      end;
   move(clave_baja[1],clave[1],6);
   LEE_USUARIOS(wr_usuarios,clave,status_io);
   case status_io of
      no_existe_llave :
         MENSAJE(' ERROR ','NO EXISTE CLAVE; BAJA CANCELADA',21,
                 error_usuario);
      ok_io     : begin
                  MENSAJE('','   Reteclear CLAVE BAJA :
                                ', 10,aviso_leer);
                  LEE_TEXTO(42,11,'',clave_baja2,false,presiono_ESC);
                  if presiono_ESC then
                     begin
                     status_io := CIERRAS(base);
                     exit;
                     end;
                  if clave_baja <> clave_baja2 then
                     MENSAJE (' ERROR ','CLAVES DIFERENTES; BAJA
                                 CANCELADA', 21, error_usuario)
                  else
                     begin
                     area_correcta := false;
                     repeat
                        MENSAJE('','Teclear AREA del USUARIO : ', 16,
                                  aviso_leer);
                        LEE_TEXTO(42,17,'',grupo_baja,true,presiono_ESC);
                        if presiono_ESC then
                           begin
                           status_io := CIERRAS(base);
                           exit;
                           end;
                         area_correcta := (((grupo_baja =
                                              supervisor_giros) or
                                           (grupo_baja =
                                              operador_giros)) and
                                          (grupo = supervisor_giros))
                                            or ((grupo_baja =
                                              supervisor_giros) or
                                           (grupo_baja = operador_giros)
                                              or (grupo_baja =
                                              interventor_giros) and
                                              (grupo =
                                              supervisor_gral));
                        if not area_correcta then
                           MENSAJE('ERROR','SG=SUP GIROS; OG=OP GIROS',
                                    21, error_usuario);
                     until area_correcta;
                     BORRA_USUARIOS(status_io);
                     if status_io <> 0 then
                        DESPLIEGA_ERROR('RUTINA BAJA_CLAVE :
                                         BORRA_USUARIOS ',status_io);
                     MENSAJE('','USUARIO: ' + clave_baja + '; DADO DE
                                BAJA',21,error_usuario);
                     end;
                  end;
else
 DESPLIEGA_ERROR ('RUTINA BAJA_USUARIO : LEE_USUARIOS ',status_io);
end;
status_io := CIERRAS(base);
if status_io <> ok_io then
DESPLIEGA_ERROR ('RUTINA BAJA_USUARIOS : CIERRAS USUARIOS ',status_io);
ENCABEZADO_PANTALLA('MENU PRINCIPAL');
END;
END.{END UPASSWORD}
```

Programa con declaraciones para el SISGIRO:

```
UNIT UPROGIR;
{ archivo con constantes para el archivo de control del sistema }

INTERFACE

CONST

    fecha_control = 0001;
    fecha_sig_control = 0002;
  {Constantes del menu}

    inicio_dia = 1000;
        cifras_apertura = 1100;

    captura_op_giros = 2000;
        altas_giros = 2100;
        bajas_giros = 2200;
        cambios_giros = 2300;
        captura_automatica_giros = 2400;

    generacion_giros = 3000;

    fin_dia = 4000;
        formatos_con  = 4100;
        act_concilia  = 4200;
        cifras_cierre = 4300;

    informes = 5000;
        reporte_generacion = 5100;
        reporte_generados  = 5200;
        reporte_captura    = 5300;

    utilerias = 6000;
        cambio_password = 6100;
        menu_usuarios   = 6200;
          altas_usuarios = 6210;
          bajas_usuarios = 6220;
        mantenimiento   = 6300;
            manto_benef = 6310;
                altas_benef   = 6311;
                bajas_benef   = 6312;
                cambios_benef = 6313;
                reporte_benef = 6314;
            manto_corresp = 6320;
                altas_corresp   = 6321;
                bajas_corresp   = 6322;
                cambios_corresp = 6323;
                reporte_corresp = 6324;
            manto_ctahabte = 6330;
                altas_ctahabte   = 6331;
                bajas_ctahabte   = 6332;
                cambios_ctahabte = 6333;
                reporte_ctahabte = 6334;
            manto_divisas = 6340;
                altas_divisas = 6341;
                bajas_divisas = 6342;
                cambios_divisas = 6343;
                reporte_divisas = 6344;
        actualiza_control_giros  = 6400;
            cancela_giros = 6410;
            cifras_control = 6420;

    salir = 7000;
    proceso_dummy = 0003;


    {Constantes de num_giro}
     p_giro_inicio_citi   = 0001;
     u_giro_inicio_citi   = 0002;
     p_giro_usado_citi    = 0003;
     u_giro_usado_citi    = 0004;
```

```
    p_giro_fin_citi       = 0005;
    u_giro_fin_citi       = 0006;
    errores_citi          = 0007;
    sig_num_imprimir_citi= 0008;
    p_giro_inicio_chemical     = 0009;
    u_giro_inicio_chemical     = 0010;
    p_giro_usado_chemical      = 0011;
    u_giro_usado_chemical      = 0012;
    p_giro_fin_chemical        = 0013;
    u_giro_fin_chemical        = 0014;
    errores_chemical           = 0015;
    sig_num_imprimir_chemical = 0016;
    p_giro_inicio_libre    = 0017;
    u_giro_inicio_libre    = 0018;
    p_giro_usado_libre     = 0019;
    u_giro_usado_libre     = 0020;
    p_giro_fin_libre       = 0021;
    u_giro_fin_libre       = 0022;
    errores_libre          = 0023;
    sig_num_imprimir_libre= 0024;

IMPLEMENTATION

END. {UPROGIRO}
```

Programa con rutinas generales del SISGIRO:

```
UNIT URUTGIRO;
{ rutinas generales del sistema }

  INTERFACE
     USES
       TPCRT,
       TPENTRY,
       TPDATE,
       TPMENU,
       UDCLGIRO;

    PROCEDURE DEFINE_VENTANA_CAPTURA (var pantalla_captura : ESRecord;
                                      xmin, ymin, xmax, ymax : byte);

    PROCEDURE LINEA_GUIA(texto : string);


    PROCEDURE ABRE_ARCHIVOS(proceso : proceso_enum; tipo_mov :char);

    FUNCTION CONFIRMA(texto : string): boolean;

    FUNCTION TPOFECHA_FECHA(fecha : tpo_fecha): DateString;

    FUNCTION FECHA_TPOFECHAST(f_fuente: DateString): string;

    PROCEDURE LIMPIA_VENTANA;

    FUNCTION FECHA_TEXTO(fecha : tpo_fecha) : string;

    PROCEDURE DESHABILITA (var menu_pr : menu);

    PROCEDURE BLOQUEA (    proceso     : longint;
                           texto       : string);

    PROCEDURE DESBLOQUEA (proceso     : longint;
                          texto       : string);

    FUNCTION BLOQUEADO(proceso : longint;
                       texto   : string) : boolean;

    FUNCTION REALIZADO (proceso : longint;
                        texto   : string) : boolean;

     PROCEDURE PIDE_FECHA_SISTEMA;

     PROCEDURE SIGUIENTE_FECHA(fecha_in : tpo_fecha;
                               dias, meses, anios : integer;
                               var fecha_sig : tpo_fecha);
   IMPLEMENTATION

