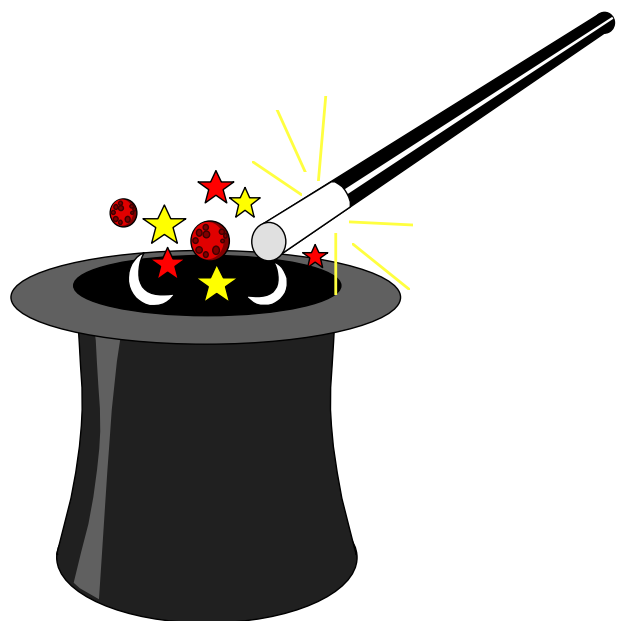


**Lenguaje Opal de
Programación de Informes**
LOPI



**Lenguaje Opal de
Programación de Informes
LOPI**

Indice de Materias

1.0. Qué es LOPI?.....	5
1.1. Cómo se trabaja con LOPI	5
1.2. Cómo ejecutar un informe LOPI	5
1.3. Cómo editar un informe LOPI	6
1.4. Macro-lenguaje LOPI	7
1.4.1. Variables.....	7
1.4.2. Calculador.....	13
1.4.3. Recálculo de variables.....	18
1.4.4. Instrucciones de control de proceso.....	19
1.4.5. Tablas.....	23
1.4.6. Consultas SQL.....	34
1.4.7. Entrada de datos por pantalla.....	35
1.4.8. Formatos especiales de impresión.....	37
1.4.9. Funciones.....	41
1.4.10. Macros.....	50
1.4.11. Interacción con el usuario.....	54
1.4.12. Salida (impresión) de los informes.....	54
1.4.13. Graficador.....	62
1.4.14. Programas LOPI comentados.....	64
2.0. Actualizaciones, adiciones y mejoras.....	65
2.1. Archivos de texto *.txt (versión 2.01 de Opal).....	65
2.2. Comandos SQL de ejecución (versión 2.2 de Opal).....	65
2.3. Modificar tablas (versión 2.2 de Opal).....	66
2.4. Abrir tablas regulares como auxiliares (versión 2.2 de Opal)...	66

1.0. Qué es LOPI ?

LOPI (Lenguaje Opal de Programación de Informes) es un macro-lenguaje que permite al usuario diseñar sus propios informes para ser ejecutados desde cualquier módulo **Opal**. **LOPI** permite acceder todas las tablas de datos de **Opal**, generar consultas mediante comandos SQL, definir variables para almacenar y modificar datos e incluye instrucciones procedimentales del tipo de lazo (similares a **while** en Pascal), de control de ejecución (similares a **if** de Pascal) y de ejecución programada de lazos (similares a **for** en Pascal).

1.1. Cómo se trabaja con LOPI?

Asumiendo que Usted ya conoce **LOPI**, para programar un nuevo informe debe seguir los siguientes pasos :

- Programar su informe en **LOPI** : un programa en **LOPI** es un archivo en formato .rtf (rich text format) ; rtf es un formato estandar de Windows y los archivos en este formato pueden editarse utilizando el editor **WordPad** de Windows, el editor **Word** de Microsoft o el editor **RTF** que viene incluido en **Opal**.
- Ejecutar su informe desde **Opal** : una vez diseñado y probado su programa en **LOPI**, cualquier usuario de **Opal** puede ejecutarlo llamándolo desde este sistema. **Opal** lee el archivo .rtf que contiene el informe programado en **LOPI** y lo ejecuta siguiendo las instrucciones del programador ; de esta forma los usuarios de **Opal**, aún no conociendo **LOPI**, pueden perfectamente ejecutar estos programas y obtener los informes que fueron previamente diseñados.

Usted como programador en **LOPI** puede prestar servicios de programación a los clientes que utilicen el sistema **Opal** y satisfacer las necesidades de éstos en cuanto a informes personalizados ; asimismo tener sus propias librerías de informes **LOPI** prediseñados que puede comercializar entre sus clientes. Usted, como usuario **Opal** y conociendo la programación en **LOPI**, puede aumentar exponencialmente la utilidad de su sistema **Opal**, al poder programar sus propios informes en la forma que Usted desee, sin necesidad de depender de programadores externos o internos, ni de invertir altas sumas en programación.

OpalSoft, como productora de **Opal** y **LOPI** pretende incluir y mantener en su sitio web, una librería de programas **LOPI** de dominio público, de forma de aumentar el valor agregado de **Opal** e incrementar la utilidad que pueden obtener los usuarios finales del uso de éste.

1.2. Cómo ejecutar un informe LOPI?

Ejecutar un programa o informe en **LOPI** es muy sencillo. En los ejemplos que vienen con el sistema **Opal** vienen los siguientes informes **LOPI** pre-diseñados para que Usted pueda utilizarlos con los datos de la empresa modelo Empresa Ejemplo, c.a. :

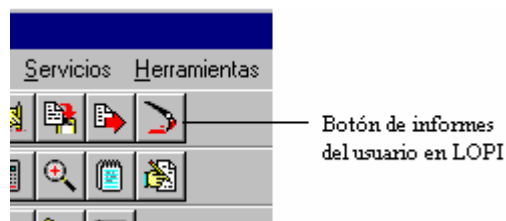
- Comparativo de Ganancias y Pérdidas (Contabilidad)
- Indices financieros (Contabilidad)
- Variación en el Capital de Trabajo (Contabilidad)
- Aplicación de Fondos (Contabilidad)
- Impresión de un cheque interno (Administrativo)
- Impresión de un cheque de proveedor (Administrativo)
- Impresión de una factura de servicios (Administrativo)
- Libro de ventas del IVA (Administrativo)
- Uso del crédito : incluye gráfico (Administrativo)
- Partidas del circulante : incluye gráfico (Contabilidad)

A la derecha de cada informe se indica el módulo **Opal** que Usted debe tener para ejecutarlo ; cómo el módulo Administrativo incluye contabilidad, todos los informes para el módulo de Contabilidad pueden ejecutarse con el Administrativo ; no obstante, los informes que corren en el módulo Administrativo no

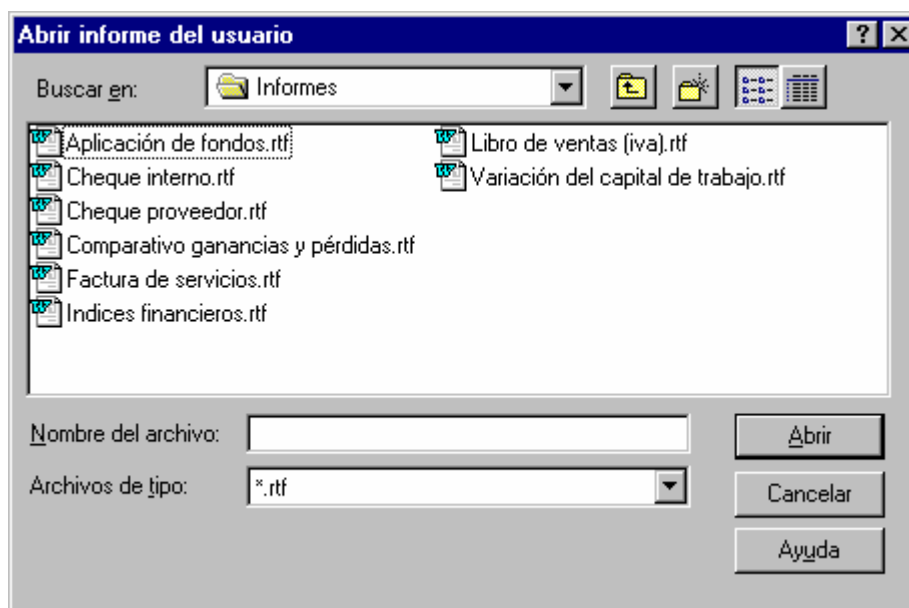
pueden ejecutarse si Usted tiene sólo el módulo de Contabilidad.

Estos informes fueron diseñados para los datos de la empresa modelo Empresa Ejemplo, c.a. que viene incluida en los ejemplos de **Opal**. Si Usted desea utilizarlos en otra(s) empresa(s), debe modificarlos para adaptarlo a los planes de cuentas contables y definición de tipos de documentos de la(s) nueva(s) empresa(s). Modificar informes pre-diseñados en un buen ejercicio para aprender a utilizar **LOPI**.

Para ejecutar alguno de estos informes entre en **Opal** y abra la empresa Empresa Ejemplo, c.a.; desde la barra de datos puede hacer clic en el botón de informes **LOPI** del usuario (forma de pincel):



También puede seguir la ruta del menú Informes – Informes del usuario (LOPI).



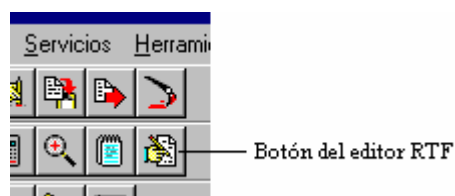
Al abrir la forma escoja el informe **LOPI** que desea imprimir y haga clic en Abrir; siguiendo las instrucciones del programa obtendrá el informe solicitado.

1.3. Cómo editar un informe LOPI?

Los informes en **LOPI** son archivos en formato .rtf (rich text format) estandar de Windows. Estos archivos pueden ser abiertos para ser revisados, impresos o editados mediante cualquier editor que reconozca el formato .rtf. Windows trae incluido como parte del sistema operativo el editor **WordPad**, el cual Usted puede invocar siguiendo la ruta Inicio-Programas-Acesorios-WordPad.

También el conocido editor **Word** de la suite de herramientas de oficina **Office** de Microsoft, permite abrir y editar archivos .rtf. Para abrir un archivo .rtf con **WordPad** o **Word**, seleccione el formato RTF en la persiana que indica el Tipo de archivo, dentro de la forma de abrir archivos.

Opal trae incluido un editor RTF muy sencillo el cual puede invocarse directamente desde el sistema ; para ello utilice el botón del editor RTF en la barra de Servicios del programa :



Los archivos modelos de informes **LOPI** que vienen en **Opal** puede encontrarlos en el directorio \Opal\Informes, si Usted instaló en el directorio \Opal el software **Opal**. Desde este directorio puede abrir cualquiera de estos archivos para revisarlos, imprimirlos o editarlos, utilizando cualquiera de los editores en el formato RTF antes mencionados.

1.4. Macro-lenguaje LOPI

A continuación una explicación detallada del macro-lenguaje **LOPI**. Para los usuarios con conocimientos de programación es relativamente sencillo aprender a utilizar **LOPI** ya que el macro-lenguaje es bastante pequeño y solamente cuenta con unas pocas instrucciones de control de proceso y unas cuantas funciones para la conversión de constantes y variables. Para los usuarios sin conocimientos previos de programación será un poco mas difícil, pero en ningún caso imposible, y creemos que haciendo un pequeño esfuerzo para entender los mecanismos del lenguaje, y con un estudio cuidadoso de los archivos de ejemplo que trae **Opal**, es muy sencillo, dedicando algunas horas, con un poco de paciencia, hacerse un programador bastante avezado en el macro-lenguaje **LOPI**.

OpalSoft pretende en un futuro, dependiendo de las sugerencias y opiniones de los usuarios y del uso del lenguaje en el diseño de informes, ir aumentando el número de funciones y facilidades que la herramienta ofrezca, de manera de incrementar la capacidad de **LOPI** para resolver problemas de diseño de informes personalizados para los usuarios de los sistemas **Opal**.

1.4.1. Variables

El primer aspecto que debemos conocer en **LOPI** es el relativo al manejo de las variables. Una variable es una zona en la memoria del computador en donde el usuario guarda una información o valor de su interés ; imagine que Usted tiene un casillero de muchas casillas ; que cada casilla viene con un nombre propio asignado que la identifica ; y que dentro de cada casilla hay un valor que Usted almacenó allí para ser utilizado en cualquier momento cuando Usted diseña su informe.

Por ejemplo : Usted va a imprimir una factura ; este documento requiere que se le imprima el nombre del cliente al cual se emite la factura. Usted necesita guardar en la memoria del computador (casillero) el nombre de este cliente, de forma que al momento de imprimir la factura Usted pueda ir a la casilla que contiene el nombre del cliente y sacar de allí el valor (nombre del cliente) para poder imprimirlo en la factura.

Supongamos que Usted desea nombrar la casilla que contiene el nombre del cliente como : nombre_cliente ; entonces si su cliente se llama, por ejemplo, 'FERRETERIA CARIBE, C.A.', Usted necesita un mecanismo para poder decir que la casilla llamada 'nombre_cliente' contiene el valor 'FERRETERIA CARIBE, C.A.'.

En **LOPI** esto se hace así :

.nombre_cliente; 'FERRETERIA CARIBE, C.A.'

Observe que el nombre de la variable comienza con un punto en la primera columna de la línea ; a continuación, totalmente pegado al punto, el nombre de la variable (casilla) ; después un punto y coma y a continuación el valor que contiene la variable, en este caso 'FERRETERIA CARIBE, C.A.'. En esta definición observemos varias cosas :

1. La definición de la variable está impresa en color rojo ; es indiferente el color a los efectos de la ejecución del informe. No obstante hemos tomado la convención de marcar la definición de variables en rojo a los efectos de facilitar la lectura de los programas en **LOPI**.
2. Debe haber un punto en la primera columna de la línea ; esto es muy importante y es la única forma de que LOPI reconozca que se está definiendo una variable ; si el punto está en el columna 2 o cualquier columna subsiguiente, aunque a la izquierda del punto sólo hayan espacios en blanco, **LOPI no identificará esto como una variable** y simplemente imprimirá la línea en el informe de salida como si fuese una línea de impresión mas.
3. El nombre de la variable no contiene espacios en blanco ; si Usted desea separar para dar mas claridad al nombre utilice el caracter _ como medio de separación. De esta forma la variable que contiene el nombre de los clientes se llama 'nombre_cliente'.
4. Los nombres de variables deben comenzar siempre por un dígito alfabético (A..Z, a..z), en mayúsculas o minúsculas (en OpalSoft preferimos usar minúsculas, pero eso es al gusto de cada usuario) ; no deben contener espacios en blanco, y debe estar formada por caracteres alfanuméricos (A..Z, a..z, 0..9) y/o el caracter especial _.
5. Nombres válidos de variables serían : nombre, codigo, año, x01, ciudad_03, tarjeta_identidad, acción, etc. ; observe que puede utilizar acentos y eñes al definir el nombre de sus variables.
6. Nombres inválidos de variables serían : nombre y apellido (contiene espacios en blanco ; debe ser : nombre_y_apellido) ; 0_pais (empieza por cero, y debe comenzar por una letra).

El valor contenido en una variable es otra cuestión que debemos estudiar. **LOPI** reconoce cuatro (4) tipos de valores a estar contenidos en una variable : caracteres, fechas, números y lógicos.

Los valores en caracteres son textos y deben estar encerrados entre comillas ; por ejemplo FERRETERIA CARIBE, C.A. es un valor en caracteres y para asignarlo a la variable nombre_cliente en una instrucción **LOPI** debe encerrarse entre comillas, como se indica arriba.

Cuando el valor en caracteres comienza por un caracter alfabético (A..Z, a..z) y no contiene espacios en blanco, puede asignarse como valor de caracteres a una variable sin utilizar las comillas, por ejemplo :

.codigo ; FC

Esta instrucción asigna el valor de caracteres FC a la variable codigo. También puede escribirse :

.codigo ; 'FC'

Cuando el valor comienza con un caracter distinto a una letra o contiene espacios en blanco es indispensable utilizar las comillas ; por ejemplo :

.codigo ; '16FC'

.codigo ; 'FC Facturas del cliente'

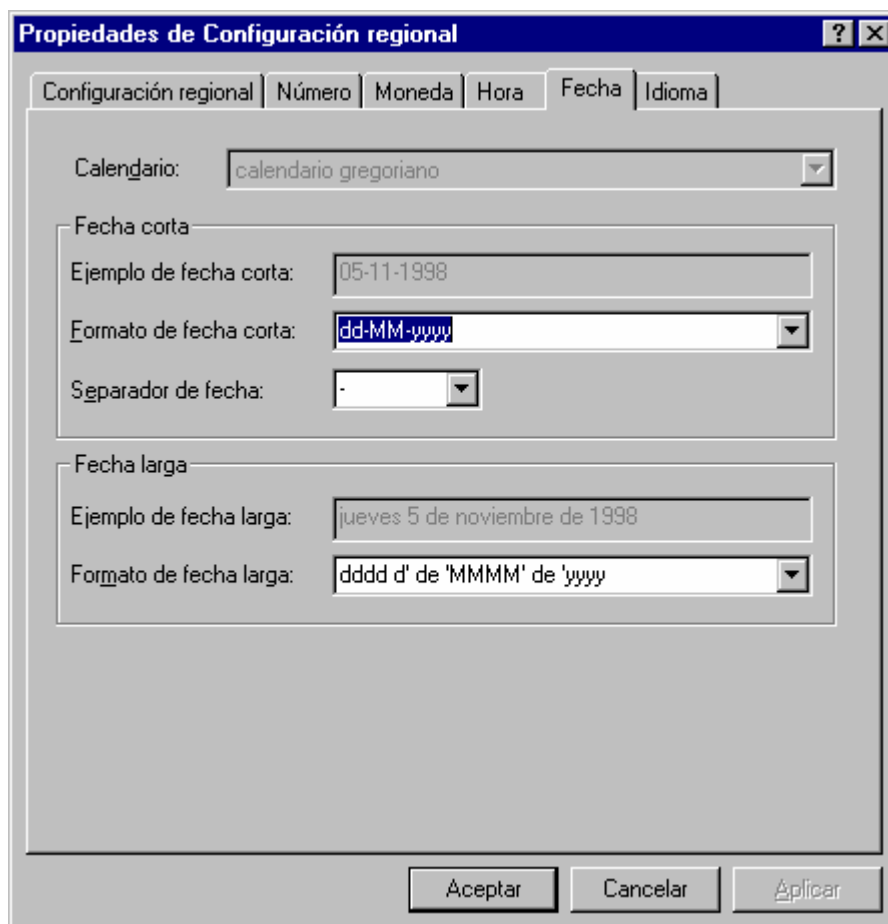
En el primer caso comienza por el caracter '1' ; en el segundo caso contiene caracteres en blanco.

Los valores de fechas son fechas y se asignan a una variable de la forma siguiente :

.fecha_cheque; &15-10-1998

En este ejemplo la variable fecha_cliente contendrá el valor correspondiente a la fecha 15 de octubre de 1.998. Observe que hay que colocar el caracter & pegado y a la izquierda de la fecha para indicar que lo que viene a continuación es una fecha.

