# TABLA DE CONTENIDO

Generador de Reportes ..... 3
Ficheros ..... 3
Grupos de Reportes ..... 3
Generador de Reporte ..... 4
Incluir Reportes ..... 4
Crear Consulta SQL ..... 5
Construcción y Parámetros del Reporte ..... 19
Motor de Reportes ..... 33
Conceptos Básicos sobre Impresión en Windows ..... 33
Ténicas de Impresión Rápida ..... 34
Manejo de La Impresora como un Objeto ..... 34
Manipular Objeto Printer ..... 35
Definir una Letra que se Adapte a un Objeto Printer Específico ..... 35
Previsualización de Trabajos de Impresión ..... 36
Uso de la Clase Tprint ..... 37
Definición Escrita o Declarativa de Reportes ..... 38
Cálculo de Totales ..... 40
Cabeceras y Pies ..... 40
Títulos, Cabeceras y Pies Multilínea ..... 40
Uso de Expresiones ..... 41
Destino del Informe ..... 41
Uso de Tipos de Letra (Fuentes) ..... 42
Cláusulas ..... 44
Configuración de Columnas ..... 44
Columnas Multilínea ..... 45
Creación de Grupos ..... 45
Informes de Resumen ..... 46
Control del Flujo del Informe ..... 47
El Objeto Report ..... 48
Recorrido Hacia Atrás ..... 49
Impresión de Arrays ..... 50
Sombreados y Rejillas ..... 50
Cambio del Aspect ..... 51
Bitmaps ..... 52
Cuadros y Líneas ..... 53
Impresión De Campos Memo ..... 53
Colores ..... 55
Impresión de un Fichero TXT ..... 55
Inclusión de Formatos Elaborados en Crystal Report ..... 56
Condiciones para el Uso de Crystal Report con Datapro ..... 58

Ejecución del Reporte ..... 59
Fijar Parámetros ..... 59
Iniciar Parámetros ..... 60
Glosario de Términos ..... 61
Reporte ..... 61
Definición de Bases de Datos ..... 61
Tabla ..... 61
Registro ..... 62
Campo ..... 62
Relación entre Tablas ..... 62
Integridad Referencial ..... 62
Clave Primaria ..... 63
Tipos de Datos ..... 63
Operadores ..... 64
Diccionario de Datos ..... 65
Referencia de Colores ..... 66
Alta Densidad ..... 66

# GENERADOR DE REPORTES

El Generador de Reportes es una práctica herramienta de alto nivel del Sistema de Nómina de DataPro Windows, permite la creación de reportes escritos en código o léxico xbase de forma manual o auto-construido a través de un potente asistente visual que facilita la creación de reportes de manera instantánea con un mínimo de conocimientos y esfuerzo. Podrá a través del, crear desde simples listados generalizados, hasta complejos reportes especificando rangos y criterios de búsqueda que el sistema Nómina Windows de DataPro realizará en pocos instantes.

El Generador de Reportes también genera el código SQL necesario para obtener las consultas o querys de diversas tablas de la base de datos, requerido para lograr el modelo de consulta que será utilizado en la definición de cada columna que conforme el reporte.

# FICHEROS

Contiene el acceso a las opciones que permiten mantener los registros de las tablas que conforman el generador de reportes.

## GRUPOS DE REPORTES

Permite crear un catálogo para la clasificación de reportes, facilitando la administración, ubicación y ejecución al estar en conjunto. Es posible crear un conjunto de reportes con un uso específico, para lograrlo es necesario asociarlos por medio de un código de grupo, permitiendo posteriormente presentarlos en una lista o browser para su ejecución. Ejemplo: Todos los reportes creados para la gerencia de finanzas están asociados al grupo “Finanzas” y se requiere presentarlos en una lista para su ejecución, el sistema provee la función REPGRUPO(“NOMBRE”) que presenta todos los reportes de finanzas con el objetivo de solicitar la emisión de cualquiera de ellos.

Para definir un grupo de reporte, presione el botón e introduzca los datos en los siguientes campos:

**Código**: Acepta (8) ocho caracteres y se refiere al código que identifica al grupo.

**Descripción**: Acepta (40) caracteres para representar al grupo.

A través del botón “Aceptar” es grabado el registro. El botón “Cancelar”, cierra el formulario y regresa al editor de registros sin modificar o guardar la información.

Para asociar los grupos con los reportes, el formulario de identificación del reporte provee el campo "Código de Grupo".

## GENERADOR DE REPORTE

El acceso a la lista de reportes se realiza a través de un editor de registros "Browser" que muestra por cada reporte: Código, Descripción, nombre de la tabla principal y número de la aplicación asociada. Conjuntamente con la lista, se encuentra una barra de botones con los diversos accesos a las opciones que brinda el generador de reportes, dentro de ellas están:

### INCLUIR REPORTES

El diseño consta de cuatro secciones: identificación del reporte, generación de la consulta SQL construcción y parámetros del reporte y edición del programa fuente.

**Identificación Del Reporte**

Presenta una ventana de dialogo que solicita los datos que identifican al reporte y sus principales características que conforman la sección "Identificación del reporte".

**Código**: Acepta (10) diez caracteres que permiten identificar el reporte y deben corresponder a nombres válidos de archivos, debido que el reporte al igual que los demás programas DpXbase están compuestos por instrucciones que requieren ser compiladas para su ejecución, se compilará dicho reporte en carpeta la \dpnmwin\dpreport\estecodigo.dxb. Así mismo todos los parámetros también son grabados en un archivo de formato plano o texto bajo el criterio de ".INI" donde queda almacenado todas las características del reporte para la ejecución. Aunque toda los parámetros del informe se almacenan en la tabla "DPREPORT", el uso de parámetros almacenados en archivos de tipo texto tiene dos finalidades, optimizar la ejecución y poseer una copia del reporte.

**Descripción:** Acepta (40) cuarenta caracteres y se refiere al nombre o descripción del reporte que será utilizado para el diseño y ejecución.

**Nombre de la Tabla:** Solicita en nombre de la tabla principal que regirá el diseño del reporte, determinará los enlaces con otras tablas y formará parte de la cláusula "FROM" de la consulta SQL, necesario para la ejecución del reporte. Si el valor introducido no corresponde a una tabla registrada en el diccionario de datos, será presentado el catálogo de tablas que mostrará en cada registro: Nombre Nativo de la tabla, Descripción del Uso, número de identificación y número de aplicación asociado, presione Doble clic en el registro de la tabla deseado para exportar el nombre. Luego de ser validado el nombre de la tabla se visualiza la descripción y uso de la misma, así mismo suministrará el valor del módulo o aplicación donde se encuentra relacionada exportándolo hacia el campo "Aplicación". El valor de la tabla no puede ser cambiado cuando se está

modificando el reporte.

**Aplicación:** Presenta un selector que contiene la lista de todas las aplicaciones que conforman el sistema, su valor es obtenido desde el campo "Aplicación" según la tabla seleccionada y tiene como finalidad determinar la aplicación asociada con este reporte. Su valor no influye en el diseño o ejecución del reporte, sólo solo es utilizado con fines administrativos permitiendo clasificar o agrupar los reportes diseñados para ser determinada la aplicación, también permite en conjunto con el parámetro "Agregar en Menú", conocer la aplicación donde será creado automáticamente el registro del menú principal del sistema que permitirá su ejecución.

**Agregar en Menú** Al finalizar con el diseño, el sistema incorporará de manera automática un registro en la tabla del menú del sistema destinada a los reportes con la instrucción que solicita la ejecución del reporte mediante la función REPORTE(cCodigo)>.

Cierra el dialogo de identificación e inicia la segunda sección para el diseño de la consulta SQL.

Cierra el dialogo sin ningún efecto, si esta en modo "Inclusión", regresa hacia el editor de registros. Si está en modo "Modificación", regresa al editor de textos del programa fuente.

## CREAR CONSULTA SQL

Esta sección del reporte lo conforman un conjunto de formularios que tiene como finalidad asistir el diseño de la consulta SQL necesaria para definir las columnas que componen el reporte.

1. **Relación entre tablas:**

   Facilita la selección de diversas tablas relacionadas entre si. El usuario no requiere conocer las reglas de relación entre las tablas, ni la sintaxis SQL para relacionarla, sólo debe seleccionar la tabla que desea incluir en la relación. El proceso de selección de tablas se realiza a través de una ventana de dialogo compuesta de dos listas de tablas, la primera **"Descripción Tabla Origen"** contiene la lista de las tablas que solicitan la relación y la segunda lista **"Tabla Destino"** contiene las tablas que aceptan el enlace.

   Cada enlace genera la siguiente sentencia SQL:

   INNER JOIN TABLAA ON CAMPOA=CAMPOB

   También es posible crear relaciones indirectas utilizando las relaciones existentes.

Ejemplo: Relacionar: TablaA con TablaB y TablaB con TablaC, de acuerdo con esto es posible relacionar indirectamente TablaA con TablaC a través de TablaB.

La primera tabla que solicita la relación es la tabla principal del reporte "Trabajador" hacia una segunda "Grupos". Para crear una segunda relación, la tabla que solicita el enlace puede ser la Trabajador o la Grupo, seleccionado cualquiera de las dos, el sistema presentará la lista de las tablas enlazables según el diccionario de datos, la nueva selección "Regiones" en conjunto con "Trabajador" y "Grupos" pueden solicitar una nueva tabla enlazada. La ejecución cíclica de éste proceso puede lograr la relación de múltiples tablas.

A partir de la relación entre varias tablas, es posible construir la consulta SQL necesaria para la definición de las columnas del reporte. Cada campo estará identificado por el puntero "." La tabla que lo origina, bajo la siguiente sintaxis:

SELECT Trabajador.Codigo,Grup.gru_Codigo FROM Trabajador INNER JOIN NMGrupo ON Trabajador.Codigo= Grup.gru_Codigo

La ventana de dialogo contiene en la parte superior una barra de botones con las siguientes opciones:

Nuevo: Prepara el editor de enlaces insertando una nueva línea que presenta las tablas ya relacionadas listas para seleccionar una de ellas que solicitará el enlace. En la segunda columna se visualiza el nombre operativo de la tabla. En la tercera columna, se presenta una ventana de selección que contiene todas las posibles tablas enlazables, luego de seleccionar la tabla enlazada es mostrado en la cuarta columna su nombre operativo.

Elimina la relación que está focalizada. Cabe destacar que luego de cerrar la ventana de dialogo, el sistema ejecutara el proceso de depuración que también eliminará de los demás componentes del reporte los campos de la tabla eliminada de la relación.

Cierra la ventana y accede al siguiente formulario "SELECT"

Cerrar, admite las tablas relacionadas, cierra el formulario y despliega el editor del generador de reportes.

2. **Seleccionar Campos (SELECT ... FROM ) :**

El reporte debe estar compuesto por columnas y cada una debe estar asociada a un campo de la consulta o Query-SQL, éste se define indicando los campos requeridos de una o varias tablas relacionadas.

Este programa permite escoger todos los campos para la consulta SQL, es decir; los campos que conforman la sentencia “SELECT”; antes de armar esta sentencia, es necesario relacionar todas las tablas que poseen vínculos entre si mediante la opción “ENLACES”

El proceso para seleccionar los campos de la consulta se realiza a través de una ventana de dialogo compuesta por dos listas contentivas de campos, la primera lista contiene todos los campos de las tablas relacionadas y la segunda lista contendrá los campos que se vayan seleccionando de la primera.

En la parte inferior de la ventana de dialogo está disponible un selector de tablas que reemplaza la primera lista con todos los campos de la tabla seleccionada.