   USES
      COLORDEF,
      TPSTRING,
      TPEDIT,
      UCTAHAB,
      UCONTROL,
      UCON_USU,
      UDIVISA,
      UGRALGIR,
      UGIROS,
      UCORRESP,
      UBEN_GIR,
      ULECTURA,
      UMSJGIRO,
      UNUMGIR,
      UPROGIR;

   CONST
      meses3 : array [1..12] of string[3] =
             ('ENE', 'FEB', 'MAR', 'ABR', 'MAY', 'JUN', 'JUL', 'AGO', 'SEP',
              'OCT', 'NOV', 'DIC');
```

```
PROCEDURE COLORES (color_fore, color_back : byte);
   BEGIN
   textcolor(color_fore);
   textbackground(color_back);
   END;


PROCEDURE XYWrite (x, y: byte; s: string);
   BEGIN
   GoToXY(x,y);
   write(s);
   END;

PROCEDURE DEFINE_VENTANA_CAPTURA (var pantalla_captura : ESRecord;
                                  xmin, ymin, xmax, ymax : byte );

   begin
   SetFrameChars(#186, #205, #188, #187, #200, #201);
   SetEntryWindow(pantalla_captura,
                  xmin, ymin, xmax, ymax,
                  true, WhiteonBlack, YellowonBlue);
   SetPromptAttr(WhiteonBlack);
   SetFieldAttr(BlackonLtGray);
   SetStringAttr(BlackonLtGray);
   SetCtrlAttr(BlackonLtGray);
   end;  {END DEFINE_VENTANA_CAPTURA}

PROCEDURE LINEA_GUIA(texto : string);
   begin
   FastWrite(pad(texto,80),25, 1, BlackonLtgray);
   end; {END LINE_GUIA}


PROCEDURE ABRE_ARCHIVOS(proceso : proceso_enum; tipo_mov :char);
  VAR
    status_io : integer;

  begin
  base:= nil;
  case proceso of
     Giros:
        begin
        {1}
        status_io := ABRES(base,'GIROS',abre_giros, modo_recuperacion,
                           cierra_giros);
        if status_io <> ok_io then
           DESPLIEGA_ERROR('ABRE_ARCHIVOS. ABRE_GIROS', status_io);
        {2}
        status_io := ABRES(base,'BENEFGIR',abre_benef, modo_lectura,
                           cierra_benef);
        if status_io <> ok_io then
           DESPLIEGA_ERROR('ABRE_ARCHIVOS. ABRE_BENEFICIARIOS',
                           status_io);
        {3}
        status_io := ABRES(base,'CORRESP',abre_corresp, modo_lectura,
                           cierra_corresp);
        if status_io <> ok_io then
           DESPLIEGA_ERROR('ABRE_ARCHIVOS. ABRE_CORRESPONSALES',
                           status_io);
        {4}
        status_io := ABRES(base,'DIVISAS',abre_divisa, modo_lectura,
                           cierra_divisa);
        if status_io <> ok_io then
           DESPLIEGA_ERROR('ABRE_ARCHIVOS. ABRE_DIVISAS', status_io);
        {5}
        status_io := ABRES(base,'CTAHABTE', abre_ctahabte, modo_lectura,
                           cierra_ctahabte);
        if status_io <> ok_io then
           DESPLIEGA_ERROR('ABRE_ARCHIVOS. ABRE_CUENTAHABIENTES',
                           status_io);
        end;
     Corresponsales:
        begin
        {1}
```

```
            status_io := ABRES(base,'CORRESP', abre_corresp,
                               modo_recuperacion, cierra_corresp);
            if status_io <> ok_io then
               DESPLIEGA_ERROR('ABRE_ARCHIVOS. ABRE_CORRESPONSALES',
                               status_io);
            end;
         Beneficiarios:
            begin
            {1}
            status_io := ABRES(base,'BENEF', abre_benef, modo_recuperacion,
                               cierra_benef);
            if status_io <> ok_io then
               DESPLIEGA_ERROR('ABRE_ARCHIVOS. ABRE_BENEFICIARIOS',
                               status_io);
            end;
         Cuentahabtes:
            begin
            {1}
            status_io := ABRES(base,'CTAHABTE', abre_ctahabte,
                               modo_recuperacion, cierra_ctahabte);
            if status_io <> ok_io then
               DESPLIEGA_ERROR('ABRE_ARCHIVOS. ABRE_CUENTAHABIENTES',
                               status_io);
            {2}
            status_io := ABRES(base,'DIVISAS', abre_divisa,
                               modo_recuperacion, cierra_divisa);
            if status_io <> ok_io then
               DESPLIEGA_ERROR('ABRE_ARCHIVOS. ABRE_CUENTAHABIENTES',
                               status_io);
            end;
         Divisas:
            begin
            {1}
            status_io := ABRES(base,'DIVISAS', abre_divisa,
                               modo_recuperacion, cierra_divisa);
            if status_io <> ok_io then
               DESPLIEGA_ERROR('ABRE_ARCHIVOS.ABRE_DIVISAS',status_io);
            end;
         end;{end case}
      end; {END ABRE_ARCHIVOS}

   FUNCTION CONFIRMA(texto :string) : boolean;
   VAR
      c: char;
   BEGIN
   FastWrite (Pad('',80), 25, 1, BlackOnLtGray);
   ReadCharacter(texto, 25, 1, BlackOnLtGray, ['s', 'S', 'n','N'], c );
   CONFIRMA := (c = 's') or (c = 'S');
   FastWrite (Pad('',80), 25, 1, WhiteonBlack);
   END; { CONFIRMA}

   FUNCTION TPOFECHA_FECHA(fecha : tpo_fecha) : DateString;
     VAR
        fecha_aux : DateString;
     begin
     fecha_aux := '  /  /  ';
     move(fecha[7], fecha_aux[1], 2);
     move(fecha[5], fecha_aux[4], 2);
     move(fecha[3], fecha_aux[7], 2);
     TPOFECHA_FECHA := fecha_aux;
     end; {END TPOFECHA_FECHA}


   FUNCTION FECHA_TPOFECHAST (f_fuente: DateString): string;

     begin
     FECHA_TPOFECHAST := copy(TodayString('yyyy/mm/dd'), 1, 4) +
                         copy(f_fuente, 4, 2) + copy(f_fuente, 1, 2);
     end; {END FECHA_TPOFECHAST}

   PROCEDURE LIMPIA_VENTANA;
     begin
     window(2, 7, 79, 23);
     clrscr;
     end; {end limpia_ventana}
```

```
FUNCTION FECHA_TEXTO(fecha : tpo_fecha) : string;
VAR
  status_io,
  mes : integer;

begin
val(copy(fecha, 5, 2), mes, status_io);
if status_io = ok_io then
   FECHA_TEXTO := copy(fecha, 7, 2) + '-' + meses3[mes] + '-' +
                  copy(fecha, 1, 4)
else
   FECHA_TEXTO := '**********';
end; {end fecha_texto}

PROCEDURE DESHABILITA (var menu_pr : menu);