El formato de la fecha a la derecha del caracter & debe ser compatible con el formato de fecha corta del Windows. Para evitar problemas y estar cónsonos con las fechas con años de cuatro dígitos compatibles con el año 2.000, lo recomendable es seguir la ruta en Windows Inicio-Panel de Control-Configuración regional-Fecha y poner la fecha en formato dd-mm-yyyy como se indica en la figura siguiente :



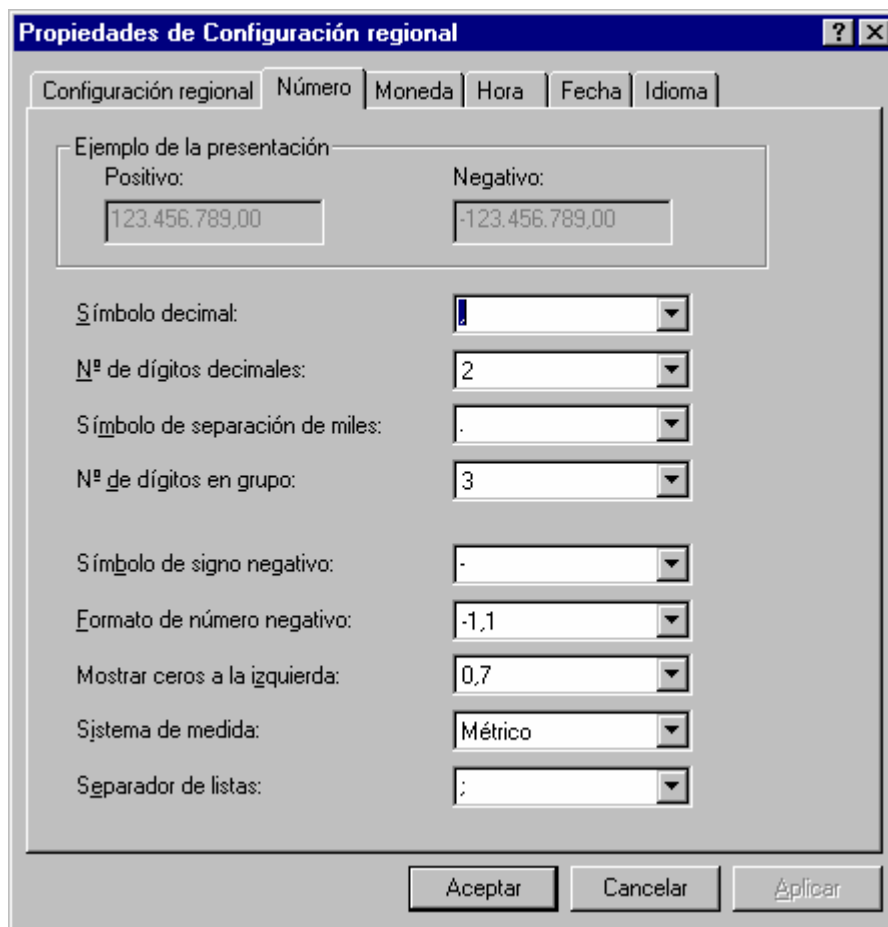
Este formato es estandar para fechas compatibles con el año 2.000 y es el utilizado en los informes modelos **LOPI** que vienen en el software **Opal**. **Es preferible que haga este cambio en su sistema** ; estará al día en lo referente al año 2.000 y los informes escritos en **LOPI** no genererán problemas al momento de ejecutarlos.

Los valores numéricos son números y se asignan a una variable de la forma siguiente :

.tasa_cambio; 571,50

En este ejemplo la variable tasa_cambio contendrá el valor correspondiente al número 571,50.

El formato del número a la derecha del punto y coma debe utilizar como caracter separador de los decimales el compatible con el formato de números de Windows. Para evitar problemas y estar cónsonos con los números escritos en español, lo recomendable es seguir la ruta en Windows Inicio-Panel de Control-Configuración regional-Número y poner el formato de números como se indica en la figura siguiente :



El símbolo decimal es coma ‘,’ y el separador de miles es punto ‘.’. Este formato es estándar para números escritos en español y es el utilizado en los informes modelos **LOPI** que vienen en el software **Opal**. Es preferible que haga este cambio en su sistema; estará al día en lo referente a los números separados en los decimales por coma y en los miles por punto y los informes escritos en **LOPI** no generarán problemas al momento de ejecutarlos.

Es importante observar que al asignar números a variables en **LOPI**, debe separar los decimales con coma (si hubiesen o fuese necesario), pero no debe separar los miles con puntos; en el ejemplo de arriba para asignar el valor numérico 1.245.679,32 a la variable tasa_cambio debe usarse la siguiente instrucción:

.tasa_cambio; 1245679,32

Los valores lógicos sólo pueden valer CIERTO o FALSO y se asignan a una variable de la forma siguiente:

.contribuyente; .t.

En este ejemplo la variable contribuyente contendrá el valor CIERTO (TRUE) y para asignarlo se usa la combinación de caracteres .t. (3 caracteres); el punto abriendo y cerrando la letra t permite a **LOPI** saber que esto es un valor lógico y no un valor de caracteres. Esta forma de definir un valor lógico lo hemos tomado del famoso lenguaje **DBASE** y los programadores conocedores de éste ya están familiarizados con esta convención. El valor de FALSO (FALSE) se asigna con .f. (3 caracteres: punto, f, punto).

Para imprimir el valor contenido en una variable **LOPI** utiliza la siguiente instrucción :

El nombre del cliente es : @nombre_cliente

Cuando **LOPI** ejecuta esta línea de programa simplemente imprimirá el texto 'El nombre del cliente es :' y luego al encontrar el caracter arroba @ seguido inmediatamente del nombre de la variable 'nombre_cliente', buscará el valor contenido en dicha variable y lo imprimirá reemplazando '@nombre_cliente' por el valor contenido en la variable, así :

El nombre del cliente es : FERRETERIA CARIBE, C.A.

Como **LOPI** trabaja con archivos en formato .rtf es posible utilizar toda la variedad de fuentes en tipos, formas y tamaños que utiliza Windows y las siguientes instrucciones en **LOPI** darán los siguientes resultados :

El nombre del cliente es : @nombre_cliente

El nombre del cliente es : **FERRETERIA CARIBE, C.A.**

El nombre del cliente es : @nombre_cliente

El nombre del cliente es : **FERRETERIA CARIBE, C.A.**

El nombre del cliente es : @nombre_cliente

El nombre del cliente es : **FERRETERIA CARIBE, C.A.**

El nombre del cliente es : @nombre_cliente

El nombre del cliente es : **FERRETERIA CARIBE, C.A.**

Ahora bien, imprimir el valor de una variable no es lo mismo que asignar un valor a una variable ; cuando asignamos valores simplemente decimos : esta variable x contiene el valor X ; pero cuando deseamos imprimir el valor contenido en la variable x podemos utilizar distintos formatos de impresión. Para resolver esta situación, en **LOPI** hacemos de esta forma :

.nombre_cliente ; 'FERRETERIA CARIBE, C.A.' ; 60

Dos puntos a la derecha del valor contenido en la variable y el número 60 nos permiten decir que la variable nombre_cliente debe ser impresa con un ancho de 60 caracteres.

Si Usted no coloca nada a la derecha del valor de la variable (no define una máscara de impresión) **LOPI** imprimirá el contenido de la variable libremente, sin controlar que no supere o sobrepase un ancho determinado. Colocando una máscara de impresión de 60, como en el ejemplo, tendremos la garantía de que la variable será impresa con un ancho de 60 caracteres, sea que mida menos o mas de esto ; si mide menos **LOPI** rellenará la diferencia con espacios en blanco ; si mide mas, truncará el texto para que alcance exactamente a 60 caracteres.

Es importante definir un ancho de impresión para las variables de caracteres cuando éstas se imprimen en informes columnares, es decir, en columnas unas a la derecha de las otras. Si no se establece un ancho de impresión los resultados quedan descuadrados, por ejemplo :

<u>Cuenta</u>	<u>Nombre de la cuenta contable</u>
1	ACTIVO
11	ACTIVO CIRCULANTE
111	DISPONIBLE

En este ejemplo no se asigna ancho de impresión a la variable que contiene el código de las cuentas contables ; como los valores varían en ancho (1, 2, 3, 4... caracteres) entonces se corren los valores a la derecha que contienen el nombre de la cuenta, dependiendo del número de caracteres que contenga la cuenta misma. Asignando un ancho de impresión de por ejemplo 20, a la variable que contiene el código de las cuentas se obtiene el efecto deseado :

<u>Cuenta</u>	<u>Nombre de la cuenta contable</u>
1	ACTIVO
11	ACTIVO CIRCULANTE
111	DISPONIBLE

Los formatos de impresión para variables numéricas, de fechas y lógicas siguen el patrón establecido en los formatos de impresión de **Opal**. Vea la sección 3, capítulo 46.0, página 69, para una explicación detallada de los formatos de impresión.

Cuando una variable numérica, de fecha o lógica es definida en **LOPI** sin asignarle un formato de impresión específico, entonces el sistema utilizará los formatos de impresión vigentes en **Opal** al momento de imprimir el informe, para asignar un formato de impresión por omisión a estas variables. Por ejemplo :

.monto1 ; 1580734,50
.monto2 ; 1580734,50 ; ##,###,##
.fecha ; &01-10-1998; dddd, dd 'de' mmmm 'de' yyyy
.año1 ; 1998 ; ###0
.año2 ; '1998' ; 10

En el primer ejemplo la variable **monto1** se imprimirá utilizando el formato de impresión de **Opal** vigente al momento de imprimir el informe ; **monto2** se imprimirá utilizando el formato de impresión indicado en la línea de definición de la variable ; **fecha** se imprimirá utilizando el formato de fechas indicado en la definición de la variable : **ddd mmmm yyyy** ; el resultado será : **jueves, 01 de octubre de 1998** ; **año1** es una variable numérica y vale 1998 (en número); **año2** es una variable de caracteres y vale 1998 (en caracteres o texto).

En general, es recomendable asignarle su formato de impresión a cada variable definida en **LOPI**. Los usuarios pueden modificar los formatos de impresión de **Opal** y si nos basamos en ellos podrían descuadrar un informe que hemos diseñado previamente en **LOPI** presuponiendo un formato que luego es modificado.

Para terminar este aparte queremos aclarar lo siguiente ; una variable puede ser definida asignándole un valor constante como hemos indicado arriba. No obstante, una variable también puede ser definida asignándole el valor de otra variable previamente definida, por ejemplo :

.monto ; 100 ; ###,###.##
.debito ; @monto ; ###,###,##0.00

En este ejemplo la variable monto vale 100 ; la variable debito también valdrá 100 pues se le asigna el valor de la variable monto con la instrucción indicada arriba. Los formatos de impresión de las variables son diferentes. De esta forma :

La variable monto vale : @monto
La variable debito vale : @debito

Se imprimirá cuando se ejecute desde **LOPI** como :

La variable monto vale : 100
La variable debito vale : 100,00

1.4.2. Calculador

LOPI incluye un mecanismo de cálculo que procesa variables numéricas, lógicas, de caracteres y de fechas. Los operadores que reconoce el calculador son los siguientes :

Operadores numéricos :

+ Sumar
- Restar
* Multiplicar
/ Dividir

Operadores de caracteres :

+ Sumar

Operadores lógicos :

= Igual
<> Diferentes
> Mayor que
>= Mayor o igual que
< Menor que
<= Menor o igual que
and Operador lógico AND
or Operador lógico OR
not Operador lógico NOT

Operadores de fechas :

- Restar

Primero haremos un breve repaso de cada operador ; luego veremos el funcionamiento del calculador de **LOPI**.

Los operadores numéricos se usan para operar con variables numéricas ; son suficientemente conocidos por todos. Si se cumple que :

X = 12
Y = 4

Entonces : X + Y = 16
X - Y = 8
X * Y = 48
X / Y = 3

El operador de caracteres + permite sumar dos variables de caracteres. Si se cumple que :

X = '25 de diciembre, '
Y = 'es el día de navidad'

Entonces : X + Y = '25 de diciembre, es el día de navidad'

Los operadores lógicos =, <>, >, >=, <, <= permiten comparar valores numéricos, de caracteres y de fechas.
Si se cumple que :

X = 10
Y = 20
Z = 10
F = 01-01-1980
G = 20-07-1981
H = 01-01-1980
R = 'GUSTAVO'
S = 'ZAMORA'
T = 'GUSTAVO'

Entonces :
X = Y es FALSO
X = Z es CIERTO
Y = Z es FALSO
F = G es FALSO
F = H es CIERTO
G = H es FALSO
R = S es FALSO
R = T es CIERTO
S = T es FALSO

X <> Y es CIERTO
X <> Z es FALSO
Y <> Z es CIERTO
F <> G es CIERTO
F <> H es FALSO
G <> H es CIERTO
R <> S es CIERTO
R <> T es FALSO
S <> T es CIERTO

X > Y es FALSO
Y > X es CIERTO
X > Z es FALSO
Z > X es FALSO
Y > Z es CIERTO
Z > Y es FALSO
F > G es FALSO
G > F es CIERTO
F > H es FALSO
H > F es FALSO
G > H es CIERTO
H > G es FALSO
R > S es FALSO
S > R es CIERTO
R > T es FALSO
T > R es FALSO
S > T es CIERTO
T > S es FALSO

X >= Y es FALSO
Y >= X es CIERTO
X >= Z es CIERTO
Z >= X es CIERTO
Y >= Z es CIERTO
Z >= Y es FALSO
F >= G es FALSO
G >= F es CIERTO
F >= H es CIERTO
H >= F es CIERTO
G >= H es CIERTO
H >= G es FALSO
R >= S es FALSO
S >= R es CIERTO
R >= T es CIERTO
T >= R es CIERTO
S >= T es CIERTO
T >= S es FALSO

X < Y es CIERTO
Y < X es FALSO
X < Z es FALSO
Z < X es FALSO
Y < Z es FALSO
Z < Y es CIERTO
F < G es CIERTO
G < F es FALSO
F < H es FALSO
H < F es FALSO
G < H es FALSO
H < G es CIERTO
R < S es CIERTO
S < R es FALSO
R < T es FALSO
T < R es FALSO
S < T es FALSO
T < S es CIERTO

X <= Y es CIERTO
Y <= X es FALSO
X <= Z es CIERTO
Z <= X es CIERTO
Y <= Z es FALSO
Z <= Y es CIERTO
F <= G es CIERTO
G <= F es FALSO
F <= H es CIERTO
H <= F es CIERTO
G <= H es FALSO
H <= G es CIERTO
R <= S es CIERTO
S <= R es FALSO
R <= T es CIERTO
T <= R es CIERTO
S <= T es FALSO
T <= S es CIERTO

Los operadores lógicos **and**, **or** y **not** permiten operar sobre valores lógicos, dando como resultado otro valor lógico ; la regla para ellos es la siguiente :

CIERTO **and** CIERTO = CIERTO
CIERTO **and** FALSO = FALSO
FALSO **and** CIERTO = FALSO
FALSO **and** FALSO = FALSO.

Vemos que el operador **and** para que de como resultado CIERTO, los dos operandos deben ser CIERTOS.

CIERTO **or** CIERTO = CIERTO
CIERTO **or** FALSO = CIERTO
FALSO **or** CIERTO = CIERTO
FALSO **or** FALSO = FALSO.

Vemos que el operador **or** para que de como resultado FALSO, los dos operandos deben ser FALSOS.

not CIERTO = FALSO
not FALSO = CIERTO

El operador **not** niega siempre la condición de su operando.

El operador – de fechas permite restar dos fechas dando su resultado en días ; ejemplo :

20-07-1998 - 10-07-1998 = 11 días.

Observe que la diferencia incluye el conteo de los días extremos ‘desde’ y ‘hasta’ ; en el ejemplo serían once días = 10, 11, 12, 13, 14, 15, 16, 17, 18, 19 y 20 de julio.

El calculador **LOPI** utiliza la notación **POLACA INVERSA** para su esquema de cálculos ; este es el esquema utilizado en las calculadoras **HEWLETT-PACKARD** de bolsillos (estandar, financieras y científicas) ; aunque tal vez este esquema sea menos intuitivo (inicialmente), una vez dominado es muy sencillo de usar ; por otra parte al seleccionar esta notación, internamente es mas sencilla y eficiente la implementación del calculador, y se evita la utilización de los paréntesis.

Veamos brevemente un ejemplo concreto ; si deseamos calcular :

$$(6 - 3) \times (9 + 1) = 30$$

En la notación polaca inversa debemos entrar :

$$6 3 - 9 1 + *$$

No es necesario usar paréntesis. Observe que es igual a las calculadoras **HP** :

$$6 \langle \text{ENTER} \rangle 3 \langle \text{MENOS} \rangle 9 \langle \text{ENTER} \rangle 1 \langle \text{MAS} \rangle \langle \text{POR} \rangle$$

Imagine que el calculador es una pila de números ; es decir, cada número que Usted ingresa se coloca encima del anterior. Para sacar un número Usted debe comenzar siempre por el último que entró. Cuando se aplica un operador éste actúa siempre sobre los dos últimos números de la pila ; se calcula el resultado y éste es de nuevo colocado arriba en la pila .

Armados con el calculador podemos ahora asignar a las variables valores calculados y dentro de los cálculos unas variables pueden a su vez depender de otras variables :

```
.i ; 10  
.j ; @i 1 +  
.k ; @i @j *  
.l ; @k @i + @j /
```

En este ejemplo de varias variables definidas en **LOPI** veamos cual es el valor final de cada una de ellas :

```
i = 10  
j = i + 1 = 10 + 1 = 11  
k = i x j = 10 x 11 = 110  
l = (k + i) / j = (110 + 10) / 11 = 120 / 11 = 10,91
```

El calculador **LOPI** también permite el manejo de los operadores lógicos, de textos y de fechas ; por ejemplo :

```
.m ; 10  
.n ; 20  
.p ; @m @n =  
.q ; @m @n <
```

En este ejemplo de varias variables definidas en **LOPI** veamos cual es el valor final de cada una de ellas :

```
m = 10  
n = 20  
p = (m = n) = FALSO = .f.  
q = (m < n) = CIERTO = .t.
```

Recuerde que en **LOPI** : FALSO = **.f.** y CIERTO = **.t.**

Las cosas pueden ser un poquito mas complicadas ; por ejemplo :

```
.g ; @p @q and
```

```
g = FALSE and TRUE = FALSE
```

O también, calculando directamente **g** partiendo de **m** y **n**.

```
.g ; @m @n = @m @n < and
```

El cálculo de días entre fechas funciona igual que una resta normal :

```
.desde ; &01-07-1998  
.hasta ; &31-07-1998  
.dias ; @hasta @desde -
```

```
dias = 31-07-1998 - 01-07-1998 = 31
```

La suma de caracteres permite crear texto partiendo de texto ; por ejemplo :

```
.ciudad ; 'Caracas'  
.postal ; '2030'  
.estado ; 'Miranda'  
.pais ; 'Venezuela'  
.espacio ; ' '  
.coma ; ','  
.punto ; '.'  
.direccion ; @ciudad @espacio + @postal + @coma + @estado + @coma + @pais + @punto +
```

La dirección es : **@direccion**

Al procesar la línea anterior en **LOPI** se obtendrá la siguiente salida :

La dirección es : **Caracas 2030, Miranda, Venezuela.**

1.4.3. Recálculo de variables

La convención anteriormente explicada en el aparte **1.4.1.** permite definir una variable ; no obstante, a veces ocurre que una variable debe ser recalculada durante el proceso de ejecución de un informe. Para ello **LOPI** prevee el caracter ~ en la primera columna de una línea ; veamos :

```
.i ; 0
```

Esta instrucción define la variable **i** inicializada (valor inicial) a cero. Ahora bien, supongamos que deseamos utilizar **i** para contar algo dentro de un programa **LOPI** ; es decir **i** vale cero inicialmente, pero deseamos que en un momento determinado valga 1, 2, 3, 4, y así sucesivamente. En **LOPI** esto se hace así :

```
~i ; @i 1 +
```

Esta instrucción recalcula **i** asignándole el valor que tenía originalmente (cero) y sumándole 1 ; el resultado (uno) es asignado nuevamente a la variable **i** que terminará valiendo uno.

Cada vez que se ejecute esta instrucción la variable **i** se incrementará en 1.

```
~1 ; @i 1 +
```

Al finalizar **i** valdrá 2.

Es muy importante tener esto en claro ; en **LOPI** una variable para poder utilizarse debe estar definida con una instrucción comenzando por punto. Una vez definida, para recalcularse, debemos escribir la instrucción de recálculo comenzando con el símbolo ~.

Muchas personas (especialmente los programadores) preferirán definir todas sus variables al inicio del programa, asignándole valores en cero, falso o blanco ; luego en el transcurso del programa redefinirlas para asignarle los valores requeridos para los efectos del informe que se desea diseñar :

```
.linea ; 0  
.hoja ; 0  
.imprimir ; .f.  
.ciudad ; ' '  
.monto ; 0
```

En el ejemplo se definen las variables **linea**, **hoja** y **monto** como numéricas y se les da valor inicial cero ; **imprimir** es una variable lógica e inicialmente vale FALSO ; **ciudad** es una variable de caracteres e inicialmente vale ‘ ‘.

Durante la ejecución del programa estas variables pueden ser recalculadas, por ejemplo :

```
~linea ; @linea 1 +
~hoja ; @hoja 1 +
~imprimir ; @linea 0 >
~ciudad ; 'Barcelona'
~monto ; 105699,90
```

La variable **linea** se incrementa en 1 ; ahora vale 1. La variable **hoja** también ; ahora vale 1. La variable **imprimir** se recalcula como **linea > 0** ; como **linea = 1** entonces **imprimir = CIERTO**. La variable **ciudad** es recalcula como ‘Barcelona’ y la variable **monto** como 105.699,90.

Finalmente para terminar este aparte falta decir que **LOPI** es muy liberal en cuanto el tipo de sus variables ; en este aspecto es mas parecido a DBASE o BASIC que DELPHI (con lo cual se desarrolló **Opal** y **LOPI**) ; una variable puede definirse como numérica, lógica, de caracteres o fecha y luego ser redefinida a otro tipo al ser recalculada ; por ejemplo :

```
.x ; &01-10-1998
```

x se define como una variable de fecha igual a 01-10-1998.