En la parte superior de la ventana se encuentra la barra de botones con las siguientes opciones:

**Campos**: Provee una lista ampliada de campos de la tabla seleccionada, cada uno muestra: Nombre, tipo, longitud, decimales y descripción. Esta lista solo es de uso informativo y en ningún momento afecta el proceso de selección de tablas.

Cierra la ventana y regresa al formulario de Relación entre tablas.

Cierra la ventana y despliega el siguiente formulario “Order By”

Aceptar, admite los campos seleccionados, cierra el formulario y regresa al editor del generador de reportes.

En la parte izquierda de la barra, también se encuentra un visualizador de la descripción del campo que se encuentra focalizado en cualquiera de las dos listas de campos.

Entre las dos listas de campos están disponibles seis botones con las siguientes funciones:

**Agregar**: Pasa un campo desde la primera lista hacia la segunda lista “Campos Seleccionados”.

**Todos**: traslada todos los campos desde la primera lista hacia la segunda lista de campos seleccionados.

**Quitar:** traslada el campo focalizado desde la segunda lista de elementos seleccionados hasta la primera para excluirlo.

**Quitar todos**: Vacía la lista de campos seleccionados y lo restaura en la primera lista de los

campos de la tabla.

Los campos de la lista se encuentran distribuidos verticalmente, mediante este control es posible hacer un cambio de lugar respecto a su disposición vertical. La posición del campo en la consulta no genera ningún efecto.

☐ Group By Permite convertir la consulta bajo la cláusula "GROUP BY" de léxico "SQL", es decir; la consulta se genera de manera resumida o sumarizada según todos los campos seleccionados**.**

*Importante: Los campos de tipo numéricos serán reemplazados por la función SUM() incluyendo el campo y el alias de la consulta asumirá el mismo nombre del campo.*

*Ejemplo:*

SELECT CAMPO FROM TABLA
Será sustituido por
SELECT SUM(CAMPO) ALIAS CAMPO FROM TABLA GROUP BY CAMPO

**3. Order By:**

Facilita la selección de los campos que conformarán la cláusula "ORDER BY" en la consulta, para lograrlo se presenta una ventana de dialogo con dos listas, la primera contiene los campos de las tablas y la segunda contiene los campos seleccionados necesarios para conformar el "Order By"

Este programa permite escoger todos los campos para la cláusula "Order By" del léxico SQL, es decir; los campos que determinaran la forma de ordenamiento para la consulta, previamente es necesario enlazar todas las tablas relacionadas según la opción "Enlaces entre tablas". No es necesario incluir los campos del Order By en el "SELECT" de la consulta.

El proceso para seleccionar los campos que ordenarán la consulta se realiza a través de una ventana de diálogo compuesta por dos listas contentivas de campos, la primera lista provee todos los campos de las tablas relacionadas y la segunda lista contiene los campos que conforman el "Order By"

En la parte inferior de la ventana de dialogo está disponible un selector de tablas que reemplaza la primera lista con todos los campos de la tabla seleccionada.

En la parte superior de la ventana se encuentra la barra de botones con las siguientes opciones:

Cierra la ventana y regresa al formulario "SELECT".

Cierra la ventana y accede al siguiente formulario “Grupos del Reporte”

Aceptar, admite los campos seleccionados, cierra el formulario y regresa al editor del generador de reportes.

En la parte superior derecha se encuentra un visualizador de la descripción del campo que se encuentra focalizado en cualquiera de las dos listas de campos.

Entre las dos listas de campos están disponibles seis controles con las siguientes funciones:

**Agregar**: Pasa un campo desde la primera lista hacia la segunda lista “Campos para el Order By”.

**Todos**: traslada todos los campos desde la primera lista hasta la segunda de campos que conforman el Order by.

**Quitar:** Excluye el campo focalizado desde la lista de campos que conforman el order by hacia la lista de campos de la tabla.

**Quitar todos**: Vacía la lista de campos que conforman el Order by y lo restaura en la primera lista de los campos de la tabla.

Facilita el cambio de posición vertical de los campos seleccionados. La posición del campo en la consulta no genera ningún efecto.

Permite agregar la sentencia “DESC” para la cláusula Order By, permitiendo solicitar la consulta ordenada de manera ascendente es decir los registros que tradicionalmente se encuentra la principio de la consulta ahora serán presentados al final de la consulta.

Ejemplo:

SELECT CAMPO FROM TABLA ORDER BY CAMPO
Será sustituido por
SELECT CAMPO FROM TABLA ORDER BY CAMPO DESC

☐ *Descending*

4. **Rango:**

Se define rango al intervalo entre dos valores “desde y hasta” donde se posibilita la capacidad de

lograr generar consultas según los valores iniciales y finales especificados.
Ejemplo: Listado de transacciones entre dos fechas.

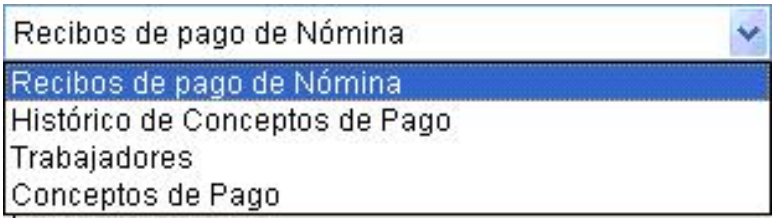
Las fechas que sean mayores o iguales a la **"Fecha Desde"** y menores o iguales a la **"Fecha Hasta"** se domina rango.

El generador de Reportes permite definir los posibles rangos que serán utilizados en el reporte, cada rango debe estar asociado a cualquier campo de la consulta o "Query", logrando construir sin intervención del usuario la sentencia WHERE aplicada en la consulta.
Ejemplo: Si es requerida la consulta de las transacciones entre dos fechas: 01/01/2004 hasta 31/12/2004, la sentencia WHERE con complementa la consulta SQL será:

WHERE TABLA.FECHA>='20040101' AND TABLA.FECHA<='20041231'

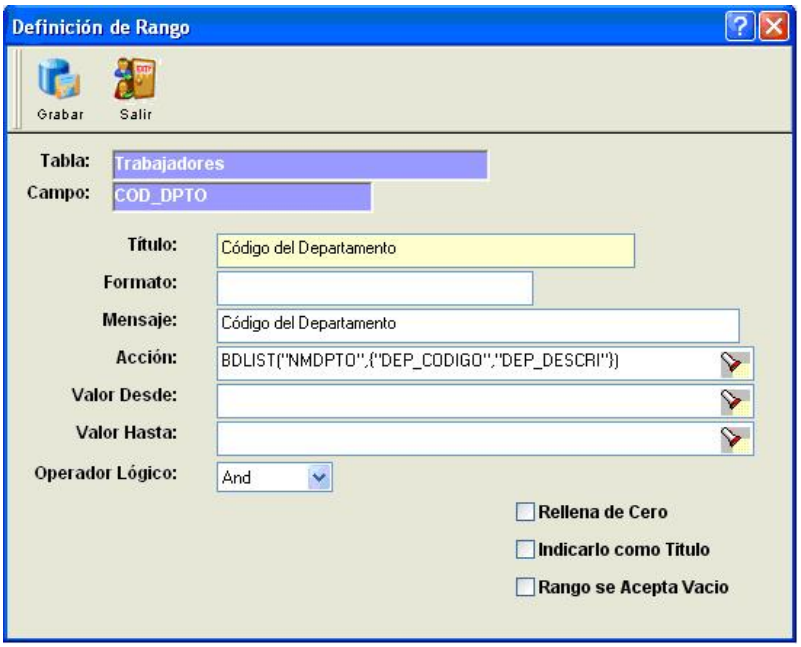
Para definir los campos que conforman el rango del Reporte es presentada una ventana de dialogo con dos columnas, la primera contiene los campos del Query previamente seleccionados a través de la opción "SELECT", la segunda columna contiene los campos seleccionados para conformar el Rango, entre ambas listas se encuentran cuatro botones ("Agregar","Quitar","Seleccionar Todos","Quitar Todos") que permite agregar o quitar campos del rango, también en la lista de campos ya seleccionados para el rango se encuentran dos botones que permite cambiar la posición vertical en cada rango.

En la parte inferior de la ventana se encuentra un selector de tablas que presenta las tablas previamente asociadas en la opción "Enlaces" y tiene como finalidad colocar en la lista de campos para seleccionar todos los campos de la tabla seleccionada.




Adicionalmente en la parte superior de la ventana, se encuentra una barra de botones con las siguientes opciones:

**Editar:** Permite personalizar el campo seleccionado como rango, presenta una nueva ventana de dialogo que modifica cada unos de los valores que conforman el rango y las opciones que facilitan el uso cuando el reporte esté en ejecución.



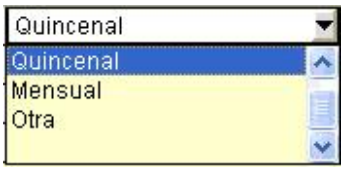

En la parte superior muestra el nombre de la tabla con la que se esta trabajando y también el nombre del campo utilizado para especificar el rango.

**Título:** Nombre que representa al rango, tiene efecto visual para el usuario y debe ser indicado con la mayor precisión posible ya que de este dependerá si se esta claro del valor que se introducirá en las demás casillas.

**Formato:** Se refiere a la forma de introducir los datos en cada casilla. Para los campos de tipo fecha debe ser: 99/99/9999 donde cada "9" significa valor de tipo numérico, el signo "/" se refiere al delimitador que separa día, mes y año. También es posible indicar cualquier formato de entrada de datos: Para rangos de tipo carácter que requiera valores solo en letras mayúsculas debe ser utilizado "@!". Para mayor comodidad el sistema sugiere un formato específico para cada tipo de datos soportado por el campo.

**Mensaje:** Corresponde al mensaje que será mostrado en la parte inferior o área de mensajes del formulario que ejecuta el reporte.

**Acción:** Está asociada a cualquier ejecución que pueda ser realizada por el usuario durante la edición de las casillas del rango y tiene como finalidad asistir la búsqueda de información que puede ser importada en cada casilla o también solicitar la asistencia de editores para visualizar los campos de una tabla e importar el contenido de un campo. Para mayor comodidad el sistema presente varias alternativas sugeridas que facilita crear un catálogo de edición bajos las siguientes funciones:



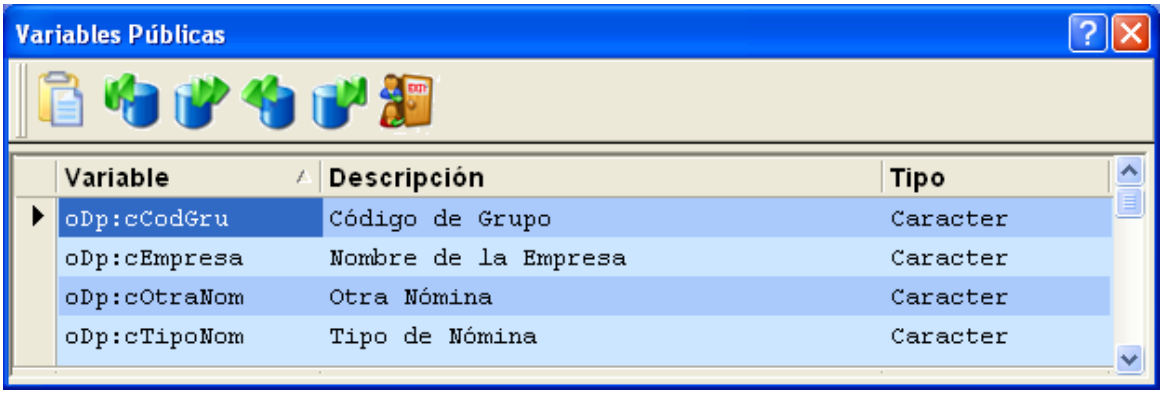

BDLIST("TABLA",{"CAMPO1","CAMPO2"}), Presenta varias alternativas empleando la función BDLIST() que permite presentar un catálogo de datos indicando el nombre de la tabla y la lista de campos que será visualizada en los items de la lista. Si el campo asociado al rango está relacionado con una o varias tablas también se presentará en la lista de opciones las diversas tablas y campos que podrán ser añadidos al catálogo de visualización.