   VAR
   clave : tpo_operador;
   grupo : tpo_grupo_usuarios;

   BEGIN

   LEE_CLAVE_USUARIO (clave, grupo);

   if grupo = supervisor_giros then
      begin
      DisableMenuItem(menu_pr,captura_op_giros);
      DisableMenuItem(menu_pr, mantenimiento);
      DisableMenuItem(menu_pr, cifras_cierre);
      end;
   if grupo = operador_giros then
      begin
      DisableMenuItem(menu_pr, generacion_giros);
      DisableMenuItem(menu_pr, menu_usuarios);
      DisableMenuItem(menu_pr, actualiza_control_giros);
      DisableMenuItem(menu_pr, cifras_cierre);
      end;
   if grupo = interventor_giros then
      begin
      DisableMenuItem(menu_pr, inicio_dia);
      DisableMenuItem(menu_pr, captura_op_giros);
      DisableMenuItem(menu_pr, generacion_giros);
      DisableMenuItem(menu_pr, informes);
      DisableMenuItem(menu_pr, act_concilia);
      DisableMenuItem(menu_pr, menu_usuarios);
      DisableMenuItem(menu_pr, mantenimiento);
      DisableMenuItem(menu_pr, actualiza_control_giros);
      end;
   END; {end deshabilita}


   PROCEDURE BLOQUEA (    proceso     : longint;
                          texto       : string);
   VAR
   proceso_aux,
   status_io,
   estado_ant : integer;

   BEGIN
   LEE_STATUS_PROCESO (proceso, estado_ant, status_io);
   if status_io <> ok_io then
      DESPLIEGA_ERROR(texto,status_io);
   ACTUALIZA_STATUS_PROCESO (proceso, estado_ant, status_io);
   if status_io <> ok_io then
      DESPLIEGA_ERROR(texto,status_io);
   ACTUALIZA_STATUS_PROCESO (proceso, bloqueado_ctr, status_io);
   if status_io <> ok_io then
      DESPLIEGA_ERROR(texto,status_io);
   END;


   PROCEDURE DESBLOQUEA (proceso     : longint;
                         texto       : string);
   VAR
```

```
    estado_proc,
    status_io      :  integer;

    BEGIN
    LEE_STATUS_PROCESO (proceso, estado_proc, status_io);
    if status_io <> ok_io then
       DESPLIEGA_ERROR(texto,status_io);
    ACTUALIZA_STATUS_PROCESO (proceso, no_realizado_ctr, status_io);
    if status_io <> ok_io then
       DESPLIEGA_ERROR(texto,status_io)
    else
       MENSAJE ('ERROR','EL PROCESO NO ESTA BLOQUEADO',21,error_usuario);
    END;


    FUNCTION BLOQUEADO(proceso : longint;
                       texto   : string) : boolean;
    VAR
       status_io,
       estado : integer;

    BEGIN

    LEE_STATUS_PROCESO(proceso,estado,status_io);
    if status_io <> ok_io then
       DESPLIEGA_ERROR(texto,status_io);
    if estado = bloqueado_ctr then
       BLOQUEADO := true
    else
       BLOQUEADO := false;
    END;


    FUNCTION REALIZADO (proceso : longint;
                        texto   : string) : boolean;
    VAR
       status_io,
       estado : integer;

    BEGIN

    LEE_STATUS_PROCESO(proceso,estado,status_io);
    if status_io <> ok_io then
       DESPLIEGA_ERROR(texto,status_io);
    if estado = realizado_ctr then
       REALIZADO := true
    else
       REALIZADO := false;

    END;

PROCEDURE INICIALIZA_CONTROL(fecha_sistema : tpo_fecha;
                             fecha_sig_sistema : tpo_fecha);

    VAR
    wr_control       : r_control;
    wr_num_gir       : r_num_gir;
    status_io        : integer;

    BEGIN
    status_io := ABRES(base, 'CONTROL', abre_control, modo_recuperacion,
                       cierra_control);
    if status_io <> ok_io then
       DESPLIEGA_ERROR('RUTINA INICIALIZA_CONTROL: ABRE_CONTROL',
                       status_io);

    LEE_PRIMER_CONTROL(wr_control, status_io);
    if status_io <> ok_io then
       DESPLIEGA_ERROR('RUTINA INICIALIZA_CONTROL: LEE_PRIMER_CONTROL',
                       status_io);

    while (status_io = ok_io) do
       begin
       if wr_control.proceso_id_ctr <> fecha_control then
          wr_control.status_ctr := no_realizado_ctr;
```

```
      ACTUALIZA_CONTROL(wr_control, status_io);
      if status_io <> ok_io then
         DESPLIEGA_ERROR('INICIALIZA_CONTROL: ACTUALIZA_CONTROL',
                         status_io);
      LEE_SIG_CONTROL(wr_control,status_io);
      if (status_io <> ok_io) and (status_io <> fin_archivo) then
         DESPLIEGA_ERROR('RUTINA INICIALIZA_CONTROL: LEE_SIG_CONTROL ',
                         status_io);
      end;
   {actualiza la fecha del dia, y sig_fecha_habil}
   LEE_CONTROL(wr_control, fecha_control, status_io);
   if status_io <> ok_io then
      DESPLIEGA_ERROR('RUTINA INICIALIZA_CONTROL: LEE_CONTROL' ,
                      status_io);
   wr_control.fecha_ctr := fecha_sistema;
   ACTUALIZA_CONTROL(wr_control, status_io);
   if status_io <> ok_io then
      DESPLIEGA_ERROR('INICIALIZA CONTROL: ACTUALIZA_CONTROL ',
                      status_io);

   LEE_CONTROL(wr_control, fecha_sig_control, status_io);
   if status_io <> ok_io then
      DESPLIEGA_ERROR('RUTINA INICIALIZA_CONTROL: LEE_CONTROL' ,
                      status_io);
   wr_control.fecha_ctr := fecha_sig_sistema;
   ACTUALIZA_CONTROL(wr_control, status_io);
   if status_io <> ok_io then
      DESPLIEGA_ERROR('INICIALIZA CONTROL: ACTUALIZA_CONTROL ',
                      status_io);