En alguna parte del programa puede recalcularse a :

```
~x ; &30-11-1998
```

Ahora es 30-11-1998 ; no obstante sigue siendo una variable de fecha. Mas adelante en un programa se podría escribir :

```
~x ; 0
```

Ahora se recalcula a cero y por supuesto el tipo de la variable pasa de fecha a numérica. No hay restricción en **LOPI** para esto.

1.4.4. Instrucciones de control de proceso

LOPI maneja tres instrucciones sencillas de control del proceso o flujo del programa. Para los usuarios programadores éstas serían equivalentes a las instrucciones **while**, **if** y **for** de los lenguajes como PASCAL, C, DBASE, FOX, etc. Claro, mucho mas sencillas y menos poderosas ; no obstante, para la función de diseñar informes son mas que adecuadas y prestan un servicio suficiente al programador en **LOPI**.

Veremos primero el equivalente a la instrucción **if**. En **LOPI** para controlar la ejecución de una sección de código Usted puede escribir :

```
.si <condicion>
```

```
.fin
```

Nota : recuerde siempre que **LOPI** es un macro-lenguaje de impresión de informes ; por tanto las instrucciones que no sean para ser impresas deben comenzar con un caracter especial en la primera columna de la línea ; para la instrucción **.si** el punto debe colocarse en la primera columna de la línea. Igual en la instrucción **.fin**.

La instrucción arriba indica que si se cumple la condición marcada en **<condicion>** entonces todas las instrucciones que están entre el **.si** y el **.fin** se ejecutarán ; si la condición no se cumple entonces las instrucciones entre el **.si** y el **.fin** serán ignoradas por **LOPI**. La condición debe ser una variable o cálculo que se evalúe a **CIERTO** si se cumple ; y a **FALSO** si no se cumple. Por ejemplo :

.si @imprimir

Esta línea se imprimirá si la variable **imprimir** evalúa a **CIERTO**

.fin

Si la variable **imprimir** es **CIERTO**, entonces se imprime la línea con el texto indicado ; si la variable es **FALSO** entonces la línea no se imprime y **LOPI** avanza hasta conseguir la instrucción **.fin**, ignorando las instrucciones contenidas entre el **.si** y el **.fin**, para continuar nuevamente con la siguiente instrucción después del **.fin**.

El ejemplo un poco mas completo sería :

.imprimir ; .f.

.si @imprimir

Esta línea se imprime si la variable **imprimir** es **CIERTO**

.fin

.si @imprimir not

Esta línea se imprime si la variable **imprimir** es **FALSO**

.fin

En este ejemplo se imprimirá la segunda línea ya que **imprimir** es **FALSO**.

Para evitar el tener que utilizar dos instrucciones **.si** para resolver el flujo del programa dependiendo del valor de una variable, **LOPI** también permite la siguiente construcción :

~imprimir ; .t.

.si @imprimir

Esta línea se imprime si la variable **imprimir** es **CIERTO**

.sino

Esta línea se imprime si la variable **imprimir** es **FALSO**

.fin

Aquí se recalcula **imprimir** y se hace **CIERTO** ; observe que en este arreglo la instrucción **.sino** indica que si no se cumple lo de arriba entonces debe ejecutarse lo de abajo. En este caso se imprimirá la línea de arriba ya que la condición nos dice que **imprimir** es **CIERTO**.

Claro, la condición a la derecha del **.si** puede ser algo mas complejo ; por ejemplo :

.m ; 10

.n ; 15

.p ; 20

.si @m @n > @p 20 = and

Esto se imprime si **m** es mayor que **n** y **p** es igual a 20 (deben cumplirse las dos cosas)

.sino

Esto se imprimirá en caso contrario (**m** menor que **n** o **p** diferente de 20)

.fin

En el ejemplo la línea de arriba **NO** se imprimirá ya que **m** no es mayor que **n**, aun cuando **p** si es igual a 20. Se imprimirá la línea de abajo.

A continuación el equivalente de la instrucción **for**. En **LOPI** para controlar la ejecución repetida de una sección de código Usted puede escribir :

.repite <valor>

.fin

Las instrucciones que hayan entre la instrucción **.repite** y la instrucción **.fin** se ejecutarán tantas veces como el valor que contenga **<valor>**. Un ejemplo aclara la instrucción :

.i ; 30

.repite @i

Esta línea se imprimirá 30 veces

.fin

Cuando se ejecute esta sección de código **LOPI** el programa imprimirá :

Esta línea se imprimirá 30 veces

Esta línea se imprimirá 30 veces

.....

Esta línea se imprimirá 30 veces

La impresión ocurrirá 30 veces, que es valor contenido en la variable **i**.

A continuación el equivalente de la instrucción **while**. En **LOPI** para controlar la ejecución repetida de una sección de código Usted puede escribir :

.mientras <condicion>

.fin

Las instrucciones que hayan entre la instrucción **.mientras** y la instrucción **.fin** se ejecutarán tantas veces mientras el valor de **<condicion>** evalúe a **CIERTO**. Un ejemplo aclara la instrucción :

.i ; 1

.mientras @i 30 <=

Esta línea se imprimirá mientras i sea menor o igual a 30

La variable i vale : @i

~i ; @i 1 +

.fin

Cuando se ejecute esta sección de código **LOPI** el programa imprimirá :

Esta línea se imprimirá mientras i sea menor o igual a 30

La variable i vale : **1**

Esta línea se imprimirá mientras i sea menor o igual a 30

La variable i vale : **2**

.....

Esta línea se imprimirá mientras i sea menor o igual a 30

La variable i vale : **30**

Observe que inicialmente **i** vale 1 (se inicializa a 1); al entrar en la instrucción **.mientras** se cumple la condición de que **i** es menor o igual a 30; por tanto se ejecuta el código entre el **.mientras** y el **.fin**. Se imprime la primera línea y la segunda con el valor de **i = 1**. A continuación la variable **i** se recalcula sumándole 1; ahora **i** valdrá 2.

Al llegar a la instrucción **.fin**, **LOPI** se devuelve arriba al **.mientras** y evalúa de nuevo la condición; ahora **i** vale 2; como 2 es menor o igual a 30 la condición evalúa a CIERTO y por tanto debe repetirse de nuevo el lazo **.mientras .fin**. Se imprimen de nuevo las dos líneas; la segunda con **i = 2**. Antes del **.fin** se recalcula nuevamente **i** sumándole 1; ahora **i** vale 3.

El lazo se repetirá 30 veces hasta que **i** se haga igual a 31. Cuando **i** es igual a 31 la condición a la derecha del **.mientras** evalúa a FALSO y se deja de repetir el lazo; **LOPI** continuará con la siguiente instrucción después del **.fin** de este lazo **.mientras**.

Para finalizar este aparte debemos puntualizar, aunque es intuitivo, que las instrucciones de control del flujo de programa pueden anidarse entre sí, es decir, una instrucción de control puede estar contenida a su vez en otra instrucción de control. Por ejemplo:

```
.i ; 1 ; ##0
.j ; 1 ; ##0
.mientras @i 10 <=
En este lugar la variable i vale : @i
.mientras @j 10 <=
    En este lugar la variable j vale : @j
~j ; @j 1 +
.fin
~i ; @i 1 +
.fin
```

En este ejemplo el **.mientras** externo se ejecuta 10 veces de **i = 1** hasta **i = 10**. El **.mientras** interno anidado en el externo se ejecutará 100 veces de **j = 1** hasta **j = 10**; cada uno 10 veces de **i = 1** hasta **i = 10**.

La salida de este código ejecutado en **LOPI** será así:

```
En este lugar la variable i vale : 1
    En este lugar la variable j vale : 1
    En este lugar la variable j vale : 2
    .....
    En este lugar la variable j vale : 9
    En este lugar la variable j vale : 10
En este lugar la variable i vale : 2
    En este lugar la variable j vale : 1
    En este lugar la variable j vale : 2
    .....
    En este lugar la variable j vale : 9
    En este lugar la variable j vale : 10
.....
En este lugar la variable i vale : 10
    En este lugar la variable j vale : 1
    En este lugar la variable j vale : 2
    .....
    En este lugar la variable j vale : 9
    En este lugar la variable j vale : 10
```

1.4.5. Tablas

LOPI permite acceder las tablas de datos del sistema **Opal**. Este conjunto de tablas constituyen lo que se denomina **BASE DE DATOS DEL SISTEMA OPAL**. Para obtener esta información, en donde se indican las tablas que maneja el sistema, incluyendo nombre, uso y columnas que contiene cada tabla, es importante tener a la mano la **ESTRUCTURA DE LA BASE DE DATOS DE OPAL**. Para ello ejecute la opción Servicios-Generar archivo modelo Ascii.txt en el menú de Servicios del software. Después de generar el archivo, **Opal** le indicará si desea ver el archivo desde el **NotePad** de Windows. Hágalo y desde allí imprima el archivo como referencia de trabajo.

Desde **LOPI**, para acceder las tablas de datos se utilizan las siguientes instrucciones ; vamos a suponer, a efectos de una mejor claridad de la explicación, que pretendemos trabajar con la tabla ‘Cuentas’, que no es mas que la tabla de cuentas contables del sistema ; esta tabla aparecerá en la **ESTRUCTURA DE LA BASE DE DATOS DE OPAL**, definida de la forma siguiente :

OPAL: ESTRUCTURA DE LA BASE DE DATOS - 09-10-1998 13:09:43

.....

Cuentas
Cuentas de contabilidad
Cuentas.TXT
Codigo,A,1,16
Nombre,A,17,40
Naturaleza,I,57,6
Tipo,I,63,6
Historicos,I,69,6
Mascara,I,75,6
SaldoContable,F,81,18
Clase,I,99,6
Generar,I,105,6
Control,I,111,6
Balanceado,I,117,6

.....

Nota : recuerde siempre que **LOPI** es un macro-lenguaje de impresión de informes ; por tanto las instrucciones que no sean para ser impresas deben comenzar con un caracter especial en la primera columna de la línea ; para las instrucciones que manejan tablas debe colocarse el caracter ^ en la primera columna de la línea.

^ cuentas=abre

Esta instrucción abre la tabla nombrada **cuentas** de **Opal** ; observe que el nombre que utilizamos para identificar la tabla es el primero que aparece en la sección que la describe en la **ESTRUCTURA DE LA BASE DE DATOS DE OPAL**. Asimismo hay un espacio en blanco entre el símbolo ^ y el nombre de la tabla.

Para poder operar con una tabla es imprescindible que esté abierta previamente ; por tanto esta es la primera instrucción que debe tener un programa en **LOPI** antes de cualquier otra instrucción que pretenda acceder la información contenida en una tabla.

^ cuentas=inicio

Esta instrucción coloca el apuntador de la tabla de **cuentas** en el primer registro o fila de la tabla. Si la tabla tiene por ejemplo 100 registros o filas, esta instrucción hace que el apuntador del registro corriente o activo se

ubique en el primer registro o fila de la tabla. Los datos contenidos en el registro o fila apuntado (registro o fila activa) son los datos que en ese momento están disponibles de dicha tabla para ser asignados a variables, ser impresos, etc.

^ cuentas=fin

Esta instrucción coloca el apuntador de la tabla de **cuentas** en el último registro o fila de la tabla. Si la tabla tiene por ejemplo 100 registros o filas, esta instrucción hace que el apuntador del registro corriente o activo se ubique en el registro o fila #100 de la tabla.

^ cuentas=avanza

Esta instrucción avanza el apuntador de la tabla de **cuentas** al siguiente registro o fila de la tabla. Si el apuntador de registros está ubicado en el registro #21, al ejecutar la instrucción el apuntador avanzará al registro #22. Al alcanzar el fin de la tabla, o último registro o fila de la tabla, la instrucción no avanzará mas allá (no hay mas registros o filas) ; por tanto al ejecutarse se mantendrá en el último registro o fila alcanzado.

^ cuentas=retrocede

Esta instrucción retrocede el apuntador de la tabla de **cuentas** al registro o fila anterior de la tabla. Si el apuntador de registros está ubicado en el registro #21, al ejecutar la instrucción el apuntador retrocederá al registro #20. Al alcanzar el inicio de la tabla, o primer registro o fila de la tabla, la instrucción no retrocederá mas allá (no hay mas registros o filas) ; por tanto al ejecutarse se mantendrá en el primer registro o fila alcanzado.

.registros ; cuentas.registros

Esta instrucción asigna a la variable **registros** el valor del número de registros que contenga la tabla de **cuentas** ; si la tabla de cuentas tiene, por ejemplo, 156 registros o filas, al ejecutar esta instrucción la variable registros tendrá como valor 156.

El nombre de la variable puede ser el que desee el programador ; por ejemplo :

k ; cuentas.registros ; ##0

La tabla de cuentas tiene : @k registros o filas.

Si la tabla de cuentas tiene 156 registros o filas, al ejecutar esta sección de código **LOPI** se imprimirá :

La tabla de cuentas tiene : 156 registros o filas.

En ciertos casos necesitamos marcar un registro o fila de una tabla para movernos a otro registro, pero no deseamos perder el control del registro o fila en donde nos encontramos inicialmente ; para resolver esta cuestión tenemos las siguientes instrucciones **LOPI** :

^ cuentas=marca

^ cuentas=retorna

La primera instrucción marca la fila o registro en donde nos encontramos al momento de ejecutar la instrucción ; la segunda regresa nuevamente al registro marcado ; por ejemplo :

```

// Estamos en el registro #1 de la tabla de cuentas
^ cuentas=inicio
// Avanzamos 5 registros usando la instrucción .repite
.repite 5
^ cuentas=avanza
.fin
// En este momento debemos estar en el registro #5 ; vamos a marcarlo
^ cuentas=marca
// Avanzamos otros 10 registros mas, con otro .repite
.repite 5
^ cuentas=avanza
.fin
// Ahora debemos estar en el registro #15
// Para regresar nuevamente al registro #5 usamos la instrucción retorna
^ cuentas=retorna
// Ahora estamos nuevamente en el registro #5 !!

```

En esta sección de código **LOPI** vemos como escribir comentarios ; las líneas que comienzan por el par de caracteres // (caracteres 1 y 2 de la línea) son líneas de comentarios y no se ejecutan ; sirven para comentar el código y facilitar su lectura mas adelante, para nosotros mismos si fuimos quienes desarrollamos el programa, o para otros programadores que deseen revisar, corregir o modificar un programa hecho por nosotros.

Los comentarios son muy importantes ; cuando hacemos un programa tenemos unas ideas y las ponemos en práctica. Pasado algún tiempo no recordamos porqué nos planteamos o asumimos ciertas soluciones ; la forma mas sencilla de recordar, revisar, analizar, modificar, mejorar, corregir y/o trabajar con un código es que éste esté bien comentado por el programador que lo desarrolló ; seamos nosotros o un tercero.

En el ejemplo arriba vimos como marcamos un registro o fila de la tabla de **cuentas** para después, con una sólo instrucción, regresar rápidamente de vuelta a él.

Para poder imprimir el contenido de una columna o campo de un registro de una tabla, debemos asignarlo previamente a una variable ; por ejemplo : la tabla de **cuentas** (vea la estructura de la tabla arriba) contiene las columnas **codigo** y **nombre** que contienen el código o número de una cuenta contable y su nombre respectivamente.

Si deseamos imprimir el valor contenido en estas columnas o campos con **LOPI** debemos proceder de la forma siguiente :

```

^ cuentas=abre
^ cuentas=inicio
.codigo ; cuentas.codigo ; 20
.nombre ; cuentas.nombre ; 40

```

La tabla de cuentas contables, llamada CUENTAS, debe estar ahora ubicada en el primer registro de la tabla, y los valores de este registro deben ser :

```

Código : @codigo
Nombre : @nombre

```

Cuando Usted crea este archivo RTF y lo ejecuta desde el ejecutor **LOPI** de **Opal**, el programa imprimirá :

La tabla de cuentas contables, llamada CUENTAS, debe estar ahora ubicada en el primer registro de la tabla, y los valores de este registro deben ser :

```

Código : 1
Nombre : ACTIVO

```

Porque la tabla de cuentas contable contiene el registro correspondiente a la cuenta 1 : **ACTIVO** en el primer registro de la tabla.

Cuando recién se abre una tabla, y después de abierta si no se ha movido el apuntador de registros, éste está normalmente apuntando al primer registro de la tabla ; por tanto en el programa anterior pudimos haber suprimido la instrucción que coloca el apuntador al inicio de la tabla, así :

```
^ cuentas=abre
```

```
.codigo ; cuentas.codigo ; 20
```

```
.nombre ; cuentas.nombre ; 40
```

La tabla de cuentas contables, llamada CUENTAS, debe estar ahora ubicada en el primer registro de la tabla, y los valores de este registro deben ser :

Código : **@codigo**

Nombre : **@nombre**

Dará el mismo resultado de antes.

Con las instrucciones que sabemos hasta ahora de **LOPI** podemos hacer un pequeño informe de la tabla de cuentas contables, que indique el código y nombre de las cuentas contenidas en la tabla para las 10 primeras cuentas que existen; veamos :

```
// Este es mi programa que imprime los códigos y nombres de las cuentas contables
// Para ser un principiante no está nada mal, no cree ?
^ cuentas=abre
.cd; cuentas.codigo ; 20
.nm; cuentas.nombre ; 40
TABLA DE CUENTAS
-----
Código                Nombre de la cuenta
-----
.repite 10
@cd                    @nm
^ cuentas=avanza
.fin
```

Al ejecutar este código se obtiene el siguiente informe :

```
TABLA DE CUENTAS
-----
Código                Nombre de la cuenta
-----
1                     ACTIVO
11                    ACTIVO CIRCULANTE
111                   DISPONIBLE
1111                  DISPONIBLE EN CAJA Y BANCOS
11111                DISPONIBLE EN CAJA
1111102              Caja Chica - Caracas
1111103              Caja Chica - Valencia
1111121              Caja Principal - Caracas
1111122              Caja Principal - Valencia
11112                DISPONIBLE EN BANCOS
```

Observando el código **LOPI** aprovecharemos para explicar algunas cosas :

- Hemos usado una convención de colores para facilitar la lectura de los programas ; usamos **rojo** para la definición y recálculo de variables, **verde** para las instrucciones que operan sobre las tablas y **azul** para las instrucciones de control de flujo del programa. Mas adelante veremos que usaremos el **fucsia** para las instrucciones que manejan el graficador. Esta convención de colores nos facilita la lectura e interpretación de los programas en **LOPI**.
- Hemos escrito nuestro mini-programa utilizando como fuente el tipo 'Courier New' ; esta fuente tiene la gran ventaja de que sus caracteres tienen un ancho fijo y alínean perfectamente en los informes columnares ; si hubiesemos usado, por ejemplo, 'Times New Roman', que es una fuente de caracteres de anchos variables, el informe no hubiese quedado cuadrado como aparece arriba. De esta forma, en

general, es preferible utilizar fuentes de anchos fijos para preparar informes en **LOPI** ; esto facilita el cuadro de columnas y nos ahorra muchos dolores de cabeza.

Nuestro programa está muy bien, pero no es muy práctico ; primero, sólo imprime las primeras 10 cuentas de la tabla y segundo, si la tabla tiene, por ejemplo, solamente 6 registros, el informe que obtendríamos sería así :

TABLA DE CUENTAS

```
-----  
Código                Nombre de la cuenta  
-----  
1                     ACTIVO  
11                    ACTIVO CIRCULANTE  
111                   DISPONIBLE  
1111                  DISPONIBLE EN CAJA Y BANCOS  
11111                 DISPONIBLE EN CAJA  
1111102               Caja Chica - Caracas  
1111102               Caja Chica - Caracas  
1111102               Caja Chica - Caracas  
1111102               Caja Chica - Caracas  
1111102               Caja Chica - Caracas
```

Repite la última cuenta cinco veces ; porqué ?.