Si el valor de la acción no está vacío, activará un botón de acción que ejecutará la instrucción definida.

**Valor Desde y Valor Hasta:** Ambos permiten definir el valor que será utilizado en las casillas desde y hasta del formulario de ejecución. El valor introducido debe ser un valor de sintaxis válida xBase y debe ser del mismo tipo de datos que soporta el campo relacionado con el rango. Para mayor comodidad, esta disponible el botón que presenta el catálogo de variables públicas que pueden suministrar su contenido hacia los valores del rango **"Desde"** o **"Hasta"**. El catálogo de variables sólo presentará las variables cuyo tipo sea compatible con el tipo de datos que soporta el campo, si este es de tipo "Carácter" sólo serán presentadas las variables y excluye a las demás variables de otro tipo. También es posible indicar valores directos, estos deben cumplir las siguientes condiciones para cada tipo de campo:

Carácter: Debe ser encerrado por comillas dobles, ejemplo: "00001"
Fecha: Deben ser acompañada por la función CTOD()
Ejemplo: CTOD("01/01/2004")
Numérico: Debe ser indicado de manera directa, ejemplo: 1000.00
Lógico: Debe ser indicado de manera directa: .T. o .F. (True o False – Verdadero o Falso)

Cabe destacar que el valor indicado para "Desde y Hasta" tiene preferencia sobre los valores "Fijados".



**Operador Lógico:** Presenta dos opciones “AND” y “OR” corresponden al operador lógico que será empleado para generar la cláusula WHERE generada por varios campos asociados con los rangos. Ejemplos con “AND” y “OR” empleando el intervalo representado por dos campos “CAMPO1” = Campo desde Y “CAMPO2” = Campo Hasta.

Caso “AND”, Deben cumplirse ambas condiciones:
(CAMPO1>=”001” AND CAMPO1<=”100”) AND (CAMPO2>=100 .AND. CAMPO2>=100)

Caso “OR” Solo debe cumplirse una de ellas:
(CAMPO1>=”001” AND CAMPO1<=”100”) OR (CAMPO2>=100 .AND. CAMPO2>=100)

☐ Rellena de Cero Realiza el autorelleno de ceros en las casillas cuando el formulario está en edición. Solo esta disponible para campos de tipo carácter. También su valor es obtenido del diccionario de datos.

☐ Indicarlo como Título Determina si los valores introducidos en la casillas “Desde y Hasta” serán mostrados en el título del reporte.
Ejemplo: Si seleccionamos esta casilla y campo1 y campo2 determinan un rango de fechas, en el título del reporte aparecerá:
Prenómina Semanal del campo1 hasta el campo2, se mostrará como título: Prenómina Semanal del 05/05/2004 hasta el 11/05/2004.

☐ Rango se Acepta Vacio Genera Cláusula WHERE para la consulta SQL aunque el valor de la casilla esté vació. La sintaxis queda de la siguiente forma: CAMPO=’’

En la parte superior de la ventana del dialogo se encuentra una barra de botones con dos opciones:

Acepta, guarda los valores del Rango, cierra la ventana y Regresa hacia el formulario de rango.

Cierra la ventana del rango sin guardar los cambios.

**Regresar:** Cierra el formulario de Rango y Regresa al formulario “COLUMNAS”

**Seguir:** Cierra el formulario de Rango y accede al formulario siguiente “Criterio”

**Aceptar**: Almacena, guarda los valores del rango, cierra el formulario y regresa al editor del reporte.

**Salir:** Cierra el formulario de Rango sin ningún tipo de cambio y regresa al editor del reporte.

## 5. Criterio

Define en la cláusula "WHERE" de ámbito relacional para la consulta SQL, empleando tres elementos: Campo, operador relacional y un parámetro introducido por el usuario logrando crear: CAMPO='VALOR'

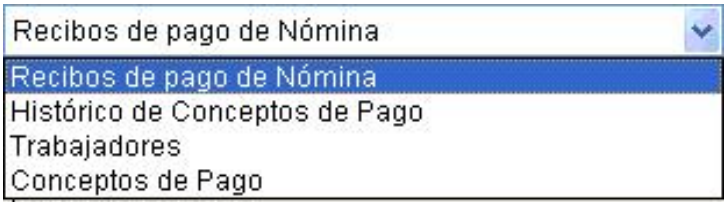
Se define criterio a la relación existente entre el valor de un campo comparado con un valor o parámetro indicado por el usuario, la comparación la realiza un operador relacional: = Igual, <>Diferente, > Mayor que ,< Menor que , >= Mayor o Igual,<= , menor o Igual ,LIKE Cadena Asociada e IN Incluido en.
Ejemplo: Para solicitar una consulta de trabajadores cuyo sueldo supera el monto 300.000, la consulta SQL debe ser:

SELECT CODIGO FROM NMTRABAJADOR WHERE SALARIO>300000

**Importante:** Recuerde que estos operadores son para el léxico SQL y no para xBase.

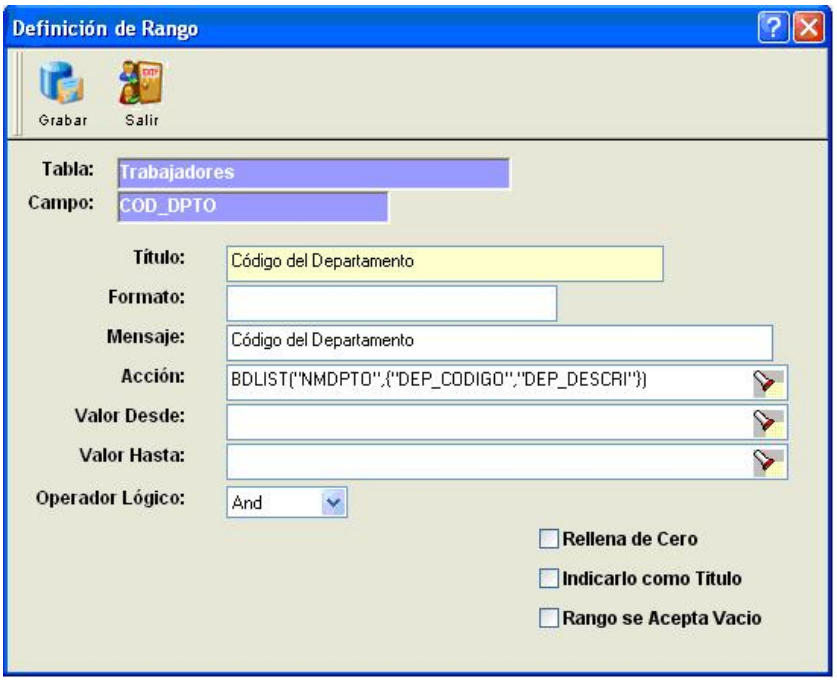
Para indicar los campos que conforman el criterio de la consulta SQL el sistema presenta una ventana de diálogo con dos columnas, la primera contiene los campos del Query previamente seleccionados a través de la opción "SELECT", la segunda columna contiene los campos seleccionados para conformar el Criterio, entre ambas listas se encuentran cuatro botones ("Agregar","Quitar","Seleccionar Todos","Quitar Todos") que permite agregar o quitar campos del rango, también en la lista de campos ya seleccionados para el criterio se encuentran dos botones que permiten cambiar la posición vertical de cada campo.

En la parte inferior de la ventana se encuentra un selector de tablas que muestra las tablas previamente relacionadas en la opción "Enlaces" y tiene como finalidad permitir incorporar campos de otras tablas para el criterio.



Adicionalmente en la parte superior de la ventana, se encuentra una barra de botones con las siguientes opciones:

**Editar:** Permite personalizar los parámetros para el campo seleccionado como criterio, presenta una nueva ventana de dialogo que modifica cada unos de los valores que conforman el criterio y las opciones que facilitan el uso cuando el reporte esté en ejecución.




En la parte superior muestra el nombre de la tabla con la que se esta trabajando y también el nombre del campo utilizado para especificar el rango.

**Título:** Nombre que representa al criterio, tiene uso visual para el usuario y debe ser indicado con la mayor precisión posible debido a que este mensaje le indica al usuario el valor que debe introducir en la casilla parámetro durante la ejecución del reporte.

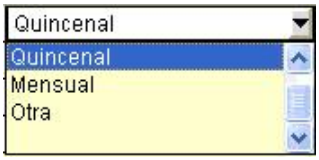
**Formato:** Se refiere a la forma de introducir los datos en cada casilla. Para los campos de tipo fecha debe ser: 99/99/9999 donde cada "9" significa valor de tipo numérico, el signo "/" se refiere al delimitador que separa día, mes y año. También es posible indicar cualquier formato de entrada de datos: Para rangos de tipo carácter que requiera valores solo en letras mayúsculas debe ser utilizado "@!". Para mayor comodidad el sistema sugiere un formato específico para cada tipo de datos soportado por el campo. El valor de esté parámetro es sugerido por el sistema según el tipo de campo seleccionado y los valores indicados en el diccionario de datos.

**Mensaje:** Corresponde al mensaje que será mostrado en la parte inferior o área de mensajes del

formulario que ejecuta el reporte.

**Acción:** Está asociada a cualquier ejecución que pueda ser realizada por el usuario durante la edición de las casillas del criterio y tiene como finalidad asistir la búsqueda de información que puede ser importada en cada casilla o también solicitar la asistencia de editores para visualizar los campos de una tabla e importar su contenido. Para mayor comodidad el sistema presente varias alternativas sugeridas que facilita crear un catálogo de edición bajos las siguientes funciones:

**Combobox:** Sólo esta disponible aquellos campos que posean opciones registradas en el diccionario de datos. Permitiendo utilizar una lista de selección al momento de editar la casilla del criterio, ejemplo: Para el campo: "TIPO_NOM" tipo de nómina asociado al trabajador, será presentada la siguiente lista cuando el reporte esté en ejecución:




BDLIST("TABLA",{"CAMPO1","CAMPO2"}), Presenta varias alternativas empleando la función BDLIST() que permite presentar un catálogo de datos indicando el nombre de la tabla y la lista de campos que mostrará la lista o Browse. Si el campo asociado al criterio está relacionado con una o varias tablas también se presentarán en la lista de opciones las diversas tablas y campos que podrán ser añadidos al catálogo de visualización.