   {inicializa el campo de errores en num_giro.dat}

   status_io := ABRES(base, 'NUM_GIRO', abre_num_gir, modo_recuperacion,
                      cierra_num_gir);
   if status_io <> ok_io then
      DESPLIEGA_ERROR('RUTINA INICIALIZA_CONTROL: ABRE_NUMGIR',
                      status_io);
   LEE1_NUM_GIR(wr_num_gir, errores_citi, status_io);
   if status_io <> ok_io then
      DESPLIEGA_ERROR('RUTINA INICIALIZA_CONTROL: LEE1_NUMGIR',
                      status_io);
   wr_num_gir.num_aux_num_gir := 0;
   ACTUALIZA1_NUM_GIR(wr_num_gir, status_io);
   if status_io <> ok_io then
      DESPLIEGA_ERROR('RUTINA INICIALIZA_CONTROL: ACTUALIZA1_NUMGIR',
                      status_io);
   LEE1_NUM_GIR(wr_num_gir, errores_chemical, status_io);
   if status_io <> ok_io then
      DESPLIEGA_ERROR('RUTINA INICIALIZA_CONTROL: LEE1_NUMGIR',
                      status_io);
   wr_num_gir.num_aux_num_gir := 0;
   ACTUALIZA1_NUM_GIR(wr_num_gir, status_io);
   if status_io <> ok_io then
      DESPLIEGA_ERROR('RUTINA INICIALIZA_CONTROL: ACTUALIZA1_NUMGIR',
                      status_io);
   LEE1_NUM_GIR(wr_num_gir, errores_libre, status_io);
   if status_io <> ok_io then
      DESPLIEGA_ERROR('RUTINA INICIALIZA_CONTROL: LEE1_NUMGIR',
                      status_io);
   wr_num_gir.num_aux_num_gir := 0;
   ACTUALIZA1_NUM_GIR(wr_num_gir, status_io);
   if status_io <> ok_io then
      DESPLIEGA_ERROR('RUTINA INICIALIZA_CONTROL: ACTUALIZA1_NUMGIR',
                      status_io);
   status_io := CIERRAS(base);
   if status_io <> ok_io then
    DESPLIEGA_ERROR('RUTINA INICIALIZA_CONTROL: CIERRAS',status_io);
   END;


PROCEDURE PIDE_FECHA_SISTEMA;
   CONST
      Enter = ^M;

   VAR
```

```
    fecha_sistema,
    fecha_usuario,
    fecha_sig_dia_habil,
    fecha_aux      : tpo_fecha;
    fecha_ok,
    habil_ok       : boolean;
    no_intentos    : byte;
    estado_proceso,
    status_io      : integer;
    status_byte    : byte;
    wtu            : integer;
    dummy          : char;


BEGIN
FECHA_HOY(fecha_sistema, status_io);
if status_io <> ok_io then
    DESPLIEGA_ERROR('RUTINA PIDE_FECHA_SIS : FECHA_HOY ',status_io);


fecha_ok    := false;
no_intentos := 0;
repeat
   no_intentos := no_intentos + 1;
   ENCABEZADO_PANTALLA('FECHA DEL SISTEMA');
   MENSAJE('','Teclear FECHA SISTEMA :           ',10,aviso_leer);
   LEE_FECHA(48,11,'',fecha_usuario);
   MENSAJE('','Teclear FECHA SIG. DIA HABIL :          ',
            14,aviso_leer);
   LEE_FECHA(50,15,'',fecha_sig_dia_habil);
   if (fecha_usuario > fecha_sistema) and
      (fecha_sig_dia_habil > fecha_usuario) then
      fecha_ok := true
   else
      if fecha_usuario < fecha_sistema then
         MENSAJE('ERROR','FECHA MENOR A LA DEL SISTEMA',21,
                  error_usuario)
      else
         MENSAJE('ERROR', 'FECHA SIG. ANTERIOR A FECHA SISTEMA', 21,
                  error_usuario);
   if fecha_ok then
      begin
      fastwrite(pad('Fechas Correcta?
                  Enter-Aceptar', 80), 25,1,BlackonLtGray);
      dummy := readkey;
      fecha_ok := (dummy = Enter);
      fastwrite(pad(' ', 80), 25,1,LtGrayonBlack);
      end;
   if REALIZADO(cifras_cierre,'CIFRAS CIERRE ') and fecha_ok then
      begin
      MENSAJE('ESPERAR','INICIALIZANDO CONTROL',21,aviso);
      INICIALIZA_CONTROL(fecha_usuario, fecha_sig_dia_habil);
      fecha_ok := true;
      end
   else
      if not REALIZADO(cifras_cierre, 'CIFRAS CIERRE ') then
         MENSAJE('ERROR','NO SE HA REALIZADO CIFRAS DE CIERRE (DIA
                  ANTERIOR)', 21, error_usuario);
until (fecha_ok) or (no_intentos = 3);
if (no_intentos = 3) and (not fecha_ok) then
   begin
   MENSAJE(' ERROR ','INTENTOS EXCEDIDOS; EJECUCION DEL SISTEMA
            SUSPENDIDA',21,error_usuario);
   halt(3);
   end;
ENCABEZADO_PANTALLA('MENU_PRINCIPAL');
END;

PROCEDURE SIGUIENTE_FECHA(fecha_in : tpo_fecha;
                  dias, meses, anios : integer;
                  var fecha_sig : tpo_fecha);
VAR
   fecha_str : DateString;
   f_juliana2,
   f_juliana1 : Date;
```

```
      string_aux : string;

    begin
    fecha_str := TPOFECHA_FECHA(fecha_in);
    f_juliana1 := DateStringtoDate('dd/mm/yy', fecha_str);
    f_juliana2 := IncDate(f_juliana1, dias, meses, anios);
    if DayString[Dayofweek(f_juliana2)] = 'Saturday'then
       f_juliana2 := IncDate(f_juliana2, 2, 0, 0)
    else
    if DayString[Dayofweek(f_juliana2)] = 'Sunday' then
       f_juliana2 := IncDate(f_juliana2, 1, 0, 0);
    fecha_str  := DatetoDateString('dd/mm/yy', f_juliana2);
    string_aux := FECHA_TPOFECHAST(fecha_str);
    move(string_aux[1], fecha_sig[1], sizeof(tpo_fecha));
    end; {end SIGUIENTE_FECHA}
END. {UNIT URUTGIRO}
```

Programa con funciones para el manejo del archivo de usuarios:

```
{ UNIDAD PARA EL MANEJO DEL ARCHIVO USUARIOS }
{$I-,V-,R-,N+,E+,D+}

UNIT UUSU_GIR;

{ Unidad que contine las llamadas de btrieve del archivo usuarios. }

INTERFACE

   USES
      UDCLGIRO;

   TYPE

      LLAVE_USUARIOS = tpo_operador;

      R_USUARIOS      = RECORD
         clave_usu    : tpo_operador;
         password_usu : tpo_password;
         grupo_usu    : tpo_grupo_usuarios;
         nombre_usu   : tpo_nombre_usuario;
      END;


   FUNCTION ABRE_USUARIOS       (     modo          : char) : integer;
   FUNCTION CIERRA_USUARIOS : INTEGER;
   PROCEDURE ACTUALIZA_USUARIOS (     wr_usuarios   : r_usuarios;
                                      var status_io : integer  );
   PROCEDURE BORRA_USUARIOS     ( var status_io     : integer);
   PROCEDURE ESCRIBE_USUARIOS   (    wr_usuarios    : r_usuarios;
                                  var status_io     : integer );
   PROCEDURE LEE_USUARIOS       ( var wr_usuarios   : r_usuarios;
                                      llave_usu     : llave_usuarios;
                                  var status_io     : integer );


IMPLEMENTATION

   USES
      UBTRIEVE;

   VAR
      bloque_pos_usu    : array [1..128] of byte;
      long_reg_usu      : integer; { Tamanio registro del archivo }