Para resolver esta situación vamos a introducir dos funciones que nos ayudan a determinar si hemos llegado al final de una tabla o al principio de ella ; veamos :

```
// Este es una nueva versión de mi programa usando la función fin  
^ cuantas=abre  
.cd ; cuantas.codigo ; 20  
.nm ; cuantas.nombre ; 40  
TABLA DE CUENTAS  
-----  
Código                Nombre de la cuenta  
-----  
  
.mientras cuantas fin <>  
  @cd                @nm  
^ cuantas=avanza  
.fin
```

En este ejemplo cambiamos la instrucción **.repite** por la instrucción **.mientras**. La condición que controla el lazo es **cuantas fin <>**; esta condición se lee: **mientras cuantas sea diferente de fin**. Mientras cuantas es diferente de fin (no hayamos alcanzado el fin de la tabla), la condición a la derecha de la instrucción **.mientras** retorna CIERTO y el lazo se repite; observe que debajo de la impresión de la línea, se avanza el apuntador de registros de la tabla con la instrucción **^ cuantas=avanza**. Llegará un momento en que avanzando sobre la tabla llegaremos al último registro; en esta situación, cuando se evalúe la condición **cuantas fin <>** esta retornará FALSO (estamos al final de la tabla y cuantas está en fin; por tanto la condición **cuantas fin <>** es FALSA, ya que **cuantas fin =** es CIERTO); en este momento se habrán terminado los registros de la tabla y finaliza el programa. Obtendremos ahora el informe completo:

TABLA DE CUENTAS

```

-----
Código          Nombre de la cuenta
-----
1              ACTIVO
11             ACTIVO CIRCULANTE
111           DISPONIBLE
1111          DISPONIBLE EN CAJA Y BANCOS
11111         DISPONIBLE EN CAJA
1111102       Caja Chica - Caracas
1111102       Caja Chica - Caracas
1111102       Caja Chica - Caracas
1111102       Caja Chica - Caracas
1111102       Caja Chica - Caracas
1111102       Caja Chica - Caracas

.....
.....

5200401        Intereses Créditos Comerciales
5200402        Intereses Créditos Bancarios
5200403        Intereses de Mora
5200404        Seguros
52005          OTROS GASTOS NO RECURRENTE
5200501        Ayudas y Donaciones
5200502        Exposiciones y Eventos
  
```

Esto está un poco mejor; no es así?

La función 'inicio' es lo contrario de la función 'fin'; retorna CIERTO cuando se llega al inicio o primer registro o fila de la tabla; repetamos el programa anterior para obtener las cuentas en orden inverso:

```

// Este es una nueva versión de mi programa usando la función inicio
^ cuentas=abre
^ cuentas=fin
.cd ; cuentas.codigo ; 20
.nm ; cuentas.nombre ; 40
TABLA DE CUENTAS
-----
Código          Nombre de la cuenta
-----

. mientras cuentas inicio <>
  @cd          @nm
^ cuentas=retrocede
.fin
  
```

Ahora abrimos la tabla, nos posicionamos en el último registro con ^ cuentas=fin y comenzamos a ejecutar el lazo subiendo desde el último registro hasta el primero, usando la instrucción ^ cuentas=retrocede. Obtendremos el informe de abajo hacia arriba:

```

TABLA DE CUENTAS
-----
Código          Nombre de la cuenta
-----
5200502        Exposiciones y Eventos
5200501        Ayudas y Donaciones
52005          OTROS GASTOS NO RECURRENTE

.....
.....

111           DISPONIBLE
11            ACTIVO CIRCULANTE
1             ACTIVO
  
```

Hasta ahora hemos hablado de primer registro o fila y último registro o fila de la tabla; ahora bien, cabría preguntar: pero..., cómo está ordenada la tabla?

La mayoría de las tablas en **Opal** traen al menos un ordenamiento básico o principal por el primer campo o columna de la tabla; este ordenamiento es el ordenamiento por omisión. Cuando Usted abre la tabla por primera vez, ella viene ordenada por este ordenamiento básico o principal.

Por ejemplo, la tabla de cuentas contables viene ordenada primeramente por la columna o campo **codigo** de la tabla; esta columna contiene el código o número de la cuenta. En el informe que preparamos arriba, recorriendo la tabla de arriba hacia abajo, vimos que las cuentas aparecen impresas ordenadas por código. Para este ordenador (básico o principal), en la tabla de cuentas de nuestro ejemplo tendremos:

```
Primer registro :      1                ACTIVO
Ultimo registro:    5200502            Exposiciones y Eventos
```

Esto es cierto tan pronto como el ordenador sea codigo. La tabla de **cuentas** trae un segundo ordenador por el nombre de la cuenta (columna o campo **nombre**); cuando ordenamos la tabla por este ordenador cambiamos la ubicación de los registros, y tendremos ahora un nuevo primer y último registro. Veamos como hacer esto:

```
// En esta versión ordenamos la tabla de cuentas por nombres.
^ cuentas=abre
^ cuentas.ordena=nombre
.cd ; cuentas.codigo ; 20
.nm ; cuentas.nombre ; 40
TABLA DE CUENTAS
-----
Código                Nombre de la cuenta
-----

.mientras cuentas fin <>
@cd                    @nm
^ cuentas=avanza
.fin
```

El resultado será lo siguiente:

```
TABLA DE CUENTAS
-----
Código                Nombre de la cuenta
-----

1                    ACTIVO
11                   ACTIVO CIRCULANTE
....
1111203              BANCO INDUSTRIAL
1111203001           Banco Industrial - Oficina Altamira
....
1111102              Caja Chica - Caracas
1111103              Caja Chica - Valencia
....
41003                Descuentos por pronto pago
111                  DISPONIBLE
....
5                    EGRESOS
....
52002                GASTOS DE ADMINISTRACION
....
2110502              Impuesto al Valor Agregado IVA
....
21106004             Jorge Marcano
11211002             La Casa del Clone, s.r.l.
....
41001                Venta de Materiales
41                   VENTAS
52002008             Viáticos y Gastos de Representación
```

Tendremos la tabla ordenada por nombres.

Para este ordenador (nombre), en la tabla de cuentas de nuestro ejemplo tendremos:

```
Primer registro :      1                ACTIVO
Ultimo registro:    52002008          Viáticos y Gastos de Representación
```

Para este caso coincidió que el primer registro ordenado por código es igual al primer registro ordenado por nombres; pero esto es sólo una particularidad de este caso específico.

En ciertos casos necesitamos filtrar una tabla para presentar solamente ciertos y determinados registros; por ejemplo: la tabla de **cuentas** trae una columna denominada **tipo** que indica si la cuenta es totalizadora o auxiliar; si la columna **tipo** vale cero, la cuenta es auxiliar; si la columna vale uno, la cuenta es totalizadora. Usaremos la función **filtro** para obtener dos informes; uno sólo con las cuentas totalizadoras y otro sólo con las cuentas auxiliares; veamos como:

```
// En esta versión imprimimos las cuentas totalizadoras
^ cuentas=abre
^ cuentas.filtro=(tipo=1)
.cd ; cuentas.codigo ; 20
.nm ; cuentas.nombre ; 40
TABLA DE CUENTAS
-----
Código                Nombre de la cuenta
-----

.mientras cuentas fin <>
 @cd                  @nm
^ cuentas=avanza
.fin
```

El resultado será lo siguiente:

```
TABLA DE CUENTAS
-----
Código                Nombre de la cuenta
-----
1                    ACTIVO
11                   ACTIVO CIRCULANTE
111                  DISPONIBLE
1111                 DISPONIBLE EN CAJA Y BANCOS
11111                DISPONIBLE EN CAJA
....
11212                CUENTAS POR COBRAR NO COMERCIALES
1122                 OTRAS CUENTAS POR COBRAR
113                  REALIZABLE
....
51                   COSTO DE VENTA
52                   GASTOS DE OPERACIONES
52001                GASTOS DE VENTAS
52002                GASTOS DE ADMINISTRACION
52003                GASTOS DE DEPRECIACION
52004                GASTOS DE INTERESES Y SEGUROS
52005                OTROS GASTOS NO RECURRENTE
```

Obtendremos solamente las cuentas totalizadoras. Ajustemos el programa de la forma siguiente y repitamos el informe:

```
// En esta versión imprimimos las cuentas auxiliares
^ cuentas=abre
^ cuentas.filtro=(tipo=0)
.cd ; cuentas.codigo ; 20
.nm ; cuentas.nombre ; 40
TABLA DE CUENTAS
-----
Código                Nombre de la cuenta
-----
. mientras cuentas fin <>
  @cd                @nm
^ cuentas=avanza
.fin
```

El resultado será lo siguiente:

```
TABLA DE CUENTAS
-----
Código                Nombre de la cuenta
-----
1111102              Caja Chica - Caracas
1111103              Caja Chica - Valencia
1111121              Caja Principal - Caracas
....
2110101              SumiComputación, c.a.
2110102              Maxi-Computers, c.a.
2110110              Oficentro, c.a.
....
311                  Capital Social
312                  Menos: Capital Social No Pagado
321                  Utilidades No Distribuidas año 1995
33002                Utilidad del Ejercicio 1.996
33900                Utilidad del Ejercicio Corriente
....
41001                Venta de Materiales
41002                Rebajas y Devoluciones en Ventas
....
51001                Inventario Inicial de Materiales
51002                Compras de Materiales
5200501              Ayudas y Donaciones
5200502              Exposiciones y Eventos
```

Obtendremos solamente las cuentas auxiliares.

Cuando hemos asignado un filtro a una tabla y deseamos eliminarlo usamos la siguiente instrucción:

```
^ cuentas.filtro=nulo
```

En muchas ocasiones necesitamos posicionar una tabla en un registro determinado que sabemos que existe; por ejemplo: deseamos posicionar la tabla de cuentas en el registro de la cuenta:

1111121 Caja Principal – Caracas

Para ello utilizamos la siguiente instrucción:

```
^ cuentas.codigo=1111121
```

Esta instrucción colocará el apuntador de registros de la tabla de cuentas en el registro o fila que corresponde a la cuenta cuyo código es 1111121. Observe que la instrucción indica: **^ tabla.columna=valor**.

Si el registro de la instrucción anterior no existe en la tabla, **LOPI** dará un error y abandonará la ejecución del programa al no conseguir el registro solicitado por el programador. No obstante lo anterior, es posible verificar previamente si un registro existe antes de ejecutar el posicionamiento en la tabla; por ejemplo:

```
.micuenta; '111121'
.existe; ^cuentas.codigo=@micuenta
.si @existe
^ cuentas.codigo=@micuenta
La tabla de cuentas está ahora ubicada en el código @micuenta
.sino
La cuenta @micuenta no existe en la tabla de cuentas
.fin
```

En este ejemplo definimos la variable **existe** como una variable lógica cuyo valor puede ser CIERTO o FALSO; la instrucción **^cuentas.codigo=@micuenta** retorna CIERTO, si el código contenido en la variable **micuenta** existe en la tabla de cuentas y retorna FALSO, si el código contenido en la variable **micuenta** no existe en la tabla de cuentas.

De esta forma, ya sabiendo que **existe** es CIERTO (existe el código contenido en **micuenta**), se posiciona la tabla utilizando la instrucción **^ cuentas.codigo=@micuenta**. Es importante saber diferenciar las dos líneas de código; en:

```
.existe; ^cuentas.codigo=@micuenta
```

Se está definiendo o calculando una variable; el código va en **rojo** (definición o cálculo de variable) y no hay separación entre el caracter ^ y el nombre de la tabla.

En:

```
^ cuentas.codigo=@micuenta
```

Se está posicionando una tabla; el código va en **verde** (posicionamiento de tablas) y hay un espacio en blanco entre el caracter ^ y el nombre de la tabla.

LOPI permite acceder un registro en una tabla utilizando hasta 2 valores aplicados a dos columnas o campos de la misma; por ejemplo: en la tabla de **documentos** sus dos primeras columnas son: **codigo** y **numero**; la primera columna contiene el tipo al cual pertenece un documento (por ejemplo, una factura de cliente podría ser: 'FC'); la segunda columna, el número del documento (por ejemplo, si el registro corresponde a la factura de cliente 102922, la primera columna contendrá 'FC' y la segunda '102922').

Para ubicar un registro con esta condición (requiere ser accedido a través de dos columnas), el código **LOPI** sería el siguiente:

```
^ documentos=abre
.tipo; 'FC'; 4
.nro; '102922'; 12
.existe; ^documentos.codigo=@tipo;numero=@nro
.si @existe
^ documentos.codigo=@tipo;numero=@nro
La tabla de documentos está ahora ubicada en el documento: @tipo @nro
.sino
El documento: @tipo @nro no existe!!
.fin
```

Observe como se separan con punto y coma las dos columnas de la tabla, para el posicionamiento de tablas utilizando valores sobre dos columnas.

En ciertos casos es preferible dejar que el usuario tenga la posibilidad de seleccionar un registro de una tabla a su criterio, es decir, que el escoja el registro en el cual debe posicionarse la tabla. La instrucción ventana cuando se aplica a una tabla, abre una ventana o forma al usuario, de forma que este posicione el cursor en el registro sobre el cual desea operar; veamos esto:

^ **documentos=abre**

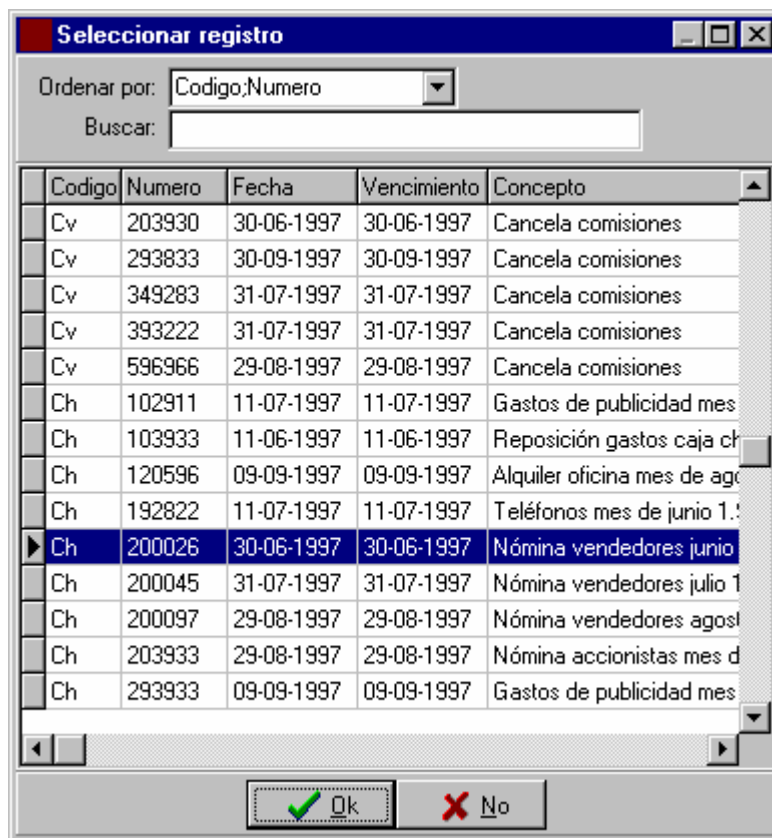
.tipo; documentos.codigo; 4

.nro; documentos.numero; 12

^ **documentos=ventana**

La tabla de documentos está ahora ubicada en el documento: **@tipo @nro**

Cuando se ejecuta este código **LOPI** el programa abrirá una ventana conteniendo la tabla de documentos al ejecutar la instrucción ^ **documentos=ventana**; desde esta ventana el usuario puede escoger el registro que desea. En el ejemplo el usuario escoge el documento **Ch 200026**.



Al hacer clic en Ok se cierra la ventana y se termina de ejecutar el código; **LOPI** imprimir la línea final del programa y el resultado será:

La tabla de documentos está ahora ubicada en el documento: **Ch 200026**

Como puede observarse la selección del registro en la tabla de **documentos** fue hecha por el usuario, en línea durante la corrida del programa.

1.4.6. Consultas SQL

Cuando deseamos acceder tablas que no pertenecen al sistema **Opal** (tablas diseñadas y llenadas por el usuario empleando otros sistemas) o deseamos efectuar una consulta SQL sobre las tablas de **Opal**, sobre las tablas de otros sistemas o sobre las tablas de **Opal** y otros sistemas combinados, utilizamos la herramienta de **LOPI** que permite definir una consulta SQL. Veamos esto:

Notas:

- Para información y un entrenamiento breve sobre SQL vea el capítulo 5.0. del manual de **Opal**.
- Las tablas que pertenezcan a otros sistemas deberán ser tablas **PARADOX** (vea el capítulo 6.0. del manual de **Opal** para mas información) y deberán ubicarse en el directorio de datos de la empresa que está siendo procesada para que **LOPI** pueda accederlas a través de consultas SQL.

? sql=miconsulta select * from cuentas

Esta instrucción **LOPI** genera una consulta SQL y le asigna el nombre **miconsulta**. A partir de esta instrucción la consulta **miconsulta** puede ser accesada como una tabla cualquiera con el nombre: **miconsulta**.

Supongamos que deseamos obtener un informe de la tabla de clientes de **Opal** indicando el total en débitos (ventas) en un determinado período. El comando SQL que hace una consulta de este tipo sería:

```
select nombre,sum(d.monto) as monto
from clientes,documentos d
where d.debito=codigo group by nombre
```

Para llevar esta consulta a **LOPI** escribimos el siguiente programa:

```
? sql=ventas select nombre,sum(d.monto) as monto from clientes,documentos d where d.debito=codigo group by nombre
.nombre; ventas.nombre; 40
.monto; ventas.monto; ###,###,##0.00
? ventas=inicio
VENTAS POR CLIENTE
-----
```

```
NOMBRE DEL CLIENTE _____ VENTAS
```

```
.mientras ventas fin <>
@nombre @monto
? ventas=avanza
.fin
```

Al ejecutarlo se obtiene el siguiente informe:

```
VENTAS POR CLIENTE
-----
```

```
NOMBRE DEL CLIENTE _____ VENTAS
InfoComp, c.a. 4.420.560,00
La Casa del Clon, s.r.l. 3.844.010,00
MicroComputer Store, c.a. 3.938.900,00
Redes y Computos, c.a. 6.143.190,00
```

Observe que exceptuando el hecho de que se usa el caracter **?** en lugar del caracter **^**, el manejo de una consulta SQL es similar al de una tabla. De hecho las siguientes instrucciones son válidas para las consultas SQL:

? consulta=inicio
? consulta=fin
? consulta=avanza
? consulta=retrocede
? consulta=marca
? consulta=retorna
? consulta.filtro=(clase=1)
? consulta.filtro=nulo
? consulta=inicio
.nrregistros; consulta.registros
? consulta.codigo=@cuenta
.existe; ?consulta.codigo=@cuenta
? consulta.codigo=@cuenta;numero=@nro

Las siguientes instrucciones son exclusivas de las tablas y por tanto no aplican a las consultas SQL:

^ tabla=abre
^ tabla=ventana

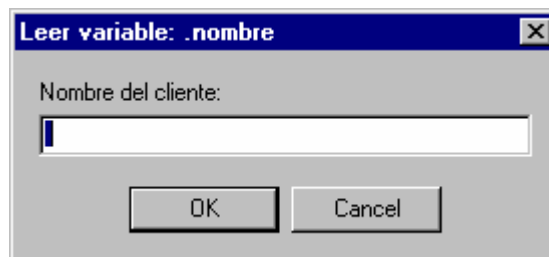
1.4.7. Entrada de datos por pantalla

En ciertas circunstancias es necesario que el valor inicial de una variable sea ingresado por el usuario al momento de la ejecución de un programa; **LOPI** ofrece una instrucción para dar entrada a variables a través de la pantalla; veamos:

.nombre; ?; 50; Nombre del cliente;; ‘ ‘

La instrucción consta de cinco partes separadas cada una por punto y coma; la primera parte indica el nombre de la variable; en la segunda parte se coloca el símbolo de interrogación **?** para indicar que el valor de la variable debe consultarse al usuario; la tercera parte es la máscara de impresión de la variable (50 caracteres de ancho en el ejemplo); la cuarta parte es el texto que deberá imprimirse en la pantalla para ayudar al usuario a identificar que dato se está solicitando; la última parte es un valor por omisión que se da a la variable. Este valor será impreso en la pantalla de solicitud de datos para que el usuario lo acepte o modifique a su gusto. Es importante señalar que el tipo del valor por omisión define el tipo de la variable leída. En el ejemplo la variable es de caracteres.

Al ejecutar esta instrucción **LOPI** presentará en la pantalla una ventana para que el usuario ingrese el valor que se asignará a la variable **nombre**; veamos:

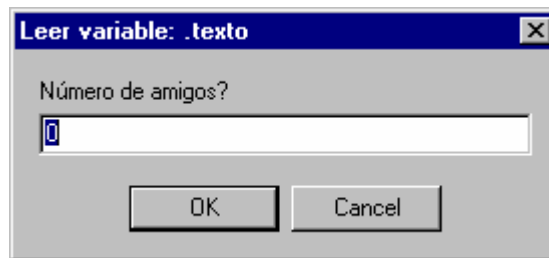
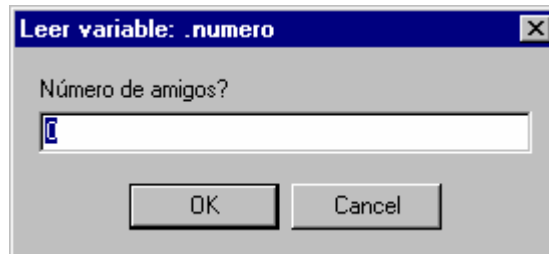


Observe que la forma indica en el título: **Leer variable: .nombre**; indicando que se está solicitando información para asignar a la variable **nombre**. Abajo, encima de la casilla de entrada de datos aparece el mensaje: **Nombre del cliente;;** éste ayuda al usuario a saber que dato está solicitando el programa. La casilla aparece con un espacio en blanco seleccionado (en azul) para que el usuario de entrada al dato solicitado. En este ejemplo usamos ‘ ‘ como valor por omisión para establecer que la variable es de caracteres.