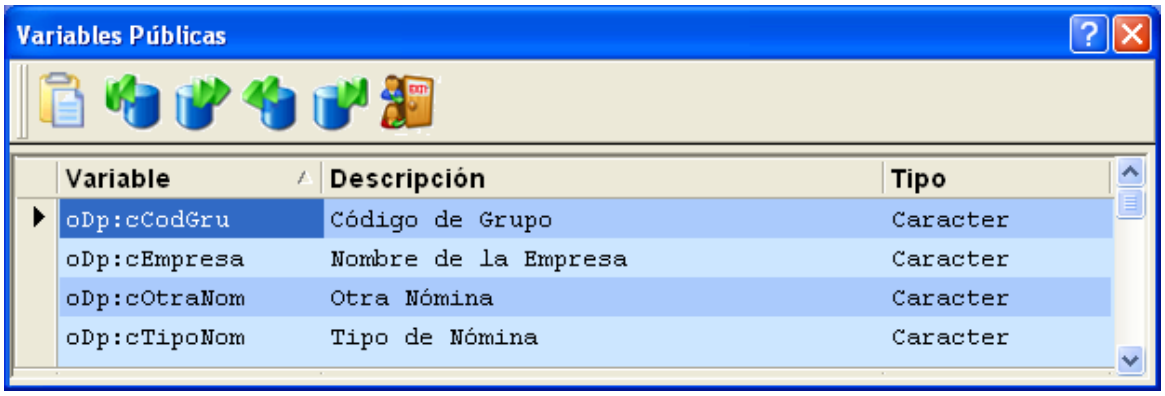
Si el valor de la acción no está vacío, será incluido en la casilla un botón que acciona la ejecución solicitada.

**Valor Inicial:** Acepta un valor compatible con el tipo de datos soportado con el campo seleccionado con el objetivo de establecer un valor inicial como parámetro de la casilla criterio, es decir; este valor ya estará incluido en el criterio cuando se solicita la ejecución del reporte. Si desea emplear valores del sistema, emplee el botón que muestra el catálogo de variables públicas que pueden suministrar el uso de su contenido. La lista sólo presentará las variables cuyo tipo sea compatible con el tipo de datos que soporta el campo, si este es de tipo "Carácter" sólo serán presentadas las variables y excluye a las demás variables de otro tipo. También es posible indicar valores directos, estos deben cumplir las siguientes condiciones para cada tipo de campo:

| | |
|---|---|
| Carácter | Debe ser encerrado por comillas dobles, ejemplo: "00001 |
| Fecha | Deben ser acompañada por la función CTOD("01/01/2004") |
| Numérico | Debe ser indicado de manera directa, ejemplo: 1000.00 |

| | |
|---|---|
| Lógico | Debe ser indicado de manera directa: .T. o .F. (True o False – Verdadero o Falso) |

| | |
|---|---|
| = | Igual |
| <> | Diferente |
| > | Mayor que |
| < | Menor que |
| >= | Mayor o Igual |
| <= | Menor o Igual |
| LIKE | Cadena asociada |




**Operador Lógico:** Presenta dos opciones "AND" y "OR" corresponden al operador lógico que será empleado para generar la cláusula WHERE generada por varios campos asociados para el criterio Ejemplos con "AND" y "OR" empleando la relación generada por un campo, operador relacional y parámetro: CAMPO='Parámetro'

| | | |
|---|---|---|
| AND | (CAMPO1='001')<br>**AND**<br>(CAMPO2=100) | Deben cumplirse ambas condiciones: |
| OR | (CAMPO1='001')<br>**OR**<br>(CAMPO2='100') | Debe cumplirse una de las condiciones. |

Operador Relacional: = Igual Presenta un selector con todos los operadores relaciones soportados por la consulta SQL.

En operador "IN" no es soportado en todos los léxicos SQL, o en algunas versiones de MySQL. Esto no implica imposibilidad de usar esta condición, el sistema lo convierte en una lista de relaciones utilizando el operador lógico "OR", ejemplo:

(CAMPO1='A' OR CAMPO2='B' OR CAMPO3='C')

El valor que debe ser introducido en la casilla debe estar encerrado entre llaves {} :

| | |
|---|---|
| Cadenas | {"A","B","C","D"} |
| Números | {1,2,3,4} |
| Fechas | {ctod("01/01/2004"),ctod("31/12/2004")} |

Para mayor comodidad, utilice la función BDSELECT() en el parámetro "Acción" que editará el un catálogo de datos selectivo que devolverá la lista de los campos seleccionados.

Hay que documentar la función BDSELECT()

☐ Rellena de Cero Realiza el autorelleno de ceros luego se ser editada la casilla en el formulario de ejecución. Sólo esta disponible para campos de tipo carácter y las reglas indicadas en el diccionario de datos.

☐ Indicarlo como Titulo Determina si el parámetro introducido en la casilla será mostrado en el título del reporte.

Ejemplo: Si 'Quincenal' es introducido como parámetro para el campo tipo nómina, será mostrado: Tipo de Nómina='Quincenal' en el encabezado del reporte.

Genera Cláusula WHERE para la consulta SQL aunque el valor de la casilla esté vació. La sintaxis queda de la siguiente forma: CAMPO=''

En la parte superior de la ventana del dialogo se encuentra una barra de botones con dos opciones:

Acepta, guarda los valores del Criterio, cierra la ventana y Regresa hacia la ventana de diálogos para criterios.

Cierra la ventana del criterio sin guardar los cambios.

## CONSTRUCCIÓN Y PARÁMETROS DEL REPORTE

Esta compuesto por un conjunto de formularios que trabajan en la construcción de las características del reporte basado en la sentencia SQL ya construida en la sección Anterior.

1. **Grupos:**
   Se refiere a las distintas secciones o grupos que conforman la información emitida en reporte o también podemos denominar "Rupturas de Control". Los reportes de listas simples emiten un listado sin "Cortes" de la información, ejemplo: "Lista de trabajadores con Código, Nombre y tipo de nómina.". Los reportes que requieren grupos cortes, seccionan las listas en varias sub-listas, cada sub-lista se rige a través de un identificador único que logra diferencias a cada sub-lista. Ejemplo: Listado de trabajadores agrupados o seccionados por Tipo de nómina cada sub-lista debe contener: Código y apellido.

   El objetivo de esta opción es permitir la definición de los grupos del reporte, esto se logra con tan solo indicar los campos que lo conforman, el sistema de manera automática se encargará de generar el código fuente del reporte con esta exigencia. También es necesario utilizar una consulta que esté ordenada por los campos que rigen los grupos del reporte, ejemplo: Si el reporte de trabajadores debe ser seccionado por tipo de nómina, la consulta debe poseer: "ORDER BY TIPO_NOM", en caso que la consulta SQL no posea esta exigencia el sistema la incorpora automáticamente en la Consulta. También suele suceder que la sentencia SQL ya está construida con otra exigencia "ORDER BY CODIGO", esto no será un inconveniente debido a que el sistema insertará loa listas de campos exigidos por el grupo dentro de la sentencia existente, Ejemplo:

   "ORDER BY CODIGO" será reemplazado por "ORDER BY TIPO_NOM, CODIGO"

   El orden de cada sub-lista será regido por el campo código, es decir, el reporte de trabajadores se presenta ordenado por código del trabajador. En caso de existir la cláusula "DESC" se mantendrá.

   La definición de grupos se realiza a través del editor de registros que permite incorporar varios de ellos. Cada línea está representada por el nombre del grupo y los campos que lo conforman. Inicialmente se presenta el símbolo "@" que significa el primer grupo, solo presione Doble clic e introduzca los siguientes datos:

   **Título:** Se refiere al título que encabezará el sub-listado. Si el campo para el grupo es "TIPO_NOM", el titulo puede ser "Tipo de Nómina"

**Expresión:** Esta compuesta por los campos de la consulta que rigen o determinan la sub-lista. También el valor será indicado en el encabezado de la sub-lista. Los campos del grupo deben indicar el objeto "oCursor" de consulta SQL. Ejemplo: <oCusor:TIPO_NOM>. Para mayor comodidad el sistema provee el botón , que presenta la lista de todos los campos que conforman la Consulta. Al seleccionar cualquier campo, su descripción será aplicada al campo "título". Para indicar varios campos para el grupo, debe utilizar la siguiente forma: oCursor:Campo1+oCursor:Campo2, recuerde que la sintaxis xBase debe concatenar expresiones con el Signo "+", y en SQL debe emplear la coma ",".

**Representa:** Se refiere al campo o expresión del cursor que representa visualmente el encabezado de cada sub-lista. Ejemplo: El reporte de histórico de pagos seccionado por trabajador, sólo muestra en el encabezado de cada sub-lista el código del trabajador, si se desea agregar en el encabezado el nombre y apellido, debe indicar en este campo: oCursor:APELLIDO+oCursor:NOMBRE

Todos los grupos definidos se ejecutan en forma concatenada de manera ascendente, es decir, la primera sub-lista rige a la segunda sub-lista. Ejemplo: Lista de trabajadores seccionado por departamento y luego dentro de este debe ser seccionados por tipo de nómina.

En la ventana de edición de grupos, se encuentra una barra de botones con opciones que facilitan su mantenimiento, las opciones son:

- Incluir o agregar un nuevo Grupo.
- Modificar un grupo existente.
- Eliminar un registro de grupo.
- Regresar, accede al formulario "Order By"
- Siguiente, avanza hacia el formulario "Columnas"

## 2. Columnas del Reporte:

Este programa permite escoger todos los campos de la consulta SQL que conformaran las columnas del reporte. Cada columna representa a un campo de cursor de datos o también a una expresión de tipo xBase.

El proceso para seleccionar las columna se realiza a través de una ventana de dialogo compuesta por dos listas contentivas de campos, la primera contiene todos los campos de las tablas previamente seleccionados en la opción "SELECT" y la segunda lista contiene las columnas que construirán el reporte.

Cada columna genera la sintaxis para la cláusula "COLUMN" del motor de informes, consulte en las siguientes páginas las diferentes alternativas que ofrece "COLUMN" así podrá personalizar con mayores alcances el diseño del reporte.

En la parte inferior de la ventana de dialogo está disponible un selector de tablas que reemplaza la primera lista con todos los campos de la tabla seleccionada.

En la parte superior de la ventana se encuentra la barra de botones con las siguientes opciones:

**Editar**: Presenta una ventada de dialogo que facilita editar los parámetros de la columna que está focalizada en la lista de columnas.

Cierra la ventana y regresa al formulario "Grupos"

Cierra la ventana y construye el código fuente del reporte según las columnas configuradas. Esta opción solo está activa cuando no ha sido construido el programa fuente.

Aceptar, admite las columnas seleccionadas sin afectar el código fuente del reporte, cierra el formulario y regresa al editor del generador de reportes.

En la parte izquierda de la barra, también se encuentra un visualizador de la descripción del campo que se encuentra focalizado en cualquiera de las dos listas de campos.