   FUNCTION ABRE_USUARIOS       (     modo          : char) : integer;
         { Abre archivo de usuarios del area internacional }
      VAR
         wr_usuarios        : r_usuarios;
         status_io          : integer;
         nom_arch           : string [50];
         nom_archivo        : array [1..50] of char;
      BEGIN
      nom_arch :=  'f:\giros\datos\usu_giro.dat';
      move (blancos[1], nom_archivo[1], 50);
      move (nom_arch[1], nom_archivo[1], length (nom_arch));
      long_reg_usu := sizeof (wr_usuarios) ;
      if (modo = modo_lectura) then
         status_io := BTRV (abre_archivo, bloque_pos_usu, wr_usuarios,
                            long_reg_usu, nom_archivo, solo_lectura )
      else
      if (modo = modo_recuperacion) then
         status_io := BTRV (abre_archivo, bloque_pos_usu, wr_usuarios,
                            long_reg_usu, nom_archivo, con_recuperacion );

      if (status_io = 20) then   {No esta btrieve}
         write('**********     Falta cargar Btrieve     ********** ');
      abre_usuarios := status_io;
      END;   { ABRE_USUARIOS }
```

```
FUNCTION CIERRA_USUARIOS : INTEGER;
  BEGIN
  cierra_usuarios := BTRV (cierra_archivo, bloque_pos_usu, dummy,
                           dummy, dummy, dummykey );
  END;


PROCEDURE ACTUALIZA_USUARIOS (     wr_usuarios   : r_usuarios;
                               var status_io     : integer  );
  VAR
    llave_usu  :  llave_usuarios ;
  BEGIN
  status_io := BTRV  ( actualiza,    bloque_pos_usu, wr_usuarios,
                       long_reg_usu, llave_usu,      llavel    );
  END;


PROCEDURE BORRA_USUARIOS      ( var status_io     : integer);
  BEGIN
  status_io := BTRV (borra,  bloque_pos_usu, dummy,
                     dummy,  dummy,          llavel  );
  END;


PROCEDURE  ESCRIBE_USUARIOS  (     wr_usuarios   : r_usuarios;
                               var status_io     : integer );
        { Escribe registro de ope_cambios }
  VAR
     llave_usu    : llave_usuarios;
  BEGIN
  status_io := BTRV (escribe,       bloque_pos_usu, wr_usuarios,
                     long_reg_usu,  llave_usu,      llavel);
  END;


PROCEDURE LEE_USUARIOS        ( var wr_usuarios   : r_usuarios;
                                    llave_usu     : llave_usuarios;
                                var status_io     : integer );
  BEGIN
  status_io := BTRV (get_equal,     bloque_pos_usu, wr_usuarios,
                     long_reg_usu,  llave_usu,      llavel );
  END;

END.{UNIT UUSUARIO}
```

Programa con rutinas para validación de datos:

```
{$F+,N+,E+}
UNIT UVALIDA;
{ procedimientos y funciones para validar captura }
    INTERFACE
    USES
       TPCRT,
       TPENTRY,
       UDCLGIRO;


    CONST
       error_vacio          = 10; {el campo es obligatorio}
       error_existe         = 11; {el registro ya existe}
       error_no_existe      = 12; {el registro no existe}
       error_folio_invalido = 13; {folio invalido}
       error_folio_proc     = 14; {el folio ya fue procesado}
       error_divisa         = 15; {divisa invalida}
       error_vacio_st          : string[23] = 'EL CAMPO ES OBLIGATORIO';
       error_ex_folio_st       : string[18] = 'EL FOLIO YA EXISTE';
       error_no_folio_st       : string[18] = 'EL FOLIO NO EXISTE';
       error_folio_invalido_st : string[14] = 'FOLIO INVALIDO';
       error_folio_proc_st     : string[44] = 'FOLIO PROCESADO. NO
                                              MODIFICABLE O CANCELABLE';
       error_divisa_st         : string[15] = 'DIVISA INVALIDA';
       error_ex_divisa_st      : string[19] = 'LA DIVISA YA EXISTE';
       error_no_divisa_st      : string[18] = 'DIVISA INEXISTENTE';
       error_no_benef_st       : string[24] = 'BENEFICIARIO INEXISTENTE';
       error_ex_benef_st       : string[22] = 'EL BENEFICIARIO EXISTE';
       error_no_corr_st        : string[24] = 'CORRESPONSAL INEXISTENTE';
       error_ex_corr_st        : string[22] = 'EL CORRESPONSAL EXISTE';
       error_ex_ctahabte_st    : string[24] = 'EL CUENTAHABIENTE EXISTE';
       error_no_ctahabte_st    : string[26] = 'CUENTAHABIENTE
                                                  INEXISTENTE';
       error_monto_st          : string[14] = 'MONTO INVALIDO';
       error_fecha_invalida_st : string[14] = 'FECHA INVALIDA';

    VAR
     wr_corr_cap     : r_corr_cap;
     wr_benef_cap    : r_benef_cap;
     wr_ctahabte_cap : r_ctahabte_cap;
     wr_giros_cap    : r_giros_cap;
     wr_divisa_cap   : r_divisa_cap;

  PROCEDURE ErrorHandler(var ESR : ESrecord; Code: Byte; Msg : string);

  FUNCTION VALIDA_FOLIO(var FR : FieldRec;
                        var ErrCode : byte;
                        var ErrorSt : StringPtr): boolean;

  FUNCTION EXISTE_FOLIO(var FR : FieldRec;
                        var ErrCode: byte;
                        var ErrorSt : StringPtr): boolean;

  PROCEDURE OBTIENE_FOLIO(var ESR :ESRecord);

  PROCEDURE SIG_FOLIO(var ESR :ESRecord);

  FUNCTION VALIDA_BENEFICIARIO(var FR : FieldRec;
                               var ErrCode : byte;
                               var ErrorSt : stringptr): boolean;

  FUNCTION EXISTE_BENEFICIARIO(var FR : FieldRec;
                               var ErrCode : byte;
                               var ErrorSt : stringPtr) : boolean;

  PROCEDURE OBTIENE_BENEF(var ESR :ESRecord);

  FUNCTION VALIDA_CORRESPONSAL(var FR : FieldRec;
                               var ErrCode : byte;
                               var ErrorSt : stringptr): boolean;
```

```
FUNCTION EXISTE_CORRESPONSAL(var FR: FieldRec;
                             var ErrCode: byte;
                             var ErrorSt: StringPtr): boolean;

PROCEDURE OBTIENE_CORRESP(var ESR :ESRecord);

FUNCTION VALIDA_CTAHABTE(var FR : FieldRec;
                         var ErrCode : byte;
                         var ErrorSt : stringptr): boolean;

FUNCTION EXISTE_CTAHABTE(var FR: FieldRec;
                         var ErrCode: byte;
                         var ErrorSt: StringPtr): boolean;

PROCEDURE OBTIENE_CTAHABTE(var ESR :ESRecord);

FUNCTION VALIDA_DIVISA(var FR : FieldRec;
                       var ErrCode : byte;
                       var ErrorSt : stringptr): boolean;

FUNCTION EXISTE_DIVISA(var FR: FieldRec;
                       var ErrCode: byte;
                       var ErrorSt: StringPtr): boolean;

PROCEDURE OBTIENE_DIVISA(var ESR :ESRecord);

FUNCTION VALIDA_CUENTA(var FR : FieldRec;
                       var ErrCode : byte;
                       var ErrorSt : stringptr): boolean;