Esto último es muy importante; cómo ya dijimos el valor por omisión define el tipo de la variable; por ejemplo:

.numero; ?; ##0; Número de amigos?; 0
.texto; ?; 10; Número de amigos?; '0'

Estas dos instrucciones generarán en la pantalla la misma ventana de solicitud de datos:



Observe que en la primera se lee la variable **.numero**; el valor que el usuario ingrese será tratado como un valor numérico; en la segunda, aunque muy parecida, se espera un valor de caracteres y lo que el usuario ingrese será tratado como una variable de caracteres.

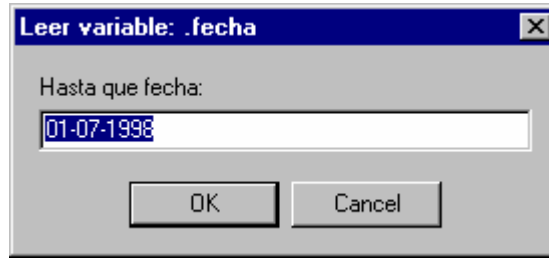
Si en el primer caso el usuario ingresa caracteres distintos de 0, 1, 2...9 y el signo menos, el programa los aceptará, pero más adelante cuando pretenda usar estos valores mal asignados generará un error. En el segundo caso, cualquiera sean los caracteres que el usuario ingrese, numéricos o alfanuméricos, los mismos serán tratados como un valor de caracteres y el programador no debe pretender usarlos posteriormente para realizar cálculos numéricos con ellos.

Nota: entre los planes de mejoras y nuevas facilidades a ser implementadas a futuro en **LOPI** está la de permitir más control del programador para determinar si los datos ingresados son válidos; esto dentro de una misma instrucción. Por ahora, el programador puede leer los datos y asignarlos a la variable, después programar la validación de éstos y si no son correctos, insistir en abrir nuevamente la forma de lectura de datos alertando previamente al usuario sobre el error cometido.

Para dar entrada a datos de fechas no olvide utilizar el carácter & para indicar que el dato esperado es una fecha. Por ejemplo:

.fecha; ?; dddd; Hasta que fecha; &01-07-1998

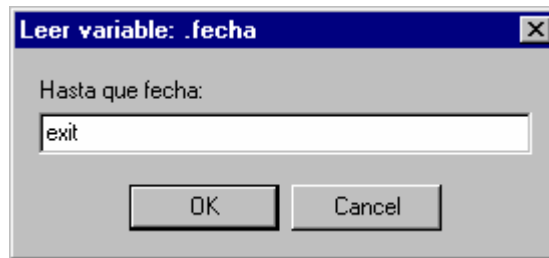
Cuando se ejecute esta instrucción **LOPI** abrirá la forma de lectura de datos siguiente:



Es importante indicar que el valor que se asigna por omisión puede ser a su vez el que esté contenido en una variable; por ejemplo:

.desde; &01-01-1998
.hasta; &31-12-1998
.desdefecha; ?; dddd; Desde que fecha;; @desde
.desdefecha; ?; dddd; Hasta que fecha;; @hasta

Finalmente para terminar este aparte hay que informar a los usuarios (los programadores deben informar a los usuarios) que hacer clic en el botón cancelar no cancela la ejecución e impresión de un informe **LOPI**. Si el usuario desea lograr este efecto debe entrar la palabra **exit** en la forma y hacer clic en el botón OK. Esto efectivamente suspende la ejecución del programa y regresa al sistema **Opal**.



1.4.8. Formatos especiales de impresión

En este aparte veremos algunos formatos de impresión que permiten obtener efectos especiales al imprimir documentos; veamos primero la impresión de **montos escritos**:

^ documentos=abre
.codigo; documentos.codigo; 4
.numero; documentos.numero; 10
.monto; documentos.monto; ###,###,##0.00
.monto_texto; documentos.monto; texto:50

^ documentos=ventana
 El documento: **@codigo @numero** tiene un monto de: **@monto**
 Este monto en texto es: **@monto_texto**

Al ejecutar esta sección de código **LOPI** el programa abrirá una forma para que el usuario escoja un documento; suponiendo que escoge el documento **Cp 194844** se obtendrá:

El documento: **Cp 194844** tiene un monto de: **2.202.083,00**
 Este monto en texto es: **DOS MILLONES DOSCIENTOS DOS MIL OCHENTA Y TRES BOLIVARES SIN CENTIMOS**

Observe que la variable **monto_texto** tiene un formato de impresión: **texto:50**. Esto significa que será impresa convirtiendo el monto en texto a **montoescrito** y con un ancho de impresión de 50 caracteres. **LOPI** ajustará la impresión para respetar este ancho e imprimirá 1, 2, 3 o más líneas de este ancho, dependiendo de la magnitud de la cifra, para obtener una cifra en montoescrito como se muestra en el ejemplo.

Si el programador no indica ancho de impresión (coloca como formato de impresión sólo la palabra **texto**), **LOPI** utilizará un ancho de impresión por omisión de 40 caracteres de ancho.

En el ejemplo se usó como divisa “**BOLIVARES**” y cómo centésimas “**CENTIMOS**”. Estos valores pueden ser cambiados en la forma de propiedades de **Opal**; siga la ruta Servicios-Propiedades-Generales del menú principal del software:



Con este ajuste el ejemplo anterior dará:

```
El documento: Cp 194844 tiene un monto de: 2.202.083,00
Este monto en texto es: DOS MILLONES DOSCIENTOS DOS MIL OCHENTA Y TRES
                        PESOS MEXICANOS SIN CENTAVOS
```

En otros casos la longitud de una columna o campo de una tabla o el valor de una variable de caracteres es muy largo y no puede imprimirse en una línea; por ejemplo, ciertas columnas de las tablas de **Opal** contienen campos tipo **memo** los cuales son abiertos, es decir, el usuario puede llenarlos libremente e ingresar el número de caracteres que desee.

Un ejemplo de este tipo de datos es el contenido en la columna **memo** de la tabla de **clientes**; en esta columna el usuario puede escribir libremente información relativa al cliente. Veamos un ejemplo de un programa para obtener de la tabla de clientes un listado del código comercial de los mismos, su nombre, experiencia y comentarios:

```

^ clientes=abre
.nombre; clientes.nombre; 30
.experiencia; clientes.experiencia
.t_exp; ' '; 20
.comentarios; clientes.memo; 40; memo
INFORMACION DE CLIENTES

Estos datos son confidenciales para uso de la gerencia *
-----*
Nombre del cliente      Experiencia  Comentarios *

.mientras clientes fin <>
.si @experiencia 0 =
~t_exp; 'Pésima'
.fin
.si @experiencia 1 =
~t_exp; 'Mala'
.fin
.si @experiencia 2 =
~t_exp; 'Regular'
.fin
.si @experiencia 3 =
~t_exp; 'Buena'
.fin
.si @experiencia 4 =
~t_exp; 'Muy buena'
.fin
.si @experiencia 5 =
~t_exp; 'Excelente'
.fin
@nombre                @t_exp      @comentarios

^ clientes=avanza
.fin

```

Al ejecutar este código **LOPI** obtenemos el informe de la página siguiente; observemos lo siguiente:

1. La columna **memo** de la tabla de **clientes** que se asigna a la variable **comentarios** es del tipo memo y por tanto podrá tener una longitud indeterminada; no obstante, con la instrucción de impresión: **40; memo** indicamos a **LOPI** que es un campo memo de longitud indeterminada y que debe ser impreso con un ancho de 40 caracteres; el número de líneas que **LOPI** imprimirá dependerá de la longitud real del texto contenido en cada registro de clientes.
2. Observe el manejo de la columna **experiencia** de la tabla de **clientes**; esta columna contiene un valor numérico entre 0 y 5. Dependiendo de su valor tendremos que la experiencia de la empresa con el cliente es: 0 = pésima, 1 = mala, 2 = regular, 3 = buena, 4 = muy buena, 5 = excelente. Usando varias instrucciones **.si** se logra inicializar el valor de la variable auxiliar **t_exp** para asignarle el valor correcto dependiendo de la experiencia del cliente en la base de datos de la empresa.

INFORMACION DE CLIENTES

Estos datos son confidenciales para uso de la gerencia *

Nombre del cliente	Experiencia	Comentarios
MicroComputer Store, c.a.	Buena	Este cliente está recomendado por el vice-presidente y trabaja con nosotros desde el año 1.990.
La Casa del Clone, s.r.l.	Regular	1.- Cliente con buena experiencia (10-02-1.996). 2.- Vice-presidencia de Finanzas ordenó hacer un seguimiento cuidadoso a los pagos de este cliente pues ellos conocen de informaciones de la Bolsa de Valores que tiene ciertos problemas internos de liquidez (15/05/1.997). Hablar con Pedro Rojas en Finanzas.
InfoComp, c.a.	Buena	1.- Recomendado por el Sr. Hernández del Banco Amazonas. No hay experiencia previa. Se les aprobó un crédito de 2 millones en base a estados financieros.
Redes y Computos, c.a.	Mala	1.- El presidente de esta firma es una persona conflictiva y complicada. Tratar con "pinzas" las negociaciones con ellos.

En no pocos casos, sobre todo en la impresión de facturas, es necesario imprimir combinaciones de cifras y números alineadas hacia la derecha. En estos casos se requiere de algunas instrucciones especiales que nos ayudan a lograr el efecto deseado; veamos este ejemplo:

```

^ documentos=abre
^ documentos=ventana
.es; ' '
.iva; 'I.C.S.V.M.'
.monto; documentos.monto; ##,###,##0.00
.ts; 16,5; #0.0%
.total_iva; @monto @ts * 100 /; ##,###,##0.00
.total_pagar; @monto @total_iva +; ##,###,##0.00
.texto_pagar; @total_pagar; texto:60
.texto_tasa; @ts; caracter:#0.0%
.texto_iva; @texto_tasa @es @iva + +; %40s

                                TOTAL ESTA FACTURA: @monto
                                : @total_iva
                                =====
                                TOTAL A PAGAR: @total_pagar

Son: @texto_pagar

```

Al ejecutar esta sección de código **LOPI** el programa abrirá una ventana para que el usuario escoja un documento; suponiendo que seleccione la factura **FC 000002**, obtendremos el resultado presentado en la siguiente página:

TOTAL ESTA FACTURA:	450.000,00
16,5% I.C.S.V.M.:	74.250,00
	=====
TOTAL A PAGAR:	524.250,00

**Son: QUINIENTOS VEINTE Y CUATRO MIL DOSCIENTOS CINCUENTA
BOLIVARES SIN CENTIMOS**

La variable **es** es un espacio en blanco; **iva** es el nombre del impuesto aplicado (en Venezuela significa Impuesto al Consumo Suntuario y Ventas al Mayor). La variable **monto** es el monto del documento que selecciona el usuario. La variable **ts** es la tasa del impuesto; para el ejemplo se fija en 16,5%.

La variable **total_iva** será igual al monto de la factura por la tasa dividido por 100; la variable **total_pagar** será el monto de la factura mas el monto del impuesto; la variable **texto_pagar** permite imprimir el total a pagar en montoescrito con un ancho de 60; la variable **texto_tasa** retorna el valor de la tasa **ts** en caracteres con un formato #0.0% (si la tasa es 16,5 retornará 16,5%). Observe como se utiliza la instrucción **caracter:#0.0%** para convertir el valor numérico a caracteres.

Requerimos la variable **ts** en caracteres (no en número) para sumarla a un espacio en blanco y al nombre del impuesto en la variable **iva**, e imprimirla con un ancho de 40 caracteres "pero alineada a la derecha"; esto se logra a través de la variable **texto_iva** con la máscara **%40s** de alineación a la derecha.

Es importante colocar la impresión de la variable **texto_iva**, que se hace con la instrucción **@texto_iva**, exactamente a 40 caracteres a la izquierda de los dos puntos ":" usados para marcar el inicio de los montos en números.

1.4.9. Funciones

LOPI incluye un pequeño conjunto de funciones que nos permiten manipular los datos y convertirlos en otros datos para facilitar la programación. A continuación haremos un breve recorrido por estas funciones:

Función **hoy**

Esta función permite asignarle a una variable el valor de la fecha de **HOY** (fecha del equipo en Windows); por ejemplo:

.hoy_es; hoy; ddddd
Hoy es: **@hoy_es**

La variable **hoy_es** contendrá el valor de la fecha de hoy.

Función **opal**

Esta función permite asignarle a una variable el valor de la fecha de proceso en el sistema **Opal**; esta fecha es fijada por el usuario por la ruta Datos-Fecha de trabajo del menú principal del software; por ejemplo:

.fecha_proceso; opal; ddddd
La fecha de proceso de Opal es: **@fecha_proceso**

La variable **fecha_proceso** contendrá el valor de la fecha de proceso de **Opal**.

Funciones **dia, mes, año**

La función **dia** permite obtener el valor numérico del día de una fecha; por ejemplo:

.fecha; &15-06-1998

.dia; @fecha dia

El día de la fecha: @fecha es @dia

En este ejemplo la variable **dia** contendrá el número del día (15) de la variable **fecha**.

En igual forma **LOPI** incluye las funciones **mes** y **año** para obtener el valor numérico del mes y año de una fecha. Brevemente funcionan así:

.fecha; &15-06-1998

.dia; @fecha dia

.mes; @fecha mes

.año; @fecha año

El día de la fecha: @fecha es @dia

El mes de la fecha: @fecha es @mes

El año de la fecha: @fecha es @año

Función !

Esta función permite asignar un valor a una variable de fecha partiendo de los valores numéricos de día, mes y año. Veamos algunos ejemplos:

.fecha; 10 11 1998 !

En este ejemplo la variable fecha tendrá el valor 10-11-1998; los parámetros de la función ! son tres números que representan el día, mes y año de la fecha que deseamos crear.

Los valores de día, mes y año pueden estar contenidos en variables; por ejemplo:

.dia; 10

.mes; 11

.año; 1998

.fecha; @dia @mes @año !

Dará el mismo resultado del caso anterior.

Función **primero**

La función **primero** permite obtener la fecha del primer día del mes y año de la fecha al cual se aplica la función; por ejemplo:

.fecha; &15-06-1998

.fecha_primer_dia; @fecha primero

La fecha del primer día del mes de la fecha: @fecha es @fecha_primer_dia

En este ejemplo la variable **fecha_primer_dia** contendrá la fecha **01-06-1998**.

Función **ultimo**

La función **ultimo** permite obtener la fecha del último día del mes y año de la fecha al cual se aplica la función; por ejemplo:

```
.fecha; &15-06-1998
```

```
.fecha_ultimo_dia; @fecha ultimo
```

La fecha del último día del mes de la fecha: @fecha es @fecha_ultimo_dia

En este ejemplo la variable **fecha_ultimo_dia** contendrá la fecha **30-06-1998**.

A primera vista pareciera que las funciones **hoy**, **opal**, **dia**, **mes**, **año**, **!**, **primero** y **ultimo** no tienen mayor importancia o utilidad; algunos ejemplos nos permitirán percatarnos de que es exactamente lo contrario.

```
.ultimo_dia; 31 12 opal año !
.hasta; ?; ddddd; Hasta fecha:; @ultimo_dia
.dia; @hasta dia
.mes; @hasta mes
.año; @hasta año 1 -
.desde; @dia @mes @año !
.año1; @desde año; #,###
.año2; @hasta año; #,###
```

En este ejemplo el usuario debe ingresar una fecha "hasta" que define la fecha de cierre de un informe que va a programarse en **LOPI**. La variable **ultimo_dia** se inicializa al último día del año de la fecha del sistema **Opal** y se utiliza como valor predeterminado para leer la fecha "hasta" del informe.

Veamos como se asigna el valor a la variable **ultimo_dia**: utilizamos la función **!**. Esta función requiere de tres parámetros; el primero es el día de la fecha: en nuestro caso es un valor constante igual a 31. El segundo parámetro es el mes de la fecha; nuevamente colocamos un valor constante igual a 12. El tercer parámetro es el año de la fecha; como queremos que el año sea el mismo que el de la fecha del sistema **Opal** aplicamos la función **año** a la función **opal**; veamos los pasos uno a uno suponiendo que la fecha de proceso de **Opal** sea 20 de mayo de 1.997:

```
.ultimo_dia; 31 12 opal año !
```

Primero la función **opal** retorna la fecha del sistema **Opal**; la instrucción de arriba es equivalente a:

```
.ultimo_dia; 31 12 &20-05-1997 año !
```

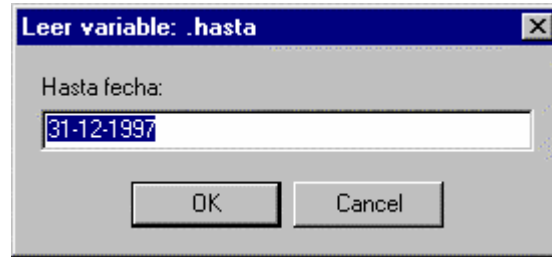
A continuación la función **año** se aplica a la fecha retornada por la función **opal**; el resultado será el valor numérico del año, o sea 1997; es decir, la instrucción de arriba es equivalente a:

```
.ultimo_dia; 31 12 1997 !
```

Finalmente la función **!** se aplica a los tres parámetros 31, 12, 1997: esto debe retornar la fecha del último día del año 1997, es decir: 31-12-1997; este valor se asigna a **ultimo_dia**.

En la siguiente instrucción se utiliza **ultimo_dia** como valor por omisión para que el usuario ingrese la fecha de cierre del informe; al ejecutarse la instrucción aparecerá la siguiente ventana:

```
.hasta; ?; ddddd; Hasta fecha:; @ultimo_dia
```



Este es el efecto que deseamos; que el valor por omisión sea el último día del año del sistema; supongamos ahora que el usuario ingresa como fecha **30-09-1997** y hace clic en OK. Las siguientes tres instrucciones extraen el día, mes y año de la fecha ingresada por el usuario. Para nuestro ejemplo se cumplirá:

```
.dia; @hasta dia
.mes; @hasta mes
.año; @hasta año 1 -
```

La variable **dia** valdrá 30; la variable **mes** valdrá 9; la variable **año** valdrá 1996 = 1997-1; observe como se resta 1 al valor retornado por la función **año** aplicada a la variable **hasta**.

En las siguientes instrucciones asignamos a la variable **desde** el valor de un año "antes" de la fecha "hasta"; la variable **año1** se inicializa a 1996 y la variable **año2** se inicializa a 1997. Suponemos que todos estos valores son requeridos por el programador para poder diseñar su informe.

```
.desde; @dia @mes @año !
.año1; @desde año; #,###
.año2; @hasta año; #,###
```

En el aparte **1.4.14.-** se especifican ejemplos de programas que hacen buen uso de estas instrucciones.

Otro ejemplo nos muestra el uso de las funciones **hoy**, **primero** y **ultimo**:

```
.primer_dia; hoy primero
.ultimo_dia; hoy ultimo
.desde; ?; dddd; Desde fecha:; @primer_dia
.hasta; ?; dddd; Hasta fecha:; @ultimo_dia
```

En forma similar al ejemplo anterior, en esta sección de código se utilizan las funciones **primero** y **ultimo** para asignar los valores por omisión a la lectura de las variables **desde** y **hasta**; la variable **primer_dia** se inicializa a la fecha del primer día del mes y año del día de hoy. Si hoy es 13-11-1998, entonces la variable **primer_dia** será inicializada a 01-11-1998.

En la misma forma la variable **ultimo_dia** será inicializada al último día del mes y año del día de hoy; para nuestro ejemplo esta fecha será 30-11-1998.

Función **longitud**

La función **longitud** permite obtener la longitud en caracteres de una variable; por ejemplo:

```
.direccion; clientes.direccion
.longdir; @direccion longitud
```

Si la dirección del cliente apuntado en este momento en la tabla de clientes tiene una longitud o largo de 114 caracteres, el valor 114 será asignado a la variable **longdir**.