Entre las dos listas de campos están disponibles seis botones con las siguientes funciones:

**Agregar**: Pasa un campo desde la primera lista hacia la segunda lista "Columnas Seleccionadas".

**Todos**: traslada todos los campos desde la primera lista hacia la segunda lista de columnas seleccionadas.

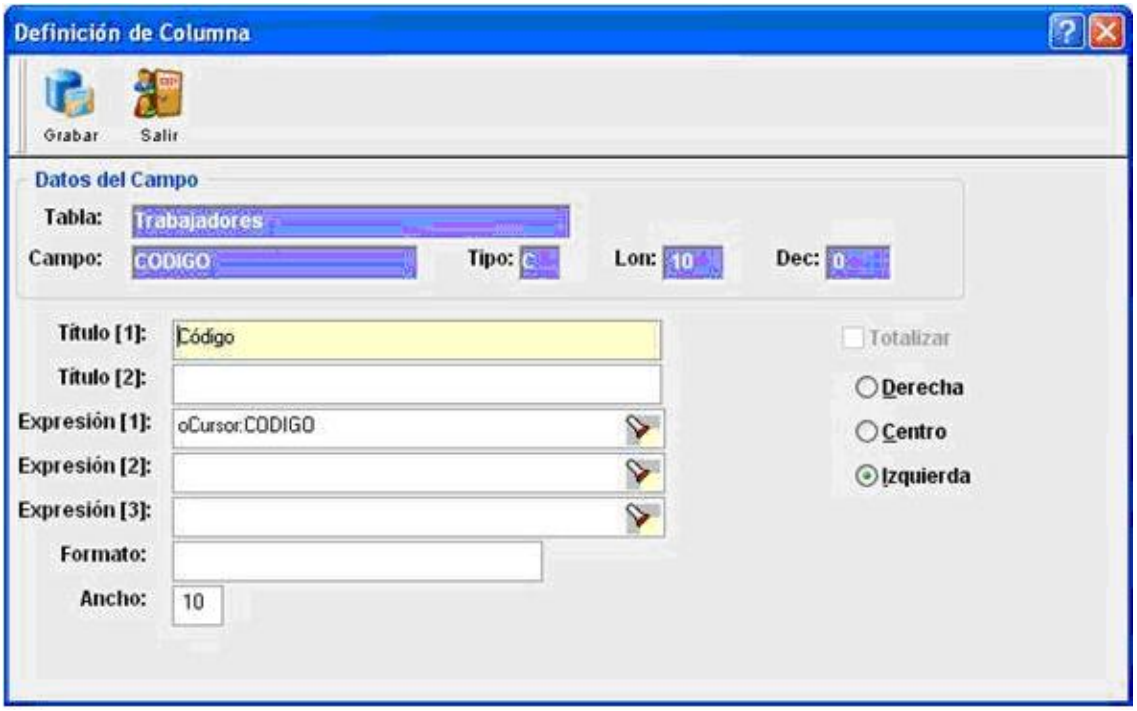
**Quitar:** retorna el campo focalizado presente en la lista de columnas o campos seleccionados hacia la primera lista.

**Quitar todos**: traslada todos los campos que conforman la lista de columnas para restaurarlos en la primera lista.

**Bajar Campo** y **Subir Campo** . Las columnas del reporte se encuentran distribuidas verticalmente, mediante estos controles es posible hacer un cambio de lugar respecto a su disposición vertical. La ubicación de los campos en la columna determinará el orden de los mismos en el reporte.

**Definición de Columna:**

Presione Doble clic sobre el campo que desea definir en columna, el sistema presenta una ventana de dialogo que muestra en la parte superior la información del campo que construye la columna y permite editar los siguientes parámetros:




**Título 1:** Contiene el título de la columna, esta casilla representa la primera línea del título. Ejemplo puede indicar "Código"

**Título 2:** Representa la segunda línea del título de la columna, ejemplo puede indicar "Trabajador", complementa la descripción del título de la columna.

**Expresión 1:** Indica la expresión que provee la información para la columna, básicamente es obtenida por un campo del cursor de datos SQL, Ejemplo oCursor: CODIGO, también puede aceptar expresiones de tipo xBase como: Left(oCursor: CODIGO,10). El uso del botón presenta la lista de campos de la consulta SQL para la selección de otro campo.

**Expresión 2:** Tiene la misma finalidad del parámetro "Expresión 1", permite crear una columna de tipo multilínea es decir; un columna que genera información en dos líneas, campo por ejemplo: Indicar en la primera línea la dirección 1 "oCursor:DIR1" y en la segunda línea la dirección 2 "oCursor:DIR2".

**Expresión 3:** Amplia la capacidad de crear tres líneas de datos en la misma columna.

**Formato:** Corresponde a la forma en que debe presentarse los datos de la primera Expresión, utiliza las mismas condiciones de la cláusula PICTURE del léxico xBase y básicamente se emplea para expresiones de tipo numérica y su valor debe estar encerrado entre comillas, ejemplo: “999,999,999.99”. Si el campo es de tipo numérico la sintaxis del formato es extraída del diccionario de datos y colocada en este parámetro.

**Ancho:** Representa la cantidad de Caracteres que serán visualizados en la columna, su valor es extraído del diccionario de datos, el usuario puede ajustarlo a sus requerimientos, Ejemplo: Si el código de trabajador tiene capacidad hasta 10 caracteres y solo utiliza 4 dígitos, puede indicar 4 en éste parámetro para que sólo se muestren cuatro dígitos del código.

**Alineación:**

Presenta tres opciones “Derecha, Centro e Izquierda” para indicar la alineación de los datos dentro de la columna. Si el campo es tipo numérico el valor por defecto es “Derecha”, así la presentación de la columna corresponde a valores numéricos.

En la parte superior de la ventana de dialogo están disponibles dos opciones:

Grabar, Acepta los valores introducidos y cierra la ventana de dialogo.

Cierra la ventana de dialogo sin ningún cambio.

Luego de personalizar cada una de las columnas puede proceder a la creación del código fuente del reporte.

## 3. Creación de Código Fuente:

Construye automáticamente todo el código fuente de tipo xBase según las columnas y grupos definidos por el usuario. Reduce significativamente el tiempo empleado en el diseño del reporte.

Si el código fuente ya existe, el sistema preguntará si desea reconstruirlo, en caso de ser afirmativo será generado nuevamente.

El código fuente construido genera las instrucciones para el motor de reportes empleando la clase TREPORT y el modelo sugerido se encuentra ubicado en forms\default.rep.

Estructura del programa fuente construido por DataPro.

```
#include "include\dpxBase.CH"
```

```
#include "include\DpxReport.ch"
```

Cabecera de definiciones, contiene la sintaxis de las sentencias para la librería DpXbase y la clase TREPORT.

```
PROCE MAIN(oGenRep)
   LOCAL cSql,oCursor,cMsg:=""

   PRIVATE oReport,nLineas:=0

   IF oGenRep=NIL
     RETURN .F.
   ENDIF
```

Declaración de la primera función ejecutada, declara las variables de ámbito local necesarias para los requerimientos del cursor de datos, sentencia SQL, Motor de Reporte, Generador de reporte y Flujo de ejecución. Uso de Variables

| | |
|---|---|
| cSql | Contiene la Sentencia SQL |
| cMsg | Acepta los mensajes que solicita la validación de la Sentencia SQL, en caso de error, cMsg contiene el mensaje. |
| nLineas | Contador de Líneas impresas, se acumula en la función ONCHANGE() |
| oGenRep | Es recibido como Parámetro se refiere al Motor de Diseño, empleado para obtener información sobre los parámetros del reporte |
| oReport | Representa al Motor del Reporte que se encarga de dirigir el flujo de ejecución. Para mayor información consulte mas adelante “Motor de Reporte” |

```
CursorWait()
```

Solicita el cursor de espera.

```
/*
// Aqui puede Personalizar la Consulta <QUERY>
   oGenRep:cSqlSelect    :="SELECT XCAMPO FROM NMTRABAJADOR"
   oGenRep:cSqlInnerJoin:=" INNER JOIN TABLAB ON CAMPOA=CAMPOB"
   oGenRep:cSqlOrderBy  :="ORDER BY XCAMPO"
*/
```

Ejemplo para personalizar componentes de la cláusula SQL en calidad de comentarios que se inician con /* y terminan en */. Puede eliminar los comentarios y emplear los datos del motor de diseño con la finalidad de obtener un cursor de datos diferente al solicitado por el diseñador. También el usuario puede incorporar otras funcionalidades que ofrecen los gestores de bases de datos y que no están incluidas en la sentencia SQL sugerida por el sistema.

| | |
|---|---|
| oGenRep:cSqlSelect | Representa la cláusula SELECT de la consulta. |
| oGenRep:cSqlInnerJoin | Representa la cláusula INNER JOIN, Incluso puede agregar nuevas solicitudes para enlazar otras tablas. |
| oGenRep:cSqlOrderBy | Representa la Cláusula ORDER BY de la consulta. Puede ser personalizada |

**oGenRep:cWhere :=oGenRep:BuildWhere()**

Construye de forma dinámica la cláusula WHERE de la Consulta según los valores y parámetros del componente rango y criterio.

**cSql :=oGenRep:BuildSql()**

Construye definitivamente la sentencia SQL sobre la variable cSql, concatenando los datos: oGenRep:cSqlSelect + oGenRep:cSqlInnerJoin + oGenRep:cWhere + oGenRep:cSqlOrderBy. Igualmente la variable cSQL puede contener una nueva sentencia escrita por el usuario, que en forma general la consulta se construye con las instrucciones indicadas en cSql.

```
IF !ChkSql(cSql,@cMsg)
  MensajeErr(cMsg,"Reporte <REPORTE>")
  Return .F.
ENDIF
```

Revisa la sintaxis de la sentencia SQL a través de la función ChkSql(<cSql>,<@cMsg>, la variable <cMsg> recibe el mensaje en caso que la consulta sea errónea o genere error.

```
IF !oGenRep:OutPut(.T.)
   RETURN .F.
ENDIF
```

Verifica el dispositivo de salida solicitado en la ejecución del Reporte y según este, puede cambiar el flujo de ejecución en caso de ser solicitado: Crystal Report, DBF u otro dispositivo diferente a: Preview, Impresora o Disco. En caso se seleccionar salida por Crystal , el sistema construye dos tablas DBF (Cuerpo y Encabezado) en la carpeta CRYSTAL\ con la finalidad de ser utilizada por el fichero de diseño “RPT” propietario de Crystal Report, también el usuario puede iniciar el diseño del reporte según la estructura de estas tablas. El programa **REPOUTPUT** escrito en DpXbase se encarga de administrar los dispositivos de salida y crear las condiciones para ejecutar diseños desde Crystal Report.

```
oCursor:=OpenTable(cSql,.T.)

IF oCursor:RecCount()=0
  MensajeErr("No fué posible Encontrar Información",;
```

```
            "Consulta Vacía Reporte <REPORTE>")
        oCursor:End()
        Return .F.
    ENDIF

    oCursor:GoTop()
```

Solicita el Cursor de datos empleando la función **OpenTable(<cSql>,<lLoad>)** según la cláusula contenida en la variable **<cSQL>.** El cursor de datos es dirigido por la clase TTABLE que emula su funcionalidad en forma analógica empleada por las tablas DBF. **IF oCursor:RecCount()=0,** verifica la cantidad de líneas contenidas en la consulta, en caso de ser "0" cero o la consulta esté vacía, mostrará un mensaje indicándole al usuario: "No fue posible encontrar la información", cierra el cursor **"Cursor:End()"** y finaliza **"Return .F."** la ejecución del reporte. Si el cursor de datos no está vació, ubica la primera línea "**oCursor:GoTop()"** y continua la ejecución del programa.

```
    REPORT oReport TITLE  "Trabajadores",;
        "Fecha: "+dtoc(Date())+" Hora: "+TIME();
        CAPTION "Trabajadores" ;
        FOOTER "Página: "+str(oReport:nPage,3)+;
            "Registros: "+alltrim(str(nLineas,5)) CENTER ;
        PREVIEW
```

Prepara y activa el motor de reportes indicando los textos y expresiones que conforman el título, encabezado y pié de página del reporte. El sistema internamente agrega los títulos conformados por parámetros del rango y Criterio.

```
  oGenRep:SetDevice(oReport)
```

Redirecciona el dispositivo de Salida: Preview, Printer o Disco. Igualmente puede incluir la cláusula para efectuar la sumatoria por columnas en el motor de reportes <oReport>

```
    COLUMN TITLE "Titulo";
        DATA oCursor:CAMPO;
        SIZE 10;
        LEFT
```

Construye la columna del reporte, indica el título "<TITLE>", el origen de los datos "<oCursor:Campo>", Ancho "<SIZE>" y alineación "<LEFT>". Sus parámetros, ubicación y cantidad son derivados de la configuración de cada columna seleccionada para el reporte.

```
        GROUP ON oCursor:CAMPO;
            HEADER GROUP01();
        FOOTER ENDGRP01()
```

La cláusula GROUP es agregada sólo si en la definición de reportes se exigen grupos. Esta cláusula prepara una sub-lista que imprimirá las definiciones de "COLUMN". Establece las

reglas de la sub-lista o Grupo con "**ON oCursor:CAMPO".** El encabezado se genera en la función GROUP01() ubicada en este mismo programa y el final o pié del listado se ejecuta en la función ENDGRP01(). Según la cantidad de grupos serán creadas las funciones que inicien y finalicen grupos GROUPnn() y ENDGRP01() donde (nn) es la posición de cada Grupo.