  IMPLEMENTATION

  USES
     TPSTRING,
     UBEN_GIR,
     UCORRESP,
     UCTAHAB,
     UGRALGIR,
     UGIROS,
     UMSJGIRO,
     URUTGIRO,
     UDIVISA;

PROCEDURE ErrorHandler(var ESR : ESrecord; Code: Byte; Msg : string);
   begin
   case Code of
      RangeError : Msg := error_monto_st;
      ReqFldError: Msg := error_vacio_st;
      FormatError: Msg := error_fecha_invalida_st;
      end;
   mensaje('ERROR', Msg, 21, error_usuario);
   end; {end ErrorHandler}


PROCEDURE CHECA_REGISTRO(existe : boolean;
                         proceso : proceso_enum;
                         var registro  : FieldRec;
                         var error_cod : byte;
                         var error_st  : StringPtr;
                         var estado    : boolean);
VAR
   wr_giros         : r_giros;
   llave_giros_ant,
   llave_giros      : llavel_giros;
   wr_benef         : r_benefgir;
   llave_benef      : llavel_ben;
   wr_corresponsal  : r_corresp;
   llave_corresp    : llavel_corr;
   wr_ctahabte      : r_ctahabte;
   llave_ctahabte   : llavel_ctahabte;
   wr_divisa        : r_divisa;
   llave_divisa     : llavel_div;
   status_io,
```

```
    llaveSt_long     : integer;
    llave_st         : string;
    ceros            : string[8];
begin
  estado := off;
  with registro do
    begin
    llave_st := Trim(EditSt^);
    llavest_long := length(llave_st);
    if llave_st <> copy(blancos, 1, sizeof(tpo_corresponsal)) then
       begin
       case proceso of
          Giros:
             begin
             ceros := '00000000';
             move(blancos[1], wr_giros, sizeof(r_giros));
             move(blancos[1], llave_giros, sizeof(llavel_giros));
             move(ceros[1], llave_giros.folio_girk[1],
                   sizeof(tpo_folio));
             move(llave_st[1],llave_giros.folio_girk[sizeof(tpo_folio) -
                   llavest_long +1],llavest_long);
             llave_giros_ant := llave_giros;
             LEE1_GE_GIROS(wr_giros, llave_giros, status_io);
             end;
          Beneficiarios :
             begin
             move(blancos[1], llave_benef[1], sizeof(tpo_corresponsal));
             move(llave_st[1], llave_benef[1], llavest_long);
             LEE_BENEF(wr_benef, llave_benef, status_io);
             end;
          Corresponsales:
             begin
             move(blancos[1], llave_corresp[1],
                   sizeof(tpo_corresponsal));
             move(llave_st[1], llave_corresp[1], llavest_long);
             LEE_CORRESP(wr_corresponsal, llave_corresp, status_io);
             end;
          Cuentahabtes :
             begin
             move(blancos[1], llave_ctahabte[1],
                   sizeof(tpo_corresponsal));
             move(llave_st[1], llave_ctahabte[1], llavest_long);
             LEE_CTAHABTE(wr_ctahabte, llave_ctahabte, status_io);
             end;
          Divisas:
             begin
             move(blancos[1], llave_divisa[1], divisa_long);
             move(llave_st[1], llave_divisa[1], llavest_long);
             LEE_DIVISA(wr_divisa, llave_divisa, status_io);
             end;
       end; {end case proceso}
       case status_io of
          ok_io : if existe then
                     begin
                     if (proceso = Giros) and
                        (wr_giros.folio_gir = llave_giros.folio_girk) and
                        (wr_giros.estado_gir <> negativo) and
                        (wr_giros.estado_gir <> blancos[1])
                        then
                        begin
                        error_cod := error_folio_proc;
                        error_st := @error_folio_proc_st;
                        end
                     else
                        if (proceso = Giros) then
                          begin
                          if llave_giros.folio_girk = ceros then
                             begin
                             error_cod := error_folio_invalido;
                             error_st := @error_folio_invalido_st;
                             end
                          else
                             begin
                             if llave_giros_ant.folio_girk <>
                                   llave_giros.folio_girk
```

```
                                then
                                  begin
                                  error_cod := error_no_existe;
                                  error_st := @error_no_folio_st;
                                  end
                                else
                                  estado := on;
                                end;
                             end
                          else
                             estado := on;
                       end
                    else
                       begin
                       error_cod := error_existe;
                       case proceso of
                          Giros :
                              error_st := @error_ex_folio_st;
                          Beneficiarios:
                              error_st := @error_ex_benef_st;
                          Corresponsales:
                              error_st := @error_ex_corr_st;
                          Cuentahabtes:
                              error_st := @error_ex_ctahabte_st;
                          Divisas:
                              error_st := @error_ex_divisa_st;
                          end; {end case}
                       end;
              no_existe_llave,
              fin_archivo : if existe then
                             begin
                             error_cod := error_no_existe;
                             case proceso of
                               Giros :
                                   begin
                                   if llave_giros.folio_girk = ceros then
                                      error_st := @error_folio_invalido_st
                                   else
                                      error_st := @error_no_folio_st;
                                   end;
                               Beneficiarios:
                                   error_st := @error_no_benef_st;
                               Corresponsales:
                                   error_st := @error_no_corr_st;
                               Cuentahabtes:
                                   error_st := @error_no_ctahabte_st;
                               Divisas:
                                   error_st := @error_no_divisa_st;
                               end;{end case}
                             end
                          else
                             begin
                             if (proceso = giros) and
                                (llave_giros.folio_girk = ceros) then
                                begin
                                error_cod := error_folio_invalido;
                                error_st := @error_folio_invalido_st;
                                end
                             else
                                estado := on;
                             end
              else
                 DESPLIEGA_ERROR('CHECA_REGISTRO.LEE_REGISTRO', status_io);
              end; {end case}
           end
        else
           begin
           error_cod := error_vacio;
           error_st := @error_vacio_st;
           end;
        end; {end with}
     end; {END CHECA_REGISTRO}
```

```
FUNCTION VALIDA_FOLIO(var FR : FieldRec;
                      var ErrCode : byte;
                      var ErrorSt : stringptr): boolean;
VAR
  resultado : boolean;
begin
   valida_folio := off;
   CHECA_REGISTRO(on, Giros, FR, ErrCode, ErrorSt, resultado);
   valida_folio := resultado;
end; {END VALIDA_FOLIO}


FUNCTION EXISTE_FOLIO(var FR : FieldRec;
                      var ErrCode : byte;
                      var ErrorSt : stringptr): boolean;
VAR
  resultado : boolean;
begin
   existe_folio := off;
   CHECA_REGISTRO(off, Giros, FR, ErrCode, ErrorSt, resultado);
   existe_folio := resultado;
end; {END EXISTE_FOLIO}


PROCEDURE OBTIENE_FOLIO(var ESR :ESRecord);