Función **espacio**

La función **espacio** permite obtener la posición del último carácter que no sea espacio en una variable, contado desde una determinada posición a la izquierda; por ejemplo:

.direccion; clientes.direccion
.inx; @direccion 50 espacio

Para explicar esto vamos a suponer que la dirección del cliente contenida en la columna **clientes.direccion** es la siguiente:

```
Avenida Nicaragua, Edificio Colombia, Piso 3, Oficina 3-1, Urbanización Industrial Matanzas.  
123456789012345678901234567890123456789012345#789+123456789012345678901234567890123456789012
```

Arriba se indica la dirección del cliente; abajo marcamos cada carácter indicando su posición.

Al aplicar la función **espacio** a la variable **direccion** vemos que hace **LOPI**; primero se ubica en la posición 50 de la cadena de caracteres contenida en la variable **direccion**. Esta posición está marcada arriba con un signo + en azul y corresponde a la letra “c” de la palabra “Oficina”. A continuación **LOPI** verifica el carácter a la derecha del carácter #50; si este carácter es espacio, entonces el último carácter de la cadena de caracteres que no es espacio sería el #50. La función retornaría 50 como resultado.

Pero en nuestro ejemplo no ocurre así; el carácter de la derecha no es espacio, sino “i”. Dado que aquí no se cumple la condición requerida, entonces **LOPI** comienza a avanzar de derecha a izquierda partiendo del carácter #50 hasta que consigue el primer carácter espacio. Este será el carácter #46 correspondiente al espacio que hay entre la coma del número 3 de “Piso 3” y la palabra “Oficina”; éste carácter está marcado en la figura con el símbolo # en rojo.

A la izquierda de este carácter no hay más caracteres espacio seguidos de éste; por tanto el último carácter de esta cadena de caracteres, que no es espacio y que está a la izquierda de la posición #50 es precisamente la coma de la posición #45; este será el valor que se asignará a la variable **inx**.

Más abajo veremos una aplicación práctica de esta función y porque es tan importante que nos hayamos tomado la molestia de explicar cuidadosamente como trabaja.

Función **copiar**

La función **copiar** permite copiar de una cadena de caracteres “n” caracteres, contados a partir de la posición “p”; el mismo ejemplo anterior aclara la circunstancia:

.direccion; clientes.direccion
.dir1; @direccion 1 30 copiar
.dir2; @direccion 31 30 copiar
.dir3; @direccion 61 30 copiar
.dir4; @direccion 91 30 copiar

La variable **dir1** será inicializada con los caracteres del 1 al 30 de la variable **direccion**; la variable **dir2** con los caracteres del 31 al 60, la variable **dir3** con los caracteres del 61 al 90 y la variable **dir4** con los caracteres del 91 al 120. Suponiendo que la dirección del cliente es la siguiente:

```
Avenida Nicaragua, Edificio Colombia, Piso 3, Oficina 3-1, Urbanización Industrial Matanzas.  
12345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012
```

dir1 será entonces: Avenida Nicaragua, Edificio Co
dir2 será entonces: lombia, Piso 3, Oficina 3-1, U
dir3 será entonces: rbanización Industrial Matanza
dir4 será entonces: s.

Esto no es muy práctico; si quisieramos escribir la dirección del cliente en cuatro líneas quedaría así:

```
@dir1
@dir2
@dir3
@dir4
```

```
Avenida Nicaragua, Edificio Co
lombia, Piso 3, Oficina 3-1, U
rbanización Industrial Matanza
s.
```

Abajo veremos que para obtener el efecto deseado es mejor utilizar la función **extraer**. No obstante hay casos en que necesitamos utilizar en forma combinada las funciones **espacio** y **copiar** para resolver una determinada situación; pero veamos primero la función **purgar** que también necesitaremos para el ejemplo que viene a continuación.

Función **purgar**

La función **purgar** permite purgar los caracteres en blanco que se encuentran a la izquierda y a la derecha de una variable de caracteres; por ejemplo:

```
.punto; '.'
.texto1; 'Este es el texto1,'
.texto2; ' Este es el texto que deseamos purgar de caracteres en blanco '
.texto_completo; @texto1 @texto2 + @punto +
.texto_purgado; @texto1 @texto2 purgar + @punto +
```

La variable **texto_completo** se obtiene sumando las variables de caracteres **texto1**, **texto2** y **punto** sin purgar previamente la variable **texto2**; el resultado será el siguiente:

```
Este es texto1, Este es el texto que deseamos purgar de caracteres en blanco .
```

Observe los espacios en blanco antes y después del texto: "Este es el texto que deseamos".

La variable **texto_purgado** se obtiene en la misma forma sumando las variables **texto1**, **texto2** y **punto**, pero purgando previamente con la función **purgar**, la cual se aplica a la variable **texto2**, los caracteres en blanco a la izquierda y la derecha del texto contenido en la variable; el resultado ahora será el siguiente:

```
Este es texto1,Este es el texto que deseamos purgar de caracteres en blanco.
```

Observe que los espacios en blanco ahora fueron purgados.

Vayamos ahora a un ejemplo que utiliza las funciones **longitud**, **espacio**, **copiar** y **purgar**.

El diseño de un modelo determinado de cheque exige que la impresión de la cifra en texto, para que quepa, sea hecha en dos líneas que no se encuentran a su vez alineadas en la misma columna; observe en la siguiente figura que la línea superior de la cifra en texto está mas a la derecha que la línea inferior.

Bs. 2.234.520,50

Páguese a
la orden de: LUIS EDUARDO GOMEZ MARCANO

La cantidad de: DOS MILLONES DOSCIENTOS TREINTA Y CUATRO MIL
QUINIENTOS VEINTE BOLIVARES CON 50/100 CENTIMOS

Caracas, 10 de octubre de 1.998

Firma y sello

El código **LOPI** para hacer esto es el siguiente:

```
^ documentos=abre
^ documentos=ventana
.monto; documentos.monto; ##,###,##0.00
.beneficiario; documentos.usuario1; 50
.montoescrito; documentos.monto; texto
.lenmontoescrito; @montoescrito longitud
.i; @montoescrito 50 espacio
.cifra01; @montoescrito 1 @i copiar; 50
.cifra02; ' '; 70
.si @lenmontoescrito 50 >
~i; @i 1 +
~cifra02; @montoescrito @i 70 copiar purgar; 70
.fin
.dia; hoy; d
.mes; hoy; mmmm
.año; hoy; yyy
```

Bs. @monto

Páguese a
la orden de: @beneficiario

La cantidad de: @cifra01
@cifra02

Caracas, @dia de @mes de @año

Firma y sello

Explicación del código **LOPI**:

Después de abrir y posicionar un registro de cheque en la tabla de documentos, se definen las variables para manejar el monto, beneficiario, montoescrito, día, mes y año del cheque.

Como el montoescrito es muy largo se dividirá en dos partes que irán ubicadas como "cifras en letras" en dos líneas en el cheque mediante las variables **cifra01** y **cifra02**. En **cifra01** se colocan los primeros 50 caracteres del montoescrito. En **cifra02** el resto de los caracteres; para saber cuantos caracteres mide la variable **montoescrito** se usa la variable **lenmontoescrito**. Supondremos para este ejemplo que la longitud total de la cifra en texto no supera los 120 caracteres.

Para dividir la variable **montoescrito** en **cifra01** y **cifra02** primero determinamos en la variable **i** el último carácter que no es espacio en los primeros 50 caracteres de montoescrito con la función **espacio**. Copiamos esta primera parte en la variable **cifra01** utilizando la función **copiar**.

La variable **cifra02** la inicializamos a ' '. A continuación verificamos si **montoescrito** es mayor de 50 mediante la variable **lenmontoescrito**. Si lo es, incrementamos **i** para que apunte al siguiente carácter a la derecha de la parte que ya copiamos en **cifra01**, y copiamos en **cifra02** desde este carácter hasta un máximo de 70 caracteres. La función **purgar** purga los caracteres en blanco a la izquierda que hayan podido quedar en **cifra02** para que quede totalmente alineada a la izquierda.

El valor de la variable **beneficiario** (beneficiario del cheque) se obtiene a través de la columna **usuario1** de la tabla de documentos. Esta columna se reserva en la tabla de documentos para que el usuario pueda asignarle un nombre y utilizarla para incluir información necesaria; en este caso se usa para guardar el nombre del beneficiario del cheque.

El resultado de ejecutar esta sección de código **LOPI** y seleccionar un cheque en la tabla de documentos será el siguiente:

Bs. 72.230,00

Páguese a
la orden de: **C.A. La Electricidad de Caracas**

La cantidad de: **SETENTA Y DOS MIL DOSCIENTOS TREINTA BOLIVARES SIN CENTIMOS**

Caracas, 14 de noviembre de 1998

Firma y sello

Función **extraer**

La función **extraer** permite extraer de una cadena de caracteres los siguientes “n” caracteres que vienen, truncando a su vez a la variable al cuál se le aplicó la función de estos caracteres y tomando en consideración el de no cortar palabras; arriba quedamos pendientes en como resolver la impresión de la dirección de un cliente que puede ser muy larga, en líneas de 30 caracteres como máximo. La solución en **LOPI** es la siguiente:

```
^ clientes=abre  
^ clientes=ventana  
.nombre; clientes.nombre  
.direccion; clientes.direccion  
.linea; ‘ ‘; 30  
@nombre  
.mientras @direccion longitud 0 >  
~linea; @direccion 30 extraer  
@linea  
.fin
```

Al ejecutar esta sección de código **LOPI** y escoger un cliente se obtendrá el siguiente resultado:

InfoComp, c.a.
Avenida Ppal. Los Cortijos de
Lourdes. Edificio Amarillo.
Piso 3. Oficina 32. Los
Cortijos.

Explicación del código **LOPI**: La dirección del cliente es un texto largo y corrido: **Avenida Ppal. Los Cortijos de Lourdes. Edificio Amarillo. Piso 3. Oficina 32. Los Cortijos.**

La variable **direccion** contiene el texto de la columna **direccion** en la tabla de **clientes**; esta columna mide 150 caracteres y se va a imprimir en líneas de 30 caracteres. Esto significa que si la columna mide mas de 30 caracteres se requerirá mas de 1 línea de impresión.

La variable **linea** se recalcula extrayendo cada vez 30 caracteres de la variable **direccion** hasta agotarla. Esto ocurre cuando **direccion** retorna una cadena sin caracteres y su longitud será cero. Se hace un lazo de control mientras la longitud de **direccion** sea mayor que cero.

Observe como la función **extraer** a medida que efectivamente va extrayendo los caracteres en grupos de 30 de la variable **direccion** y asignándolos a la variable **linea**, al mismo tiempo va truncando estos caracteres extraídos de la variable **direccion** y finalmente la agotará dejándola en cero. Observe también que **LOPI** extrae los caracteres procurando no cortar palabras y haciendo los cortes en forma automática por los espacios en blanco existentes entre las palabras que constituyen el texto de la dirección del cliente.

Es muy importante indicar que durante el proceso de extracción de caracteres de la variable **direccion** esta no debe reimprimirse, recalcularse o reutilizarse en cálculos, pues hacerlo recalcularía la variable y la regresaría a su valor original; veamos esto:

```
^ clientes=abre
^ clientes=ventana
.nombre; clientes.nombre
.direccion; clientes.direccion
.linea; ‘ ‘; 30
@nombre
.mientras @direccion longitud 0 >
~linea; @direccion 30 extraer
@linea
@direccion
.fin
```

En este ejemplo, al tratar de imprimir la variable **@direccion** después de la variable **@linea**, hará que **LOPI** recalcule nuevamente el valor de la variable **direccion** buscando su valor inicial en la columna **direccion** de la tabla de **clientes**. Esto efectivamente restituye el valor original de la variable y no permite que **extraer** la agote que es el efecto deseado; al recalcularse no se agotará y el lazo **.mientras** se ejecutará indefinidamente colgando el sistema. El programador **LOPI** deberá estar muy pendiente de esta posibilidad.

Función \$

Para finalizar las funciones **LOPI** veremos una de las funciones mas importantes del macro-lenguaje; la función **saldo de cuenta contable** o función **\$**. Un ejemplo nos indica su uso:

```
.fecha; ?; dddd; Fecha del saldo?; opal
^ cuentas=abre
^ cuentas=ventana
.cuenta; cuentas.codigo
.nombre; cuentas.nombre
.saldo; @cuenta @fecha $; ##,###,##0.00
El saldo de la cuenta: @cuenta : @nombre
A la fecha: @fecha es: @saldo
```

Observe como en la línea que define la variable **saldo** se aplica la función **\$**; esta función requiere dos parámetros, primero la cuenta al cual se le calculará el saldo y luego la fecha al cual el saldo debe ser calculado. Al ejecutar esta sección de código **LOPI** se abrirá una ventana para que el usuario escoja la cuenta que desea calcular; una vez seleccionada se obtendrá el siguiente resultado:

El saldo de la cuenta: **1111103** : Caja Chica - Valencia
A la fecha: 20-09-1997 es: **10.000,00**

La cuenta no tiene que ser necesariamente una cuenta auxiliar o de movimientos; si la cuenta es totalizadora **LOPI** se encarga de calcular su saldo totalizando todas las cuentas auxiliares que constituyen su grupo.

El saldo de la cuenta: **1111** : DISPONIBLE EN CAJA Y BANCOS
A la fecha: 20-09-1997 es: **35.007.517,87**

En este ejemplo se escoge la cuenta totalizadora 1111; se obtiene el mismo resultado.

Para cuentas de balance esta solución está muy bien; pero para las cuentas nominales se requiere normalmente el saldo neto entre dos fechas. Para una cuenta de ingresos del cual queremos saber su movimiento de un período usamos la siguiente solución:

.d; 1 12 opal año !; ddddd
.b; 31 12 opal año !; ddddd
.desde; ?; ddddd; Desde que fecha?; @d
.hasta; ?; ddddd; Hasta que fecha?; @h
^ cuentas=abre
^ cuentas=ventana
.cuenta; cuentas.codigo
.nombre; cuentas.nombre
.saldo; @cuenta @hasta \$ @cuenta @desde \$ -; ##,###,##0.00
El saldo de la cuenta: @cuenta : @nombre
En el período @desde al @hasta es: @saldo

En este caso para calcular el saldo calculamos primero el saldo a la fecha “hasta”, luego el saldo a la fecha “desde” y luego restamos los saldos.

El saldo de la cuenta: **41001** : Venta de Materiales
En el período 01-08-1997 al 31-08-1997 es: **-3.399.560,00**

Las alternativas que ofrece esta función para obtener los mas variados reportes contables y financieros son incontables; mas adelante, en la sección **1.4.14.-** los informes:

- Comparativo de Ganancias y Pérdidas (Contabilidad)
- Indices financieros (Contabilidad)
- Variación en el Capital de Trabajo (Contabilidad)
- Aplicación de Fondos (Contabilidad)
- Uso del crédito : incluye gráfico (Administrativo)
- Partidas del circulante : incluye gráfico (Contabilidad)

En su totalidad son diseñados utilizando como herramienta de trabajo la función **\$**. Invitamos al lector a examinarlos y percatarse por si mismo de las inmensas posibilidades que se abren con el uso adecuado de **LOPI** como lenguaje de programación de informes.

1.4.10. Macros

En ciertos casos una determinada sección de código **LOPI** se requiere para ser utilizada en un programa repetidas veces. No tendría ningún sentido que esta sección, que debe reutilizarse, haya que reescribirla nuevamente en su totalidad. El uso de los macros (pequeñas secciones de código **LOPI** reutilizables) facilitan el diseño de los programas y ayuda a una mejor comprensión y manejo de los mismos. Veamos un ejemplo sencillo de esto:

Queremos generar un informe que liste algunas columnas de nuestro interés de la tabla de documentos; deseamos tener un encabezado del reporte y que cada determinado número de líneas impresas, se repita el encabezado para la siguiente página. Seleccionando adecuadamente el máximo de líneas que deben ser impresas por página podemos controlar los saltos de páginas en la impresora. Veamos el programa en **LOPI**:

```
.macro encabezado
~p; @p 1 +
LISTA DE DOCUMENTOS

Fecha del informe: @hoy
Fecha del sistema: @sistema
Página Nro: @p
Documento   Fecha       Vencimiento           M o n t o
-----

.fin
^ documentos=abre
.hoy; hoy; dddd
.sistema; opal; dddd
.l; 0
.p; 0; 00
.fl; .f.
.maxl; 57
.total; 0; ###,###,##0.00
.cd; documentos.codigo; 3
.numero; documentos.numero; 8
.fecha; documentos.fecha; dddd
.vence; documentos.vencimiento; dddd
.monto; documentos.monto; ###,###,##0.00
.mientras documentos fin <>
.si @l 0 =
&encabezado
.fin
@cd @numero @fecha      @vence      @monto
~l; @l 1 +
~fl; .t.
~total; @total @monto +
.si @l @maxl >=

~l; 0
.fin
^ documentos=avanza
.fin
.si @fl

T O T A L ..... @total
.fin
```

En este ejemplo definimos el macro **encabezado** que utilizaremos para imprimir el encabezado del informe; observe como el macro se define con la instrucción **.macro** abriendo y se finaliza con la instrucción **.fin**.

Esta sección de código no se ejecutará hasta que no se ejecute el macro; es decir, la sección sólo define el macro **encabezado**, pero no lo ejecuta. Para ejecutarlo, el macro debe ser invocado utilizando el carácter **&** seguido del nombre del macro, como se observa mas abajo, en la parte del código que ejecuta el informe propiamente dicho.

El macro al ejecutarse incrementa la variable **p** que cuenta el número de páginas impresas; luego imprime el encabezado. Una vez definido el macro podemos continuar con el programa: se abre la tabla de **documentos**; las variables **hoy** y **sistema** contendrán las fechas del sistema operativo y de proceso de **Opal**. La variable **I** (ele) se utiliza para controlar el número de líneas impresas por página; la variable **p** (pe) controla el número de la página impresa; observe que se utiliza arriba en el macro pero se define posteriormente; esto no importa ya que lo que se requiere es que cuando el macro se invoque ya esté definida.

La variable **fl** es una bandera o policía que se inicializa a FALSO y se activa a CIERTO si hubo impresión en el informe (la tabla de **documentos** no está vacía); si hay impresión de líneas, entonces abajo se imprimirá el total sólo cuando la variable **fl** sea CIERTA.

La variable **maxl** es el máximo de líneas a imprimirse (fuera del encabezado y pie de página); este valor se inicializa a 57 pero puede ajustarse según el tipo de impresora y papel utilizado para lograr el salto de página exactamente al finalizar la hoja física de papel.

La variable **total** será utilizada para totalizar los montos individuales de los documentos; se inicializa a cero al comenzar. Las variables **cd**, **numero**, **fecha**, **vence** y **monto** son las columnas de la tabla de documentos que deseamos imprimir.

La impresión del informe la controla un lazo **.mientras** que se ejecuta mientras la tabla de **documentos** no llegue a su final; al entrar en el lazo verificamos si la variable **I** es igual a cero, y si lo es, imprimimos el encabezado invocando al macro **encabezado**; a continuación imprimimos la línea del informe que incluye las variables de las columnas de la tabla de **documentos**.

Incrementamos la variable **I** en 1 para indicar que imprimimos una nueva línea; hacemos CIERTA a la variable **fl** para indicar que el informe si tiene líneas impresas; totalizamos el monto del documento corriente en la variable **total**.

Verificamos a continuación si hemos alcanzado el máximo de líneas permitidas comparando la variable **I** con la variable **maxl**; si **I** es mayor o igual a **maxl** hemos alcanzado el máximo de líneas; entonces saltamos dos líneas del pie de página y hacemos cero a la variable **I**; esto obligará a imprimir un nuevo encabezado al regresar al siguiente ciclo de la instrucción **.mientras**.

Avanzamos al siguiente registro de la tabla de documentos; regresamos nuevamente a evaluar la condición de ejecución del **.mientras**. Al agotarse los registros de la tabla de **documentos** saldremos del lazo **.mientras**; verificamos si la variable **fl** es CIERTA (se han impreso registros) y si lo es, imprimimos el total del informe.

El resultado (sólo pondremos una parte del informe por cuestiones de espacio) será el siguiente:

LISTA DE DOCUMENTOS

Fecha del informe: **14-11-1998**

Fecha del sistema: **20-09-1997**

Página Nro: **01**

Documento	Fecha	Vencimiento	M o n t o
Av 000001	16-06-1997	16-06-1997	10.000,00
Av 000002	30-06-1997	30-06-1997	35.000,00
Av 000003	06-08-1997	06-08-1997	15.000,00
.....			
B1 000001	30-06-1997	30-06-1997	76.000,00
B1 000002	30-06-1997	30-06-1997	32.000,00
B1 000003	31-08-1997	31-08-1997	50.000,00
B1 000004	31-08-1997	31-08-1997	50.000,00
Cv 003922	29-08-1997	29-08-1997	75.014,00
Cv 005948	31-07-1997	31-07-1997	39.000,00

Cv	039222	30-06-1997	30-06-1997	101.750,00
Cv	104797	30-06-1997	30-06-1997	31.250,00
...				
Ch	103933	11-06-1997	11-06-1997	103.450,00
Ch	120596	09-09-1997	09-09-1997	242.500,00
...				