**END REPORT**

Finaliza las declaraciones del reporte.

**oReport:bSkip:={||oCursor:DbSkip()}**

Establece el avance del cursor de datos solicitado por el flujo de ejecución del reporte.

```
ACTIVATE REPORT oReport ;
    WHILE !oCursor:Eof();
    ON STARTGROUP oReport:NewLine();
    ON STARTPAGE  RepBitmap();
    ON CHANGE ONCHANGE()
```

Inicia la ejecución del reporte activando el motor <oReport>. "**WHILE oCursor:Eof()"** indica la condición de finalización."**ON STARTGROUP".** Evento que se ejecuta al iniciar el Grupo, "**oReport:NewLine()",** solicita avance de línea o dejar línea vacía. **"ON STARTPAGE",** evento que se ejecuta al iniciar cada página. **"RepBitmap()",** función que coloca el logotipo en la parte superior de la página. "**ON CHANGE ONCHANGE()"** evento que se ejecuta cada ves que se avanza un registro del cursor de datos, la función ONCHANGE() permite la creación de fórmulas.

**oGenRep:OutPut(.F.)**

Revisa los dispositivos de salida seleccionados para la ejecución del reporte, en caso de ser: "Ventana", es presentado en un editor de textos el resultado del reporte.

## *Funciones autoconstruidas para el flujo del reporte.*

Están compuestas por un conjunto de funciones creadas por DataPro y que tienen labores específicas según la naturaleza de cada reporte, se encuentran definidas en el mismo programa fuente a través de la cláusula "FUNCTION NAME()"

### Onchange ()

```
/*
// En Cada Registro se puede Aplicar Fórmulas
// Es llamado por Skip()
*/
```

```
FUNCTION ONCHANGE()

  nLineas:=nLineas+1 // Es Posible Aplicar Fórmulas

/*
// Si Desea Imprimir líneas Adicionales que no estén vacías

*/

 // PrintMemo(CAMPOMEMO,1,.F.,1) // Imprimir Campo Memo

RETURN .T.
```

Es ejecutada por el evento ONCHANGE() cada ves que avanza un registro del cursor de datos, tiene como finalidad permitir la manipulación de datos, tales como crear fórmulas, variables y otros valores que pueden ser utilizados por el reporte.
**"nLineas:=nLineas+1"** un contador de líneas, empleando la variable privada "nLineas" declarada al principio de la línea. **"PrintMemo(CAMPOMEMO,1,.F.,1)",** está como comentario "//" e inhabilitada, tiene como finalidad indicarle al usuario la sintaxis necesaria para imprimir el contenido de un campo memo en cualquier columna, el parámetro <CAMPOMEMO> debe ser reemplazado por oCursor:CAMPOMEMO. "**RETURN .T"** finaliza la ejecución de la función.

## Printmemo()

```
/*
// Imprime Campos Memos
*/
FUNCTION PrintMemo(cMemo,nCol,lData,nIni)
   LOCAL nFor,aLines

   IF Empty(cMemo)
     RETURN ""
   ENDIF

   // Inicio del Línea
   DEFAULT nIni:=1

   cMemo :=STRTRAN(cMemo,CHR(10),"")
   aLines:=_VECTOR(cMemo,CHR(13))

   IF lData // Requiera la Primera Línea de Datos
     Return aLines[1]
   ENDIF

//  oReport:BackLine(1) // Retroceder una Línea
//  oReport:Newline()   // Adelanta una Línea
   FOR nFor := nIni TO LEN(aLines)
      oReport:StartLine()
```

```
      oReport:Say(nCol,aLines[nFor])
      oReport:EndLine()
    NEXT
    oReport:Newline()

  RETURN ""
```

**PrintMemo(cMemo,nCol,lData,nDesde).** Se encarga de imprimir el contenido de campos memos. <cMemo> es el contenido del campo (memo o texto). <nCol> indica la posición de la columna donde será impreso. <lData> Indica si sólo necesita la información encontrada en la primera línea del memo y no realiza impresión del contenido. <nIni>, el número de línea que inicia la impresión. El uso de esta función permite crear columnas sólo compuestas de campos memos, o también imprimir los campos memos debajo de una línea, logrando ajustarse a múltiples necesidades.

```
    LOCAL nFor,aLines
```

Declara variables de ámbito local, <nFor> realiza el flujo de impresión de cada línea, <aLines> contiene en forma de arreglo cada línea del campo memo.

```
    IF Empty(cMemo)
      RETURN ""
    ENDIF
```

Valida que el campo memo no esté vació, en caso de estar vació, finaliza la función.

```
    // Inicio del Línea
    DEFAULT nIni:=1
```

Asumen <nIni>:=1 en caso no ser NIL o no recibir valor.

```
    cMemo :=STRTRAN(cMemo,CHR(10),"")
```

Extrae el CHR (10) del CRLF que separa cada línea.

```
    aLines:=_VECTOR(cMemo,CHR(13))
```

Genera un arreglo o lista según CHR (13) o separador de líneas, si el memo tiene 10 líneas el arreglo estará compuesto de 10 elementos.

```
    IF lData // Requiera la Primera Línea de Datos
      Return aLines[1]
    ENDIF
```

Si el Parámetro <lData> es verdadero devuelve el contenido de la primera línea.

```
     oReport:BackLine(1)
```

Retrocede una línea de impresión.

**oReport:Newline()**

Adelanta una línea de impresión.

**FOR nFor := nIni TO LEN(aLines)**

Ejecuta un proceso cíclico que fluye sobre cada elemento de la lista, desde el inicio hasta el final.

**oReport:StartLine()**

Prepara una nueva línea de impresión.

**oReport:Say(nCol,aLines[nFor])**

Imprime el contenido de la línea <aLines[nFor]> del campo memo sobre la columna indicada por <nCol>

**oReport:EndLine()**

Culmina el uso de la línea de impresión activada en <oReport:StartLine() y avanza hacia la siguiente línea.

**NEXT**

Finalización de las instrucciones cíclicas activadas por <FOR> envía el control de ejecución hacia la sentencia <FOR>

**oReport:Newline()**

Al finalizar la impresión de todas las líneas solicita avanzar una nueva línea de separación.

**RETURN ""**

Finaliza la función PrintMemo y devuelve un carácter vacío.

## Group01()

Es ejecutada cada vez que se inicia un grupo o sub-lista, suministra la información requerida por el encabezado. Igualmente pueden aplicarse las instrucciones de la función <PrintMemo()> para imprimir sobre la posición de cualquier columna.

/*

```
 Encabezado Grupo : Código
*/
FUNCTION GROUP01()
  LOCAL cExp:="",uValue:=""
  cExp  :="Código: "
  uValue:=oCursor:CODIGO
  uValue:=cValtoChar(uValue)+" "+cValToChar(oCursor:DESCRI)
RETURN cExp+uValue
```

A continuación detalle de cada instrucción utilizada en la función

**FUNCTION GROUP01()**

Declaración de la función.

**LOCAL cExp:="",uValue:=""**

Declara e inicia con data las variables <cExp> y <uValue> de ámbito local.

**cExp  :="Código: "**

Asigna la expresión "Código" a la variable <cExp>, corresponde al parámetro "título" de la definición del grupo.

**uValue:=oCursor:GTR_CODIGO**

Obtiene sobre la variable <uValue>, la expresión <oCursor:CODIGO> suministrada por el cursor de datos y definida en el parámetro **"Expresión"** en la definición del grupo.

**uValue:=cValtoChar(uValue)+" "+cValToChar(oCursor:DESCRI)**

Concatena en una sola cadena los valores de <uValue> más la expresión <oCursor:GTR_DESCRI> indicada en el parámetro **<Representa>** de la definición de la columna. La función **cValtoChar()** convierte la expresión en tipo carácter. Finalmente <uValue> estará conformada por el código más la descripción que rigen el grupo o la sub-lista.

**RETURN cExp+uValue**

Finaliza la función y devuelve en una sola lista concatenada el valor <cExp> + <uValue> que serán impresos en la cabecera del grupo o sub-lista de la siguiente manera: "Código: 0001 Nombre del Grupo"

## Endgrp01()

Es ejecutada cada vez que se finaliza un grupo o sub-lista, suministra la información requerida por el pie o cierre de grupo. Igualmente puede aplicarse las instrucciones de la función

<PrintMemo()> para imprimir cualquier expresión sobre la posición de cualquier columna.

```
/*
 Finalizar Grupo : Código
*/
FUNCTION ENDGRP01()
  LOCAL cExp:="",uValue:="",cLines:=""
  cExp  :="Total Código:  "
  uValue:=oReport:aGroups[1]:cValue
  uValue:=uValue
  uValue:=cValtoChar(uValue)
  cLines:=ltrim(str(oReport:aGroups[1]:nCounter))
  cLines:=" ("+cLines+")"
RETURN cExp+uValue+cLines
```

A continuación detalle de cada uno de los componentes.

Declaración de la función.

```
FUNCTION ENDGRP01()
```

```
LOCAL cExp:="",uValue:="",cLines:=""
```

Declara e inicia con data las variables <cExp>, <uValue>  y <cLines> de ámbito local.

```
cExp  :="Total Código:  "
```

Genera la expresión concatenada con “Total” más el parámetro título indicado en la definición del grupo.

```
uValue:= oReport:aGroups[1]:cValue
```

Contiene el valor que rige las reglas del grupo o sub-lista, se define en la declaración <GROUP ON **oCursor:CAMPO>.** La data del motor del reporte indicada en **<oReport:aGroups[1]:cValue>,** contiene el valor que rige el primer grupo y desde aquí se suministra para la variable <uValue>

```
uValue:=cValtoChar(uValue)
```

Convierte a <uValue> en expresión de tipo Carácter.

```
cLines:=ltrim(str(oReport:aGroups[1]:nCounter))
```

Obtiene de la data del motor de informes, el valor de las líneas o registros impresos para la sub-lista **<oReport:aGroups[1]:nCounter>.** La función STR() convierte la expresión en cadenas y la función lTrim() extrae los espacios vacíos, si la sub-lista empleo 10 registros del cursor de datos, el valor final de <cLines> es “10”.

**cLines:=" ("+cLines+")"**

Encierra entre paréntesis la cantidad de líneas contadas por el grupo, el contenido de <cLines> será “(10)”

**RETURN cExp+uValue+cLines**

Finaliza la función y concatena en una sola expresión. “Total Código: 0001 (10)”, es decir; el grupo se cierra indicando el código que lo rige y la cantidad de registros utilizados

### *Sugerencias*

El código creado por DataPro cumple con las definiciones indicadas por el usuario no obstante cabe destacar que es posible mejorar notablemente la presentación del informe, empleando fuentes de letras, colores, líneas, bordes, etc. En las siguientes páginas encontrará detalladamente cada una de las posibilidades que brinda el motor de reportes “TREPORT”.

# MOTOR DE REPORTES

Explica las posibilidades de construir poderosos reportes de forma declarativa y el uso de todos los componentes disponibles, entre ellos la clase de objeto TPRINT y TREPORT creadas exclusivamente para el uso de impresión.