VAR
ceros          : string[8];
wr_giros       : r_giros;
llave_giros    : llavel_giros;
status_io      : integer;
string_aux     : string;
begin
with ESR do
   begin
   if (CurrentID = 0) and
      (Trim(CurrentField^.EDitSt^) <> '') then
      begin
      ceros := '00000000';
      ChangeProtection(ESR, 0, on);
      string_aux := Trim(CurrentField^.EditSt^);
      move(blancos[1], llave_giros, sizeof(llavel_giros));
      move(ceros[1], llave_giros.folio_girk[1], sizeof(tpo_folio));
      move(string_aux[1],
          llave_giros.folio_girk[sizeof(tpo_folio) - length(string_aux) +
                                 1], length(string_aux));
      LEE1_GE_GIROS(wr_giros, llave_giros, status_io);
      if status_io <> ok_io then
         DESPLIEGA_ERROR('OBTEN_GIROS.LEE1_GE_GIROS', status_io);
      with wr_giros, wr_giros_cap do
         begin
         string_aux := copy(blancos, 1, sizeof(tpo_fecha));
         fecha_gir_cap := TPOFECHA_FECHA(fecha_gir);
         move(divisa_gir[1], string_aux[1], divisa_long);
         divisa_gir_cap := string_aux;
         string_aux := copy(blancos, 1, sizeof(tpo_corresponsal));
         move(corresponsal_gir[1], string_aux[1],
              sizeof(tpo_corresponsal));
         corresponsal_gir_cap := string_aux;
         monto_gir_cap := monto_gir;
         string_aux := copy(blancos, 1, sizeof(tpo_beneficiario));
         move(beneficiario_gir[1], string_aux[1],
              sizeof(tpo_beneficiario150));
         beneficiario_gir_cap := string_aux;
         string_aux := copy(blancos, 1, sizeof(tpo_cuenta));
         move(cuenta_cargo_gir[1], string_aux[1], sizeof(tpo_cuenta));
         cuenta_cargo_gir_cap := string_aux;
         move(cuentahabte_gir[1],cuentahabte_gir_cap[1],
              sizeof(cuentahabte_gir));
         end; {end with}
      DrawEditScreen(ESR);
      end;
   end;{end with}
```

```
end; {END OBTIENE FOLIO}


PROCEDURE SIG_FOLIO(var ESR :ESRecord);

VAR
wr_giros       : r_giros;
llave_giros    : llave1_giros;
num_folio,
status_io      : integer;
string_aux     : string;
begin
with ESR do
   begin
   if (CurrentID = 0) then
      begin
      LEE1_ULT_GIROS(wr_giros, llave_giros, status_io);
      if (status_io <> ok_io) and (status_io <> fin_archivo) then
         DESPLIEGA_ERROR('OBTEN_GIROS.LEE1_GE_GIROS', status_io);
      with wr_giros, wr_giros_cap do
         begin
         if status_io = fin_archivo then
            folio_gir_cap := '         1'
         else
            begin
            string_aux := folio_gir;
            if str2int(string_aux, num_folio) then
               begin
               inc(num_folio);
               string_aux := Long2Str(num_folio);
               CurrentField^.EditSt^ := LeftpadCh(string_aux,' ', 8);
               folio_gir_cap := LeftpadCh(string_aux, ' ', 8);
               end
            else
               DESPLIEGA_ERROR('SIG.FOLIO', 999);
            end;
         end;
      DrawEditScreen(ESR);
      end;
   end;{end with}

end; {END SIG. FOLIO}


FUNCTION VALIDA_BENEFICIARIO(var FR : FieldRec;
                             var ErrCode : byte;
                             var ErrorSt : stringptr): boolean;
VAR
  resultado : boolean;
begin
  valida_beneficiario:= off;
  if length(Trim(Fr.EditSt^)) <= sizeof(tpo_corresponsal) then
     begin
     CHECA_REGISTRO(on, Beneficiarios, FR, ErrCode, ErrorSt, resultado);
     valida_beneficiario:= resultado;
     end
  else
     valida_beneficiario := on;
 end; {END VALIDA_BENEFICIARIO}

FUNCTION EXISTE_BENEFICIARIO(var FR: FieldRec;
                             var ErrCode: byte;
                             var ErrorSt: StringPtr): boolean;
VAR
  resultado : boolean;
begin
   existe_beneficiario:= off;
   CHECA_REGISTRO(off, Beneficiarios, FR, ErrCode, ErrorSt, resultado);
   existe_beneficiario:= resultado;
end; {END EXISTE_BENEFICIARIO}


PROCEDURE OBTIENE_BENEF(var ESR :ESRecord);
VAR
wr_benef       : r_benefgir;
```

```
llave_benef    : llavel_ben;
status_io      : integer;
string_aux     : string;

begin
with ESR do
   begin
   if (CurrentID = 0) and
      (CurrentField^.EditSt^ <> copy (blancos, 1,
                                      sizeof(tpo_corresponsal)))
   then
      begin
      ChangeProtection(ESR, 0, on);
      string_aux := CurrentField^.EditSt^;
      move(blancos[1], llave_benef[1], sizeof(tpo_corresponsal));
      move(string_aux[1], llave_benef[1], length(string_aux));
      LEE_BENEF(wr_benef, llave_benef, status_io);
      if status_io <> ok_io then
         DESPLIEGA_ERROR('OBTEN_BENEFICIARIO.LEE_BENEF', status_io);
      with wr_benef, wr_benef_cap do
         begin
         string_aux := copy(blancos, 1, sizeof(nombre_ben));
         move(nombre_ben[1], string_aux[1], sizeof(nombre_ben));
         nombre_benef_cap := string_aux;
         string_aux := copy(blancos, 1, sizeof(extra_ben));
         monto_benef_cap:= monto_ben;
         move(extra_ben[1], string_aux[1], sizeof(extra_ben));
         extra_benef_cap:= string_aux;
         end;
      DrawEditScreen(ESR);
      end;
   end;{end with}
end; {END OBTIENE BENEFICIARIO}


FUNCTION VALIDA_CORRESPONSAL(var FR : FieldRec;
                             var ErrCode : byte;
                             var ErrorSt : stringptr): boolean;
VAR
  resultado : boolean;
begin
  valida_corresponsal := off;
  CHECA_REGISTRO(on, Corresponsales, FR, ErrCode, ErrorSt, resultado);
  valida_corresponsal := resultado;
 end; {END VALIDA_CORRESPONSAL}

FUNCTION EXISTE_CORRESPONSAL(var FR: FieldRec;
                             var ErrCode: byte;
                             var ErrorSt: StringPtr): boolean;
VAR
  resultado : boolean;
begin
   existe_corresponsal := off;
   CHECA_REGISTRO(off, Corresponsales, FR, ErrCode, ErrorSt, resultado);
   existe_corresponsal := resultado;
end; {END EXISTE_CORRESPONSAL}


PROCEDURE OBTIENE_CORRESP(var ESR :ESRecord);

VAR
wr_corresp     : r_corresp;
llave_corresp  : llavel_corr;
status_io      : integer;
string_aux     : string;
begin
with ESR do
   begin
   if (CurrentID = 0) and
      (CurrentField^.EditSt^ <> copy (blancos, 1,
                                      sizeof(tpo_corresponsal)))
   then
      begin
      ChangeProtection(ESR, 0, on);
```