LISTA DE DOCUMENTOS

Fecha del informe: **14-11-1998**
Fecha del sistema: **20-09-1997**

Página Nro: **02**

Documento	Fecha	Vencimiento	M o n t o	
Ch	900857	30-06-1997	30-06-1997	342.000,00
...				
D1	MEO0997	30-09-1997	30-09-1997	53.750,83
D2	VEH0697	30-06-1997	30-06-1997	104.171,67
...				
Dc	0697-02	03-06-1997	03-06-1997	5.365.288,00
...				
Dv	306958	14-08-1997	14-08-1997	1.198.785,00
FC	000001	15-06-1997	15-07-1997	1.450.000,00
FC	000002	17-06-1997	18-08-1997	450.000,00
...				
Fm	293833	06-06-1997	06-07-1997	1.345.600,00
Fm	392288	08-07-1997	08-08-1997	2.560.320,00
...				
FP	5483	06-06-1997	06-07-1997	120.000,00

LISTA DE DOCUMENTOS

Fecha del informe: **14-11-1998**
Fecha del sistema: **20-09-1997**

Página Nro: **03**

Documento	Fecha	Vencimiento	M o n t o	
FP	7123	24-09-1997	24-10-1997	252.500,00
FP	97860	30-06-1997	30-06-1997	48.900,00
...				
Nc	103922	30-06-1997	30-06-1997	1.820.000,00
Nd	069666	31-08-1997	31-08-1997	4.000,00
Rc	000001	30-07-1997	30-07-1997	524.250,00
...				
Tl	123846	19-08-1997	19-08-1997	400.000,00
Tc	000001	04-08-1997	04-08-1997	5.600,00
Tc	000002	08-09-1997	08-09-1997	4.200,00
...				
Vc	000001	25-07-1997	25-07-1997	165.000,00
Vv	000001	12-08-1997	12-08-1997	45.600,00
Vv	000002	29-08-1997	29-08-1997	103.000,00
=====				
T O T A L			127.460.949,87	

1.4.11. Interacción con el usuario

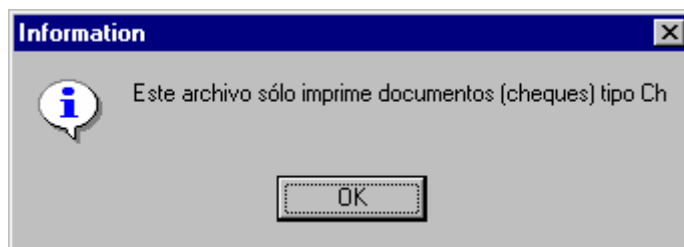
LOPI incluye dos instrucciones, **usuario** y **abortar**, que permiten dar un mensaje al usuario y abandonar un informe programáticamente; veamos:

```
.macro verificar_cheque
.si @tipo Ch <>
.usuario Este archivo sólo imprime documentos (cheques) tipo Ch
.abortar
.fin
.fin
^ documentos=abre
^ documentos=ventana
.tipo; documentos.codigo
&verificar_cheque
```

En este ejemplo pretendemos imprimir un documento de la tabla de documentos que será seleccionado por el usuario desde una ventana; ahora bien, el informe ha sido diseñado para un tipo específico de documentos; los documentos tipo 'Ch'. No podemos permitir que por error de selección del usuario el informe se aplique a otro tipo de documento: nota, factura, depósito, etc.

Para ello diseñamos el macro **verificar_cheque**; una vez que el usuario selecciona el documento a imprimir ejecutamos el macro, el cual compara la variable **tipo** que contiene el tipo de documento escogido con el deseado, es decir 'Ch'. Si el tipo es diferente de 'Ch' (Cheque inteno), entonces se da un mensaje al usuario con la instrucción **.usuario** y se abandona la ejecución del programa con la instrucción **.abortar**.

El resultado será:



Cuando el usuario haga clic en OK se cerrará la forma y se abandonará la ejecución del informe.

1.4.12. Salida (impresión) de los informes

LOPI permite al usuario seleccionar el canal de salida de los informes generados al ejecutar un programa; los canales posibles son:

- A través de la unidad de impresión de caracteres de **Opal** (impresoras matriz de puntos).
- A través del impresor de reportes de **Opal** (impresoras chorro de tinta o laser).
- Al editor RTF de **Opal**.
- A la forma de verificación de informes **LOPI** de **Opal**.
- Directamente al servidor de impresión de Windows.

Las instrucciones para indicar hacia donde se enviará el informe es conveniente colocarla en los programas al inicio de ellos, antes de las otras instrucciones que ejecutan el informe en sí. Veamos a continuación cuales son estas instrucciones:

```
.dos
.fin
```

La instrucción **.dos** permite dos cosas: primero, informar a **LOPI** que el informe debe ser enviado a la unidad de impresión de caracteres de **Opal** (impresoras matriz de puntos) y segundo, definir el encabezado que debe tener el informe a imprimir. Observe que la instrucción debe cerrarse con **.fin**.

Dentro de **.dos** y **.fin** debe indicarse el encabezado a utilizar; si el encabezado va a ser generado por el mismo programa **LOPI** entonces las instrucciones **.dos** y **.fin** no deben contener nada entre ellas, como se indica arriba; en estos casos es responsabilidad del programador generar los encabezados y controlar los saltos de página, como se hizo arriba en el ejemplo presentado para mostrar el uso de los macros. En ese ejemplo colocamos la instrucción **.dos** al inicio del programa:

```
.dos
.fin
.macro encabezado
~p; @p 1 +
LISTA DE DOCUMENTOS
```

.... resto del programa

En este ejemplo se envía el informe directo a la unidad de impresión de caracteres de **Opal**, pero al no indicar manejo del encabezado será responsabilidad del programador controlar los saltos de página e impresión de encabezados.

La instrucción **.dos** permite al programador dejar bajo la responsabilidad de **Opal** el manejo de encabezados y saltos de página; el ejemplo que vimos arriba del macro **encabezado** se simplifica enormemente usando la instrucción **.dos**; veamos:

```
.dos
hLISTA DE DOCUMENTOS
cDocumento Fecha Vencimiento Monto
.fin
^ documentos=abre
.fl; .f.
.total; 0; ###,###,##0.00
.cd; documentos.codigo; 3
.numero; documentos.numero; 8
.fecha; documentos.fecha; ddddd
.vence; documentos.vencimiento; ddddd
.monto; documentos.monto; ###,###,##0.00
.mientras documentos fin <>
@cd @numero @fecha @vence @monto
~fl; .t.
~total; @total @monto +
^ documentos=avanza
.fin
.si @fl
T O T A L ..... @total
.fin
```

Ahora no necesitamos ocuparnos del control de los encabezados y saltos de página; arriba, en la instrucción **.dos** **LOPI** informa a **Opal** que el informe será enviado a la unidad de impresión de caracteres y que debe usarse el encabezado de informe y de columnas indicado con las letras **h** y **c**.

Al ejecutar el informe **Opal** se encargará de colocar los encabezados incluyendo fecha del informe y fecha de proceso, controlar los saltos de páginas e imprimir el informe; el resultado será ahora:

```

LISTA DE DOCUMENTOS
=====
Empresa:      Empresa Ejemplo, C.A.
Fecha Opal:  20-09-1997
Fecha Dos:   14-11-1998
Página Nro:  1
=====
Documento    Fecha          Vencimiento          M o n t o
=====
Av 000001    16-06-1997    16-06-1997          10.000,00
Av 000002    30-06-1997    30-06-1997          35.000,00
Av 000003    06-08-1997    06-08-1997          15.000,00
Av 000004    25-08-1997    25-08-1997          10.000,00

```

Etc., etc., ...

Dentro de la instrucción **.dos** puede indicarse una línea de encabezado, hasta cinco líneas de sub-encabezado y cinco líneas de encabezado de columnas; por ejemplo:

```

.dos
hLISTA DE DOCUMENTOS
sLínea 1 de sub-encabezado
sLínea 2 de sub-encabezado
cDocumento    Fecha          Vencimiento          M o n t o
cLínea 2 de encabezado de columna
cLínea 3 de encabezado de columna
.fin
^ documentos=abre
.fl; .f.
.total; 0; ###,###,##0.00

```

Dará como resultado:

```

LISTA DE DOCUMENTOS
=====
Empresa:      Empresa Ejemplo, C.A.
Fecha Opal:  20-09-1997
Fecha Dos:   14-11-1998
Página Nro:  1
Línea 1 de sub-encabezado
Línea 2 de sub-encabezado
=====
Documento    Fecha          Vencimiento          M o n t o
Línea 2 de encabezado de columna
Línea 3 de encabezado de columna
=====
Av 000001    16-06-1997    16-06-1997          10.000,00
Av 000002    30-06-1997    30-06-1997          35.000,00
Av 000003    06-08-1997    06-08-1997          15.000,00
Av 000004    25-08-1997    25-08-1997          10.000,00
Av 000005    25-08-1997    25-08-1997          15.000,00

```

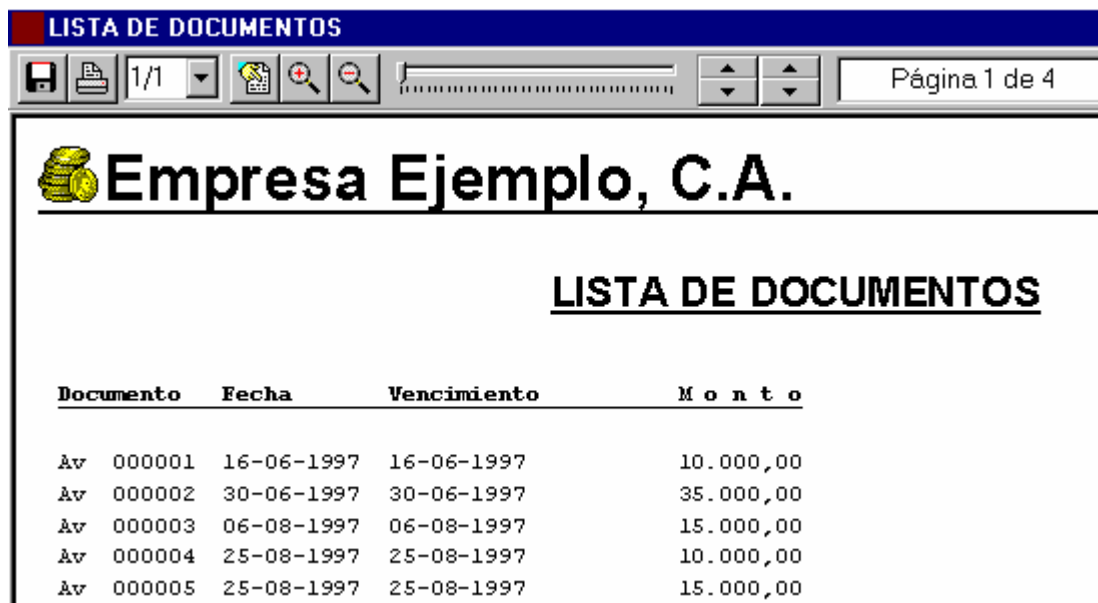
`.win`
`.fin`

La instrucción `.win` trabaja en la misma forma que la instrucción `.dos`, pero envía el informe al impresor de informes de **Opal** (impresoras de chorro de tinta y laser); el ejemplo anterior usando `.win`:

```
.win
hLISTA DE DOCUMENTOS
cDocumento Fecha Vencimiento M o n t o
.fin
^ documentos=abre
.fl; .f.
.total; 0; ###,###,##0.00
```

.... resto del programa

Se obtendrá el siguiente resultado:



The screenshot shows a window titled "LISTA DE DOCUMENTOS" with a toolbar containing icons for file operations and navigation. The main content area displays the company name "Empresa Ejemplo, C.A." and a table titled "LISTA DE DOCUMENTOS". The table has four columns: "Documento", "Fecha", "Vencimiento", and "M o n t o".

Documento	Fecha	Vencimiento	M o n t o
Av 000001	16-06-1997	16-06-1997	10.000,00
Av 000002	30-06-1997	30-06-1997	35.000,00
Av 000003	06-08-1997	06-08-1997	15.000,00
Av 000004	25-08-1997	25-08-1997	10.000,00
Av 000005	25-08-1997	25-08-1997	15.000,00

Nota: No debe utilizar la instrucción `.win` sin indicar encabezado como puede hacerlo con `.dos`. En estos casos no se obtiene ningún informe. El impresor de informes de **Opal** requiere de un encabezado previo para ser impreso y no permite que el usuario controle por sí mismo la impresión del encabezado. Por tanto esta instrucción:

`.win`
`.fin`

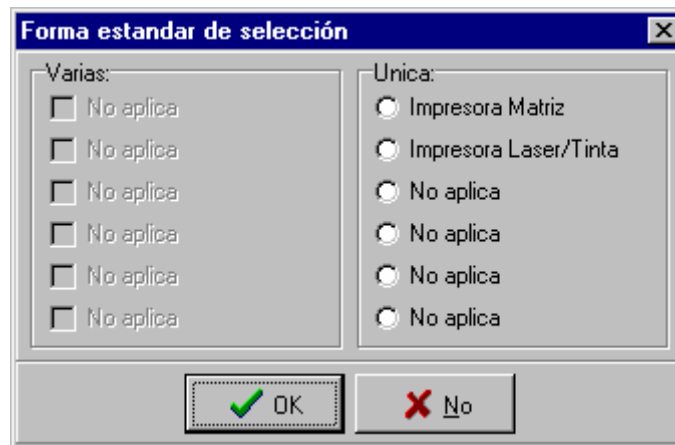
No generará ningún informe.

La instrucción **.doswin** consultará al usuario cual canal utilizar; si la unidad de impresión de caracteres o el impresor de reportes **Opal**; veamos:

```
.doswin
hLISTA DE DOCUMENTOS
cDocumento Fecha Vencimiento M o n t o
.fin
^ documentos=abre
.fl; .f.
.total; 0; ###,###,##0.00
```

.... resto del programa

Abrirá la siguiente forma para que el usuario seleccione el canal de impresión:



El usuario indicará el canal y hará clic en OK; el canal por omisión es **.dos**.

La instrucción **.editor** enviará el informe al editor RTF de **Opal**; por ejemplo:

```
.editor
.macro encabezado
~p; @p 1 +
LISTA DE DOCUMENTOS

Fecha del informe: @hoy
Fecha del sistema: @sistema
Página Nro: @p


---

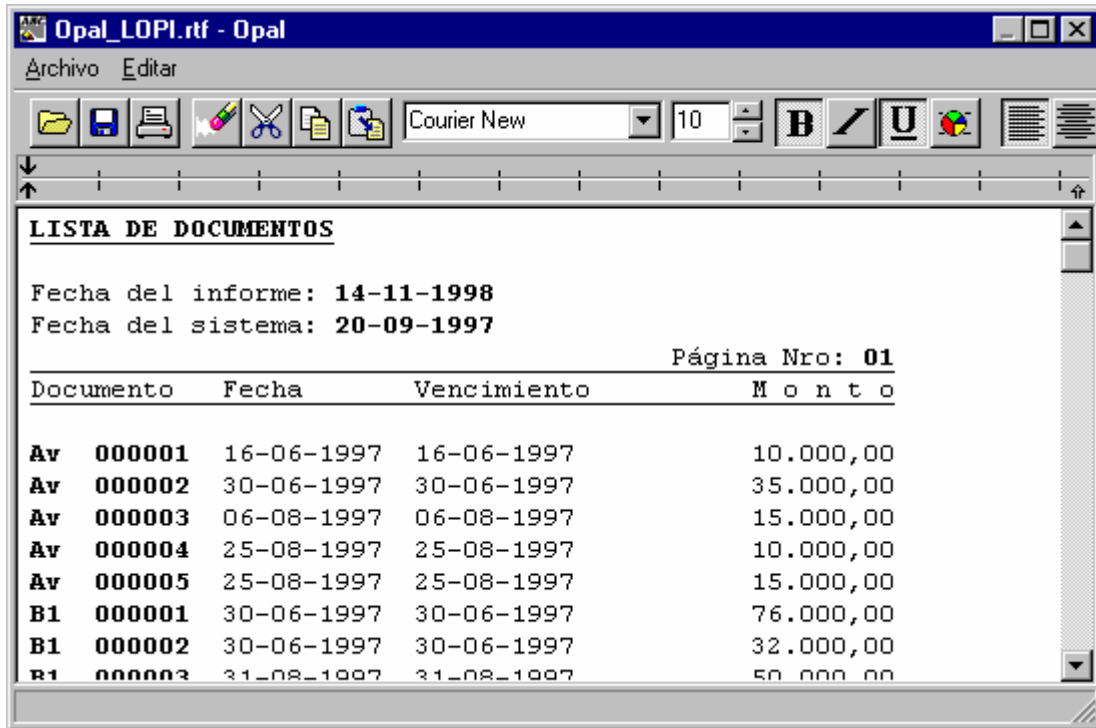

Documento Fecha Vencimiento M o n t o


---


.fin
^ documentos=abre
.hoy; hoy; ddddd
.sistema; opal; ddddd
```

.... resto del programa

Abrirá el editor RTF de **Opal** con el resultado de la ejecución del programa:



La instrucción **.impresor** enviará el informe directamente al servidor de impresión de Windows; desde allí el usuario puede escoger la impresora a utilizar y otras configuraciones:

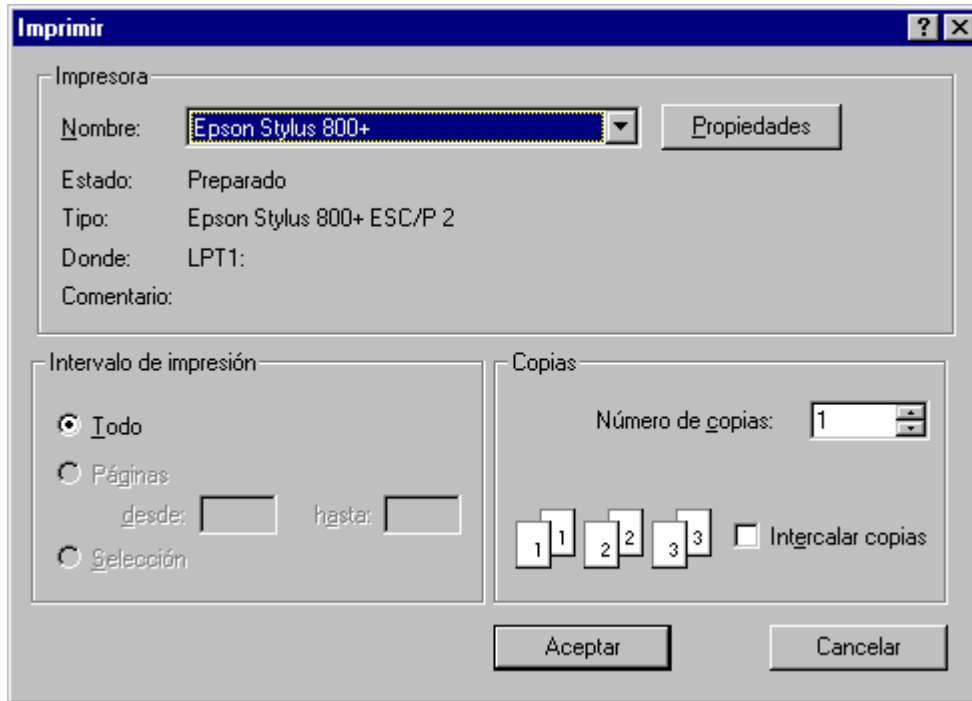
```
.impresor
.macro encabezado
~p; @p 1 +
LISTA DE DOCUMENTOS

Fecha del informe: @hoy
Fecha del sistema: @sistema
Página Nro: @p
Documento Fecha Vencimiento M o n t o

.fin
^ documentos=abre
.hoy; hoy; dddd
.sistema; opal; dddd
```

.... resto del programa

Abrirá la forma de diálogo de impresión estandar de Windows:



La instrucción **.test** enviará el informe a la forma de verificación de informes **LOPI** de **Opal**. Esta forma es muy conveniente para diseñar los programas por la variedad de opciones que permite:

```
.test
.macro encabezado
~p; @p 1 +
LISTA DE DOCUMENTOS
```