## CONCEPTOS BÁSICOS SOBRE IMPRESIÓN EN WINDOWS

La principal diferencia entre el manejo de una impresora en MsDos y Windows es el hecho de que, en Windows, la impresora se gestiona en modo gráfico.

Esto supone que se van a manejar diferentes tamaños de letra, trazar algunos dibujos y, posiblemente, previsualizar todo en pantalla.

Y, lo más importante, que diferentes impresoras pueden tener diferentes resoluciones. Y todo esto tiene que controlarse.

FiveWin automatiza estos procesos ofreciéndole máxima potencia con la mínima complejidad. FiveWin clasifica los trabajos de impresiones en estos tipos diferentes.

- Técnicas de impresión rápida
- Manejo del objeto Printer
- El motor de informes

## TÉNICAS DE IMPRESIÓN RÁPIDA

Además del motor de informes de FiveWin, FiveWin le ofrece técnicas RAD (rapid aplication development) para crear rápidamente y sin esfuerzo prototipo de sus impresiones:

- Generación de un informe completo del área de trabajo actual: Llame a la función Report() para obtener, de forma automática, un informe completo y paginado del área de trabajo actual.

- Generación automática de un informe desde cualquier browse: Si ejecuta <oBrw>:Report() siendo <oBrw> una referencia a su browse, fiveWin generará un informe completo con el mismo contenido de su browse. Por tanto, la forma más fácil de implementar un informe de su browse, se basa en situar un botón bitmap en su cuadro de diálogo con la acción <oBrw>:Report().

- Impresión de una copia de una ventana o control: Si desea obtener una impresión de un cuadro de diálogo, basta con ejecutar: <oDlg>:HardCopy(<nZoom>) donde <oDlg> es un objeto cuadro de diálogo y <Zoom> es un factor de escala.

Se recomienda a usar el valor 3 para mantener las dimensiones naturales. Puede usar un bitmap en lugar de un cuadro de diálogo o cualquier ventana o control.

Combinando estas técnicas, puede crear un prototipo rápido de sus primeras impresiones. No las olvide.

Utilice estas técnicas en las primeras etapas de su aplicación.

## MANEJO DE LA IMPRESORA COMO UN OBJETO

FiveWin maneja la impresora como un objeto, por lo controlar el proceso de impresión es tan fácil como manipular, desde el punto de vista de la orientación a objetos, el objeto pinter (impresora)

¿Cómo se crea un objeto printer?

El motor de impresión se basa en la clase Tprinter para manejar la impresora como un objeto, así como mandatos xBase específicos para facilitar su manipulación:

PRINT oPrn NAME “Mi Impresión” PREVIEW

```
PAGE
. . .
oPrn:say(  nRow, nCol,  cText, . . .  )
. . .
ENDPAGE

ENDPRINT
```

El mandato PRINT crea un objeto TPRINT. Desde ese momento, puede manipularlo directamente, usando los datos y métodos ofrecidos por la clase Tprinter.

**MANIPULAR OBJETO PRINTER**

Habitualmente, las páginas se crean usando varios mandatos del tipo:

```
PAGE
...
ENDPAGE
```

Dentro de cada sección, se envían mensaje al objeto printer. El mensaje más habitual <oPrn>:Say():

```
oPrn:Say(nLinea,nColumna,cTexto,oFont)
```

Recuerde que las coordenadas de impresión dependen de la resolución de la impresora, que esté usando. Para controlar esto debe interrogar al objeto printer sobre su resolución específica.

oPrn:VertRes() es el número de pixel en Vertical.
oPrn:HorzRes() es el número de Pixel Horizontal.

Debe dividir esos valores entre el número de filas y columnas que desea manejar, e incrementar nRow,nCol de acuerdo a estos pasos.

**DEFINIR UNA LETRA QUE SE ADAPTE A UN OBJETO PRINTER ESPECÍFICO**

Para crear un objeto letra que adapte su tamaño a la resolución específica de una impresora, se utiliza  la cláusula contenedora OF: Ejemplo:

```
PROCEDURE MAIN ()
Local oPrn,oFont,nRow:=1,nCol:=1

PRINT oPrn
```

```
DEFINE FONT oFont NAME "ARIAL"  SIZE 0,-10 OF oPrn

 PAGE

   oPrn:Say(nRow,nCol,oDp:cEmpresa,oFont)

    ENDPAGE

 ENDPRINT
 oFont:End()
 RETURN NIL
```

Recuerde destruir la fuente <oFont:End()> una vez que finalice la impresión.

**PREVISUALIZACIÓN DE TRABAJOS DE IMPRESIÓN**

Puede generar una previsualización del trabajo de impresión indicando la cláusula PREVIEW al crear un objeto printer. Ejemplo:

```
PRINT oPrn NAME "Visualizar" PREVIEW

DEFINE FONT oFont NAME "ARIAL"  SIZE 0,-10 OF oPrn

 PAGE

   oPrn:Say(nRow,nCol,oDp:cEmpresa,oFont)

    ENDPAGE

 ENDPRINT
 oFont:End()
 RETURN NIL
```

Realiza la impresión en video, dando al usuario la posibilidad de previsualizar y de seleccionar o no la impresión del trabajo.

El motor de reportes utiliza archivos METAFILE de Windows para guardar cada página a imprimir. La dirección de estos ficheros se obtiene desde el valor "TEMP" del entorno de Windows.  SET TEMP C:\WINDOWS\TEMP

El motor de objetos almacenar los archivos temporales WMF (Windows Metafile) en esta carpeta y los borrará una vez finalice la previsualización.

**USO DE LA CLASE TPRINT**

Todas las instrucciones se escriben en forma de mandato.

```
PRINT <oPrint>;
  NAME <cName>;
  FROM USER;
  PREVIEW;
  TO <cModel>

  PAGE

    oPrint:Say(nRow,nCol,cText,oFont,nWidth,nClrText)

   ENDPAGE

  ENDPRINT
```

Significado de los Parámetros y comandos.

**PRINT** Genera el Objeto de Impresión.

**oPrint**: Referencia del objeto Impresora.

**cName:** Nombre del documento a Generar; es el nombre que aparece en la cola de impresión y como título de la ventana de previsualización.

**PAGE** Inicia una página nueva.

oPrint:Say(), Método que permite imprimir texto en las coordenadas específicas medida en unidades de resolución de la impresora. Es recomendable trabajar en centímetros o pulgadas utilizando métodos similares: oPrint:CmSay() o oPrint:InchSay() en los que las coordenadas estan en centímetros o pulgadas. Además, obtiene la misma presentación con diferentes impresoras.

<cTexto> Valor a imprimir, puede se cualquier tipo y el método lo convierte en una expresión de Cadena.

<oFont> Objeto oFont a usar sobre el valor <cText>

<nWidth> Ancho o espacio a utilizar medido en unidades específicas de la impresora. Si este

parámetro no es enviado, el método lo calcula automáticamente.

<nClrText> Color opcional a utilizar para <cText>, puede utilizar colores indicándolos en Definicion, Decimal o RGB(), para mayo información consulte referencia de colores.
Ejemplo: oPrint:CmSay(1,1,"Azul",oFont,nil,CLR_BLUE)

**ENDPAGE** Finaliza la actual página Impresa.

**ENDPRINT** Finaliza la Impresión.

**PREVIEW** Ejecuta el sistema de previsualización que ofrece al usuario una vista preliminar de la impresión antes de enviarla a la impresora.

## DEFINICIÓN ESCRITA O DECLARATIVA DE REPORTES

La clase Treport creada por "Ignacio Ortiz", es un motor de reportes de muy fácil definición y es altamente productiva.

Si observamos la clase TPRINT para construir reportes, es necesario tomar una calculadora y medir el tamaño de cada columna, distribuirlas eficientemente, crear un complejo de programa que administre totales, rupturas de control salto de páginas, lectura de datos, filtrado de registros, asignación de fuentes, entro otros. Para simplificar esta compleja labor la clase TREPORT proporciona un conjunto de métodos, valores y otras clases de objetos que se encargan de todo esto, solo necesita por parte del programador la declaración de actividades en forma de mandatos.

Todos los Reportes tiene la siguiente construcción:

```
    REPORT oReport ...
COLUMN ...
COLUMN ...
END REPORT

 GROUP ON ...
 GROUP ON ...

ACTIVATE REPORT oReport ...
```

Como puede ver, la construcción básica es muy simple. A continuación, se indica la sintaxis completa con todas sus opciones:

REPQRT [<oReport>];

```
[TITLE <bTitle,...>  [<LEFT │CENTER│CENTERED│RIGHT >]];
[HEADER <bHead, ...> [<LEFT │CENTER│CENTERED│RIGHT >]];
[FOOTER  <bFoot,...> [<LEFT │CENTER│CENTERED│RIGHT >]];
[FONT <oFont, ... > ];
[PEN <oPen, ... > ];
[< lSum:SUMMARY >] ;
[< file: FILE lFILENAME lDISK ><cRptFile> ];
[< resource: NAME │RESNAME │ RESOURCE ><cResName> ] ;
[< toprint: TO PRINTER >];
[< toScreen: PREVIEW >] ;
[TO FILE <toFile>];
[TO DEVICE <oDevice>];
[CAPTION <cName>]

GROUP [<oRptGrp>];
[ON <bGroup>];
[HEADER <bHead>];
[FOOTER <bFoot>];
[FONT <uFont>];
[< lEject:EJECT >]

COLUMN [<oRptCol>];
[TITLE <bTitle, ... > ];
[AT <nCol> ] ;
[DATA <bData, ...> ] ;
[SIZE <nSize> ] ;
[PICTURE <cPicture, ... > ]
[FONT <uFont> ] ;
[< total: TOTAL >[ FOR <bTotalExpr> ] ] ;
[< ColFmt: LEFT │ CENTER │CENTERED │ RIGHT >] ;
[< LShadow:SHADOW >] ;
[< lGrid:GRID >[ <nPen> ] ]

ENDREPORT

ACTIVATE REPORT <oReport> ;
[FOR <for> ] ;
[WHILE <while> ] ;
[ON INIT <uInit> ] ;
[ON END <uEnd> ] ;
[ON STARTPAGE <uStartPage> ] ;
[ON ENDPAGE <uEndPage> ] ;
[ON STARTGROUP <uStartGroup> ] ;
[ON ENDGROUP <uEndGroup> ] ;
[ON STARTLINE <uStartLine> ]
```

[ON ENDLINE <uEndLine> ] ;
[ON CHANGE <bChange> ]

Con el mandato COLUMN (vea la sintaxis) se indican todas las columnas que el informe tendrá, incluyendo los datos a mostrar y el titulo de cada columna. No puede ser más simple. Por ejemplo:
COLUMN TITLE "Nombre" DATA oCursor:Nombre

## CÁLCULO DE TOTALES

Si desea totalizar una columna, añada la cláusula TOTAL al mandato. Por ejemplo:
COLUMN TITLE "Sueldo" DATA oCursor:Salario TOTAL

## CABECERAS Y PIES

Puede añadir cabeceras y pies al informe declarándolos en el mandato REPORT. Por ejemplo:

```
REPORT oReport TITLE "Mi primer informe”;
    HEADER "Esta es la cabecera”;
    FOOTER "Este es el pie"
```

Puede incluso especificar su posición (a la izquierda, derecha o centrado) añadiendo LEFT, RIGHT o CENTER │ CENTERED. Por ejemplo:

```
REPORT oReport TITLE "Mi primer informe”;
    HEADER "Esta es la cabecera" RIGHT;
    FOOTER "Este es el pie" CENTER
```

Recuerde: Por defecto los títulos se centran, y las cabeceras y pies aparecen a la izquierda.