```
      string_aux := CurrentField^.EditSt^;
      move(blancos[1], llave_corresp[1], sizeof(tpo_corresponsal));
      move(string_aux[1], llave_corresp[1], length(string_aux));
      LEE_CORRESP(wr_corresp, llave_corresp, status_io);
      if status_io <> ok_io then
         DESPLIEGA_ERROR('OBTEN_CORRESPONSAL.LEE_CORRESP', status_io);
      with wr_corresp, wr_corr_cap do
         begin
         string_aux := copy(blancos, 1, sizeof(nombre_corr));
         move(nombre_corr[1], string_aux[1], sizeof(nombre_corr));
         nombre_corr_cap := string_aux;
         string_aux := copy(blancos, 1, sizeof(direccion_corr));
         move(direccion_corr[1], string_aux[1], sizeof(nombre_corr));
         direccion_corr_cap:= string_aux;
         string_aux := copy(blancos, 1, sizeof(tpo_cuenta));
         move(cuenta_corr[1], string_aux[1], sizeof(tpo_cuenta));
         cuenta_corr_cap := string_aux;
         end;
      DrawEditScreen(ESR);
      end;
   end;{end with}
end; {END OBTIENE CORRESP}


FUNCTION VALIDA_CTAHABTE(var FR : FieldRec;
                         var ErrCode : byte;
                         var ErrorSt : stringptr): boolean;
VAR
  resultado : boolean;
begin
  valida_ctahabte:= off;
  CHECA_REGISTRO(on, Cuentahabtes, FR, ErrCode, ErrorSt, resultado);
  valida_ctahabte:= resultado;
end; {END VALIDA_CTAHABTE}

FUNCTION EXISTE_CTAHABTE(var FR: FieldRec;
                         var ErrCode: byte;
                         var ErrorSt: StringPtr): boolean;

VAR
  resultado : boolean;
begin
  existe_ctahabte:= off;
  CHECA_REGISTRO(off, Cuentahabtes, FR, ErrCode, ErrorSt, resultado);
  existe_ctahabte:= resultado;
end; {END VALIDA_CTAHABTE}

PROCEDURE OBTIENE_CTAHABTE(var ESR :ESRecord);

VAR
wr_ctahabte    : r_ctahabte;
llave_ctahabte : llave1_ctahabte;
status_io      : integer;
string_aux     : string;

begin
with ESR do
   begin
   if (CurrentID = 0) and
      (CurrentField^.EditSt^ <> copy (blancos, 1,
                                      sizeof(tpo_corresponsal)))
   then
      begin
      ChangeProtection(ESR, 0, on);
      string_aux := CurrentField^.EditSt^;
      move(blancos[1], llave_ctahabte[1], sizeof(tpo_corresponsal));
      move(string_aux[1], llave_ctahabte[1], length(string_aux));
      LEE_CTAHABTE(wr_ctahabte, llave_ctahabte, status_io);
      if status_io <> ok_io then
         DESPLIEGA_ERROR('OBTEN_CUENTAHABTE.LEE_CTAHABTE', status_io);
      with wr_ctahabte, wr_ctahabte_cap do
         begin
         string_aux := copy(blancos, 1, sizeof(nombre_ctahabte));
         move(nombre_ctahabte[1], string_aux[1],
              sizeof(nombre_ctahabte));
```

```
            nombre_ctahabte_cap := string_aux;
            string_aux := copy(blancos, 1, sizeof(tpo_cuenta));
            move(cta_tesofe_ctahabte[1], string_aux[1], sizeof(tpo_cuenta));
            cta_tesofe_ctahabte_cap := string_aux;
            string_aux := copy(blancos, 1, sizeof(tpo_cuenta));
            move(cuenta_ctahabte[1], string_aux[1], sizeof(tpo_cuenta));
            cta_ctahabte_cap := string_aux;
            end;{end with}
          DrawEditScreen(ESR);
          end;
      end;{end with}
  end; {END OBTIENE CUENTAHABTE}

  FUNCTION VALIDA_DIVISA(var FR : FieldRec;
                         var ErrCode : byte;
                         var ErrorSt : stringptr): boolean;
  VAR
    resultado : boolean;
    longst    : word;
  begin
    valida_divisa:= off;
    longst := length(Trim(Fr.EditSt^));
    if longst = divisa_long then
       CHECA_REGISTRO(on, Divisas, FR, ErrCode, ErrorSt, resultado)
    else
       begin
       if longst = 0 then
         begin
         ErrCode := error_vacio;
         ErrorSt := @error_vacio_st;
         end
       else
         begin
         ErrCode := error_divisa;
         ErrorSt := @error_divisa_st;
         end;
       resultado := off;
       end;
    valida_divisa:= resultado;
   end; {END VALIDA_DIVISA}

  FUNCTION EXISTE_DIVISA(var FR: FieldRec;
                         var ErrCode: byte;
                         var ErrorSt: StringPtr): boolean;
  VAR
    resultado : boolean;
    longst    : word;
  begin
    existe_divisa:= off;
    longst := length(Trim(FR.EditSt^));
    if longst = divisa_long then
       CHECA_REGISTRO(off, Divisas, FR, ErrCode, ErrorSt, resultado)
    else
       begin
       if longst = 0 then
         begin
         ErrCode := error_vacio;
         ErrorSt := @error_vacio_st;
         end
       else
         begin
         ErrCode := error_divisa;
         ErrorSt := @error_divisa_st;
         end;
       end;
    existe_divisa:= resultado;
  end; {END EXISTE_DIVISA}


  PROCEDURE OBTIENE_DIVISA(var ESR :ESRecord);

  VAR
  wr_divisa     : r_divisa;
  llave_divisa  : llavel_div;
  status_io     : integer;
```

```
  string_aux     : string;
  begin
  with ESR do
     begin
     if (CurrentID = 0) and
        (CurrentField^.EditSt^ <> copy (blancos, 1, divisa_long))
     then
        begin
        ChangeProtection(ESR, 0, on);
        string_aux := CurrentField^.EditSt^;
        move(blancos[1], llave_divisa[1], divisa_long);
        move(string_aux[1], llave_divisa[1], divisa_long);
        LEE_DIVISA(wr_divisa, llave_divisa, status_io);
        if status_io <> ok_io then
           DESPLIEGA_ERROR('OBTEN_DIVISA.LEE_DIVISA', status_io);
        with wr_divisa, wr_divisa_cap do
           begin
           string_aux := copy(blancos, 1, sizeof(tpo_beneficiario));
           move(nombre_esp_div[1], string_aux[1],
                sizeof(tpo_beneficiario));
           nombre_esp_div_cap:= string_aux;
           string_aux := copy(blancos, 1, sizeof(tpo_beneficiario));
           move(nombre_ing_div[1], string_aux[1],
                sizeof(tpo_beneficiario));
           nombre_ing_div_cap := string_aux;
           end;
        DrawEditScreen(ESR);
        end;
     end;{end with}
  end; {END OBTIENE CORRESP}

  FUNCTION VALIDA_CUENTA(var FR : FieldRec;
                         var ErrCode : byte;
                         var ErrorSt : stringptr): boolean;

  begin
     valida_cuenta:= true;
  end; {END VALIDA_CUENTA}


END.
```