Fecha del informe: @**hoy**

Fecha del sistema: @**sistema**

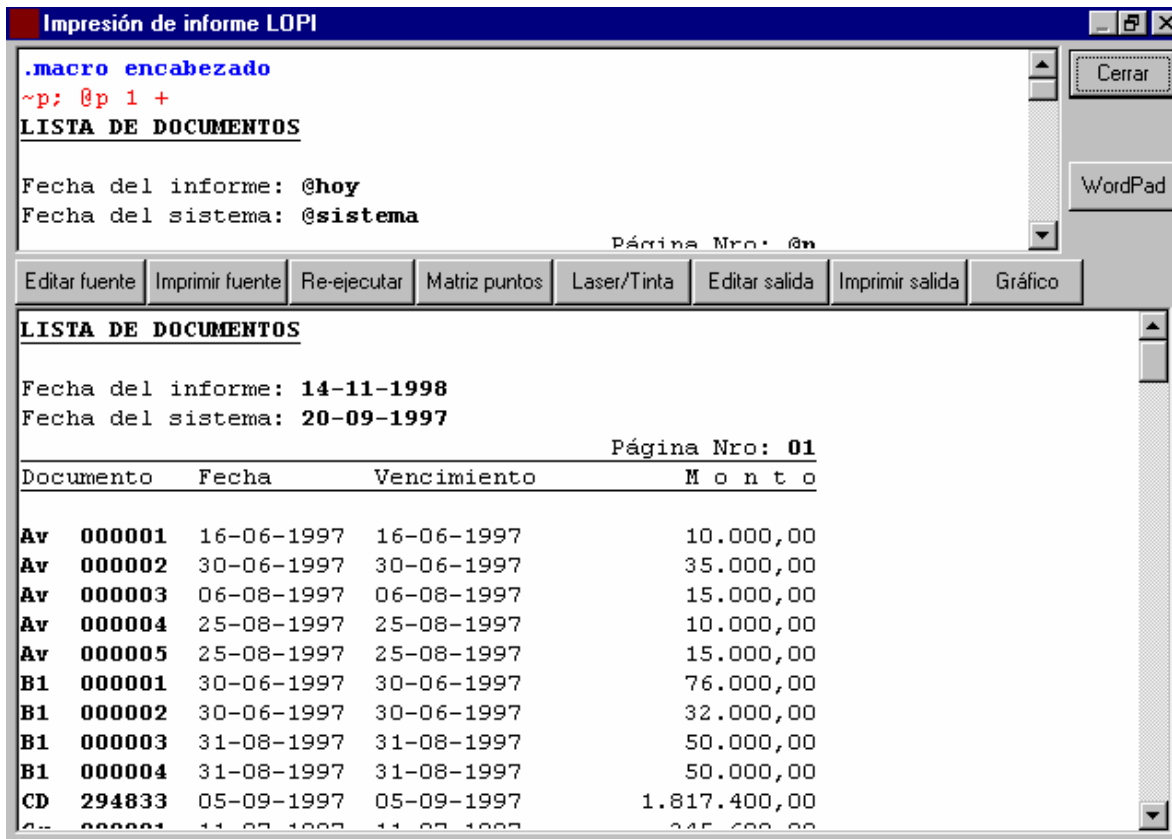
Página Nro: @**p**

Documento	Fecha	Vencimiento	M o n t o
-----------	-------	-------------	-----------

```
.fin
^ documentos=abre
.hoy; hoy; ddddd
.sistema; opal; ddddd
```

.... resto del programa

La instrucción **.test** es la salida por omisión de **LOPI**; si no se indica específicamente una salida con **.dos**, **.win**, **.doswin**, **.editor** o **.impresor**, **LOPI** asumirá por omisión enviar el informe a la forma de verificación de informe **LOPI** de **Opal**:



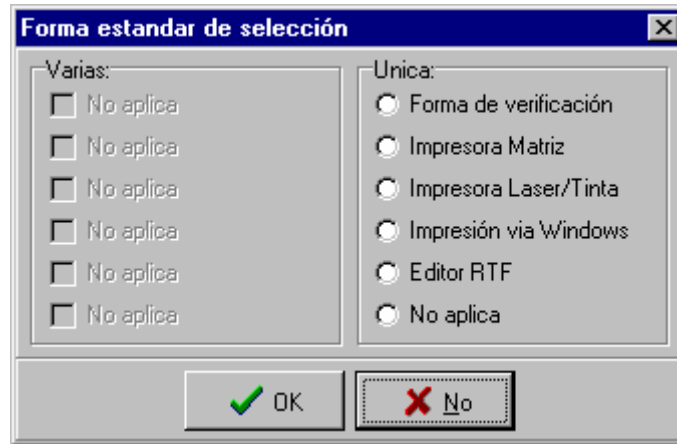
En la parte superior de la forma tenemos el código fuente del programa en LOPI; en la parte inferior la salida del programa. Los botones de la forma permiten:

- Editar fuente: permite abrir el editor RTF conteniendo el programa **LOPI** para ser editado.
- Imprimir fuente: permite imprimir por Windows el programa **LOPI**.
- Re-ejecutar: después de haber modificado el programa para corregir algo este botón permite re-ejecutarlo.
- Matriz de puntos: envía la salida (informe) a la unidad de impresión de caracteres de **Opal**.
- Laser/Tinta: envía la salida (informe) al impresor de documentos de **Opal**.
- Editar salida: envía la salida (informe) al editor RTF de **Opal**.
- Imprimir salida: imprime la salida (informe) a través del servidor de impresión de Windows.
- Gráfico: si la salida (informe) incluye un gráfico permite visualizarlo.
- WordPad: abre el editor **WordPad** de Windows con el programa fuente en **LOPI** para ser editado.
- Cerrar: cierra la forma de verificación de informes **LOPI**.

La instrucción **.?** abre una forma para que el usuario seleccione a cual canal desea enviar el informe:

```
.?
.macro encabezado
~p; @p 1 +
LISTA DE DOCUMENTOS
...
.fin
^ documentos=abre
.hoy; hoy; ddddd
.sistema; opal; ddddd
```

.... resto del programa



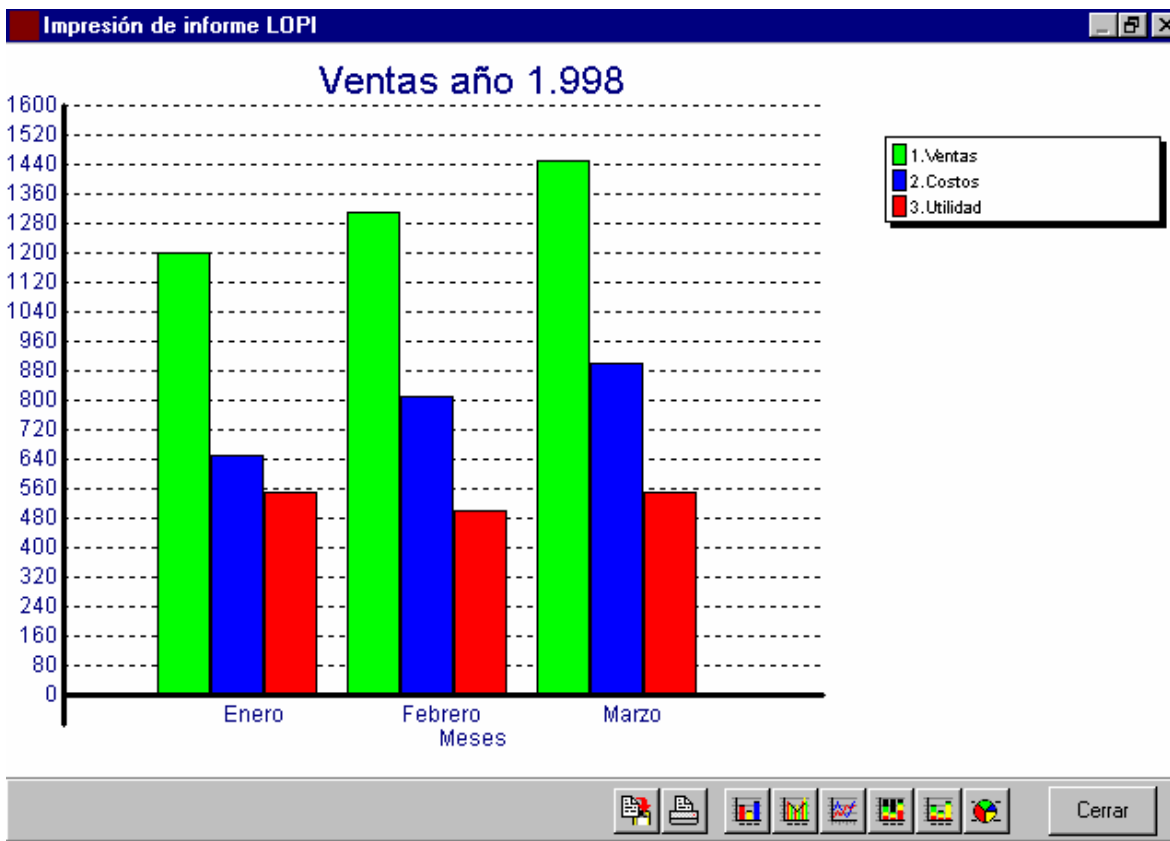
El usuario indicará el canal y hará clic en OK; el canal por omisión es la forma de verificación de informes **LOPI**.

1.4.13. Graficador

LOPI incluye un graficador sencillo y muy fácil de usar. Permite presentar cualquier número de gráficos; un gráfico es una representación de una serie de pares de datos x-y. Los datos pueden graficarse en líneas, barras o pasteles. Por ejemplo:

```
.ventas_enero; 1200
.ventas_febrero; 1310
.ventas_marzo; 1450
.costos_enero; 650
.costos_febrero; 812,50
.costos_marzo; 901,10
$ 1 tipografico
$ 'Ventas año 1.998' encabezado
$ 'Meses' pie
$ 'Ventas' 1 identifica
$ 'Costos' 2 identifica
$ 'Utilidad' 3 identifica
$ 'Enero' 1 etiqueta
$ 'Febrero' 2 etiqueta
$ 'Marzo' 3 etiqueta
$ @ventas_enero 1 1 asignar
$ @ventas_febrero 1 2 asignar
$ @ventas_marzo 1 3 asignar
$ @costos_enero 2 1 asignar
$ @costos_febrero 2 2 asignar
$ @costos_marzo 2 3 asignar
$ @ventas_enero @costos_enero - 3 1 asignar
$ @ventas_febrero @costos_febrero - 3 2 asignar
$ @ventas_marzo @costos_marzo - 3 3 asignar
```

Al ejecutar este programa se abrirá la forma de verificación de informes **LOPI**; desde esta forma haga clic en el botón Gráfico; obtendrá el siguiente gráfico:



Observe que en la imagen tenemos 3 gráficos: el verde representa las ventas, el azul los costos y el rojo la utilidad. Los valores asignados en el programa se representan en el gráfico. Veamos una explicación detallada de las instrucciones utilizadas:

Primero se definen las variables ventas y costos y se asignan los valores de ventas y costos que vamos a graficar. Estamos hablando de un gráfico sencillo con los resultados tabulados de 3 meses:

Mes	Ventas	Costos	Utilidad
Enero	1.200,00	650,00	550,00
Febrero	1.310,00	812,50	497,50
Marzo	1.450,00	901,10	548,90

La primera instrucción **tipografico** define el tipo de gráfico deseado (1=Barras; 2=Barras apiladas; 3=Líneas; 4=Barras con promedio; 5=Barras apiladas base 100; 6=Pastel). Se escoge 1 para un gráfico de barras. A continuación con la instrucción **encabezado** asignamos el nombre del gráfico; observe que las instrucciones del graficador comienzan con el símbolo \$ y por convención las coloreamos en **fucsia**.

Con la instrucción **pie** definimos el texto al pie del gráfico, para identificar los elementos del eje X; con la instrucción **identifica** identificamos los textos que definen los nombres de los gráficos a desplegar. El gráfico 1 será Ventas, el gráfico 2 será Costos y el gráfico 3 será Utilidad.

Con la instrucción **etiqueta** asignamos las etiquetas a los valores en el eje X; en este caso son los meses de Enero, Febrero y Marzo. Observe que Enero es el valor 1, Febrero el valor 2 y Marzo el valor 3.

Finalmente con la instrucción **asignar** asignamos los valores del eje Y para cada gráfico y valor del eje X. La instrucción **asignar** funciona así:

\$ <valor Y> <grafico> <valor X> asignar

De esta forma para los costos del mes de marzo:

<valor Y> = Costos de marzo = 901,10

<grafico> = Costos = gráfico 2

<valor X> = Marzo = X = 3

\$ @costos_marzo 2 3 asignar

Abajo, en la forma que presenta el gráfico, hay 9 botones que permiten:

1. Copiar el gráfico en el portapapeles; de esta forma puede pegarlo al texto de algún informe ejecutado en **Word, WordPad**, etc. Puede pegar también el resultado del informe **LOPI** en texto, con el resultado del informe en gráfico, aplicando la técnica Windows de copiar y pegar.
2. Imprimir el gráfico.
3. Presentar el gráfico en forma de barras (como se muestra arriba, en la figura).
4. Presentar el gráfico en forma de barras con promedio.
5. Presentar el gráfico en forma de líneas.
6. Presentar el gráfico en barras apiladas base 100.
7. Presentar el gráfico en barras apiladas.
8. Presentar el gráfico en pastel.
9. Cerrar la forma de presentación del gráfico.

1.4.14. Programas LOPI comentados

En el directorio \Opal\Informes se encuentran los archivos “fuente” de los siguientes programas en **LOPI** los cuales están profusamente comentados. El estudio cuidadoso de estos programas servirá de mucho al usuario para adiestrarse y mejorar su capacidad de programación en este macro-lenguaje. Los programas son de uso público y puede el usuario ejecutarlos, editarlos y/o adaptarlos a sus necesidades específicas.

- Comparativo de Ganancias y Pérdidas (Contabilidad)
- Indices financieros (Contabilidad)
- Variación en el Capital de Trabajo (Contabilidad)
- Aplicación de Fondos (Contabilidad)
- Impresión de un cheque interno (Administrativo)
- Impresión de un cheque de proveedor (Administrativo)
- Impresión de una factura de servicios (Administrativo)
- Libro de ventas del IVA (Administrativo)
- Uso del crédito : incluye gráfico (Administrativo)
- Partidas del circulante : incluye gráfico (Contabilidad)

Recuerde que para abrirlos debe utilizar un editor de archivos .rtf, tales como **Word, WordPad** o el editor **RTF** de **Opal**.

2.0. Actualizaciones, adiciones y mejoras

En esta sección se irán agregando actualizaciones, adiciones y mejoras que se vayan efectuando sobre **LOPI**. Un macro-lenguaje siempre tiene gran espacio de maniobra para nuevas facilidades que iremos compilando en este capítulo.

2.1. Archivos de texto *.txt (versión 2.01 de Opal)

LOPI permite trabajar, adicionalmente a los archivos *.rtf de Windows (rich text format), con archivos de texto *.txt. Estos archivos, de texto puro pues no manejan fuentes ni colores, **deben ser editados con editores de texto plano, tales como NotePad de Windows o Edit de Ms-Dos.**

Utilizar estos archivos presenta una desventaja importante y una ventaja significativa. La desventaja es que no pueden utilizarse fuentes y colores en la preparación de los informes. La ventaja es que son procesados mucho más rápido por **LOPI** que sus contrapartes en RTF.

Para la impresión de informes planos, borradores o de trabajo, o de impresión sobre formas pre-impresas (facturas, cheques, etc.), que se imprimen normalmente a través de la unidad de impresión de caracteres de **Opal**, es preferible la utilización de archivos de texto *.txt. Estos informes o formas no requieren de fuentes o colores especiales y por tanto deben diseñarse en texto.

Lo anterior es particularmente cierto para informes largos de varias o muchas páginas; en estos casos es preferible, aconsejable y hasta obligatorio utilizar archivos de texto, porque ejecutarlos bajo RTF es demasiado lento y no se justifica este retardo para procesar texto plano.

Por lo demás el diseño y operación es idéntico al de los archivos RTF en lo que a **LOPI** se refiere. Usted programa igual, ejecuta igual y actualiza, corrige o modifica de la misma forma; la única diferencia es que los archivos deben ser guardados en formato de texto plano *.txt al grabarlos la primera vez, y por tanto no pueden contener fuentes ni colores en su diseño.

Nota: Los archivos de código **LOPI** para las llamadas "Reglas del negocio" (vea manual de Seguridad de Opal para las versiones 2.2 y posteriores) deben ser del tipo texto *.txt.

2.2. Comandos SQL de ejecución (versión 2.2 de Opal)

La instrucción **xsq1** en **LOPI** permite ejecutar un comando SQL del tipo "update" o "delete" o inclusive comandos para crear o modificar tablas, tales como "create", "index", "drop", etc.

Se usa en forma similar a la instrucción **sql** (ver aparte 1.4.6):

? xsq1=micomando update documentos set status = 0 where codigo = 'FC'

Esta instrucción **LOPI** crea el comando SQL denominado "micomando" y lo ejecuta. Al ejecutarse la tabla de documentos será modificada fijando en cero la columna "status" para aquellos registros cuyo valor de la columna "codigo" sea FC.

El comando se nombra solamente para mantener la compatibilidad con la instrucción **sql** de **LOPI** para las consultas SQL; realmente "micomando" no puede utilizarse posteriormente ya que no retorna data alguna, solamente modifica la data indicada en el comando mismo.

Esta instrucción es muy importante ya que nos permite crear nuevas tablas que pueden ser utilizadas posteriormente; por ejemplo:

? xsql=test create table "test.db" (codigo char(20), monto decimal(20,2), primary key (codigo))

Crea la tabla "test.db" con una columna llamada "codigo" de 20 caracteres de ancho y una columna llamada monto, valor real de 20 caracteres de ancho y 2 decimales. La clave primaria es "codigo".

2.3. Modificar tablas (versión 2.2 de Opal)

LOPI permite ahora modificar las tablas subyacentes (**Usted debe usar esta opción con precaución y conociendo suficientemente la estructura de la base de datos de Opal**). Veamos con un ejemplo:

? xsql=test create table "test.db" (codigo char(20), monto decimal(20,2), primary key (codigo))

```
^ xtest=abre  
^ xtest=agrega  
^ xtest^codigo=100001  
^ xtest^monto=1200  
^ xtest=postea
```

La primera instrucción crea la tabla "test.db"; ya la vimos en la sección anterior. Las tablas que no son tablas regulares de **Opal** (no son Clientes, Documentos, Items, etc.) se pueden operar como las tablas regulares de **Opal** agregando la letra "x" a la izquierda del nombre de la tabla. De esta forma cuando decimos "xtest" nos estamos refiriendo a la tabla "test". "db" es la extensión por omisión para las tablas PARADOX (debe colocar siempre ".db" al crear una tabla).

La segunda instrucción abre la tabla; la tercera "agrega" un registro en blanco a la tabla. La cuarta y quinta instrucciones asignan los valores "100001" (caracter) y 1200 (número) a los campos o columnas "codigo" y "monto" de la tabla. La última instrucción "postea" los datos agregados a la tabla cerrando el nuevo registro. La tabla "test" ahora contendrá un registro.

Ademas de "agrega" y "postea", la instrucción "elimina" elimina un registro de una tabla; por ejemplo:

```
^ documentos=abre  
^ documentos.codigo=FC; numero=000001  
^ documentos=elimina
```

Estas instrucciones abren la tabla de "documentos", la posicionan en el registro "FC:000001" y luego eliminan el registro.

2.4. Abrir tablas regulares como auxiliares (versión 2.2 de Opal)

En los casos que Usted requiera abrir las tablas regulares de Opal como auxiliares, o abrir dos veces la misma tabla, puede anteponer "x" al nombre de la tabla para ejecutarlo; por ejemplo:

```
^ documentos=abre  
^ documentos.codigo=FC; numero=000001  
^ xdocumentos=abre  
^ xdocumentos.codigo=Cz; numero=002001
```

En este ejemplo la tabla de "documentos" se abre dos veces; las dos primeras instrucciones abren y posicionan la tabla "regular" de "documentos" de **Opal** en el registro "FC:000001". Las tercera y cuarta instrucciones abren una segunda vez la tabla de "documentos" en caracter de "auxiliar" posicionando la tabla en el registro "Cz:002001".

Hay casos es que es necesario abrir la misma tabla dos (y hasta mas) veces (**LOPI** permite abrirla dos veces). Un ejemplo típico es en los códigos **LOPI** requeridos para las llamadas "Reglas del negocio". Vea el manual de Seguridad de Opal para mas información.