## TÍTULOS, CABECERAS Y PIES MULTILÍNEA

Si desea que un título ocupe más de una línea, separe cada parte del título mediante una coma. Por ejemplo:

```
REPORT oReport;
TITLE "Mi primer informe", "con Datapro"
```

Lo mismo se aplica a cabeceras y pies.

El motor de informes NUNCA añade una línea en blanco entre la cabecera y el título, o entre el título y los títulos de columna. Es su responsabilidad añadir líneas en blanco donde lo desee. Por ejemplo, si desea añadir una línea en blanco entre la cabecera y el título, simplemente indique una segunda línea en la cabecera con nada en ella (una cadena vacía):

```
REPORT oReport TITLE "Mi informe”;
    HEADER "Mi cabecera", " "
```

**USO DE EXPRESIONES**

Puede añadir prácticamente lo que quiera en sus informes ya que se permite el uso de cualquier expresión que devuelva una cadena de caracteres. Esta es la clave para mostrar cualquier cosa. Por ejemplo, si desea añadir la fecha al informe y el número de página actual, haga lo siguiente:

```
REPORT oReport TITLE "Mi informe"
“Fecha: "+DtoC (Date ());
FOOTER "Número página: "+;
Str(oReport:nPage,3)
```

Puede indicar expresiones en lugar de cadenas de caracteres en cualquier parte: títulos, cabeceas, - pies, títulos de columna, datos de columnas, etc...

**DESTINO DEL INFORME**

Por defecto, el informe se envía a la impresora, pero puede cambiar el destino al crear el informe:

```
REPORT oReport.... ;
   TO PRINTER=> (Por defecto)

REPORT oReport;
   PREVIEW (A la pantalla)

REPORT oReport....;
  TO FILE <cFile>  (Formato texto)
```

Cuando envía el informe a pantalla (PREVIEW), el motor de informes crea todas las páginas hasta que pulsa el botón Preview. A continuación, se muestra inmediatamente la primera página creada.

Al crear un informe puede especificar un nombre. Este nombre será el título de la ventana de previsualización, si envía el informe a la pantalla, o la descripción mostrada en el spooler, si lo envía a la impresora. (En versiones anteriores, se usaba la palabra NAME.)

**USO DE TIPOS DE LETRA (FUENTES)**

Puede usar cualquier tipo de letra en sus informes. Para ello:

Defínalas primero con el mandato DEFINE FONT oFont ....

```
DEFINE FONT oFontl NAME "ARIAL" SIZE 0,-10
DEFINE FONT oFont2 NAME "ARIAL" SIZE 0,-10 BOLD
```

Las especificaciones de tamaño (anchura, altura) emplean unidades de medida diferentes en función de si se imprime o se muestra en pantalla. En pantalla, las unidades son píxeles, pero al imprimir las unidades son "puntos" del tipo de letra. Por tanto, para texto impreso sólo se usa la segunda especificación de tamaño (altura) para dar el tamaño del punto, dejando la "anchura" a "O". El tamaño en puntos se expresa como un número negativo; por ejemplo "10 puntos" es "-10".

Recuerde que el tamaño en puntos para una letra proporcional es diferente que el "paso", término utilizado con frecuencia para tipos de letra no proporcionales como Courier. El tamaño en puntos refiere a la altura de la letra, mientras que el paso expresa el número de letras imprimidas por pulgada. El tipo de letra "pica" normal tiene un paso de 10, es decir 10 letras por pulgada. Pero, presada en tamaño, tiene 12 puntos. Por contra, el tipo de letra "elite", más pequeño, tiene un paso de 12 (12 letras por pulgada) y 9 puntos. (Resulta algo confuso, ya que cuando mayor es el paso, menor es la fuente, todo lo contrario que cuando se habla del tamaño en puntos.)

Además, recuerde que, en el motor de informes, FiveWin determina que está trabajando con tamaños en puntos en lugar de píxels si se usan números negativos (y debe usar puntos en lugar del paso para tipos de letra no proporcionales como Courier). En este caso, el signo "-" no significa (como las reglas matemáticas sugieren) que el tamaño del punto es menor cuanto mayor es el valor entero, sino todo lo contrario. Cuanto mayor es el entero, mayor es el tamaño.

Cuando cree el informe, especifique los tipos de letra que va a utilizar, separándolos con comas.
Por ejemplo:

```
REPORT oReport TITLE.... ;
      FONT oFontl, oFont2, oFont3
```

El primer tipo de letra de la lista (oFont1) será el estándar. Se utilizará siempre que no especifique o tipo de letra para una columna.

Si desea utilizar el segundo tipo de letra en otra columna, haga lo siguiente:

```
COLUMN “CODIGO”;
      DATA oCursor:CODIGO;
      FONT 2
```

Como puede ver, se indica el número ordinal que ocupa el tipo de letra en la lista definida al crear informe.

```
COLUMN “Sueldo”;
     DATA oCursor:SALARIO;
     FONT IIF (oCursor:SALARIO>l00000,2,1)
```

Recuerde que debe liberar los tipos de letra al finalizar el informe.

Aquí tiene la sintaxis completa:

Definir un tipo de letra:

```
     DEFINE FONT <oFont> ;
[ NAME <cNarne> ];
[ SIZE <nWidth>, <nHeight> ] ;
[ FROM USER ];
[ BOLD ] ;
[ ITALIC ] ;
[ UNDERLINE ] ;
[ WEIGHT <nWeight> ] ;
[ OF <oDevice> ] ;
[ NESCAPEMENT <nEscapernent> ] ;

     ACTIVATE FONT <oFont>

     DEACTIVATE FONT <oFont>

     RELEASE FONT <oFont>

     SET FONT [ OF <oWnd>] [TO <oFont> ]
```

**CLÁUSULAS**

| | |
|---|---|
| FROM USER | Muestra el cuadro de diálogo de Tipos de letra |
| BOLD | Crea un tipo de letra con el atributo negrita. |
| ITALIC | Crea un tipo de letra con el atributo cursiva. |
| UNDERLINE | Crea un tipo de letra con el atributo subrayado. |

**IMPRESIÓN VERTICAL**

Puede ser definido a través de las fuentes de letra, ejemplo:

DEFINE FONT oFont NAME "Arial" SIZE 0,-10 **NESCAPEMENT -900**

Si el valor nEscapement es positivo, la fuente será girada en sentido inverso al movimiento de las manecillas de un reloj, si el valor es negativo la fuente será girada en sentido a las manecillas del reloj. Con el valor 900 queda vertical. Luego se Asigna la fuente al reporte.

**CONFIGURACIÓN DE COLUMNAS**

Al crear las columnas, puede especificar su configuración:

- PICTURE: Idéntico a los GETS. Formato de la Expresión
- SIZE: Longitud de la columna (número de caracteres)
- AT: Imprime en una posición específica (mejor no usarlo)
- FONT: Tipo de letra a utilizar (número en la lista de fuentes)
- TOTAL: Si la columna debe tener un total.

Si especifica que la columna tiene un total, DATA debe ser numérico. Si no lo es, no se preocupe; el motor de informes no se detendrá. La cláusula TOTAL puede incluir una condición FOR. Por ejemplo:

```
COLUMN;
  TITLE “Salario”;
  DATA oCursor:SALARIO
  TOTAL FOR oCursor:COD_DPTO=”Ventas”
```

- LEFT│CENTER│CENTERED│RIGHT: También puede especificar la forma en que el texto se imprime (a la izquierda, derecha o centrado). Por defecto, todo se alinea a la izquierda

excepto los datos numéricos, que se alinean a la derecha.

## COLUMNAS MULTILÍNEA

Esto es algo que le encantará. Suponga que necesita añadir otra columna, pero la anchura del papel no es suficiente.

Podría usar un tipo de letra menor, pero no es una buena solución. La mejor forma es usar varias líneas en una misma columna. Esto significa que un registro de la base de datos puede usar más de una línea en el informe, y que es posible hacer cosas como esta:

| Trabajador | Salario |
|---|---|
| oCursor:APELLIDO<br>oCursor:NOMBRE | oCursor:Salario |

Es tan simple como:

```
COLUMN TITLE “Trabajador”;
   DATA oCursor:APELLIDO,oCursor:NOMBRE

COLUMN TITLE “Salario”;
   DATA oCursor:SALARIO
```

Basta con separar mediante una coma los datos a imprimir.

Puede hacer lo mismo con el título de la columna:

```
   COLUMN;
TITLE "Nombre", "Apellido"
   DATA oCursor:NOMBRE, oCursor:APELLIDO
```

Si usa la cláusula TOTAL en esa columna, se sumarán los valores numéricos. Aunque todos los datos usarán la misma PICTURE, este comportamiento se puede cambiar, como luego se verá.

## CREACIÓN DE GRUPOS

Creando en un grupo se pueden obtener subtotales sobre un dato en particular. Con la utilidad Report Form de Clipper puede crear dos grupos. Con otros generadores, puede crear hasta nueve, pero con el motor de informes no hay límite.

Si desea que los totales de los grupos se impriman, debe al menos totalizar una columna.

Los datos deben ordenarse de acuerdo con la expresión del grupo.

Para crear un grupo, siga los siguientes pasos:

```
REPORT oReport
      TITLE “Grupos”

COLUMN TITLE “Código”;
     DATA oCursor:CODIGO

COLUMN TITLE “Apellido”;
     DATA oCursor:APELLIDO

GROUP ON oCursor:COD_CARGO;
     EJECT

END REPORT
```

El informe totalizará las columnas creadas con la cláusula TOTAL para cada Cargo. Asimismo, se expulsará la página al cambiar de Cargo. Recuerde solicitar la consulta SQL con la cláusula “ORDER BY COD_CARGO”

Puede incluso especificar una cabecera, pie y tipo de letra específicas para ese grupo (sólo se admite una línea en la cabecera y el pie):

```
GROUP ON oCursor:COD_CARGO;
FOOTER "Total Cargo:";
FONT 2;
EJECT
```

Recuerde que puede crear tantos grupos como desee.

**INFORMES DE RESUMEN**

Si especifica la cláusula SUMMARY al crear el informe:

```
REPORT oReport TITLE ... SUMMARY
```

Sólo se imprimirá la información sobre grupos. La cláusula SUMMARY sólo debe usarse cuando, al menos, haya un grupo definido.

Al utilizar la cláusula SUMMARY, el motor de informes no añade ninguna línea de separación

de grupos.

**CONTROL DEL FLUJO DEL INFORME**

Al activar el informe, es posible definir condiciones FOR y WHILE. Por ejemplo:

```
ACTIVATE REPORT oReport;
WHILE oCursor:COD_CARGO= "Ventas";
FOR oCursor:SALARIO > 100000
```

Puesto que el motor de informes puede crear un informe incluso a partir de un array, la condición WHILE es, por defecto, "!oCursor:Eof()". Si utiliza la condición WHILE y genera un informe de una base de datos, es aconsejable incluir en la condición WHILE el texto seguido de ".and. !oCursor:Eof()":

```
ACTIVATE REPORT oReport;
        WHILE oCursor:COD_DPTO= "Compras" .AND. !oCursor:Eof();
        FOR oCursor:SALARIO > 100000
```

Puede controlar el flujo del informe de la misma forma que controla otras ventanas.

Al activar el informe, puede indicar qué funciones serán llamadas cuando el informe se encuentre en un estado particular:

```
ACTIVATE REPORT oReport ;
        ON INIT ... ;
        ON END .... ;
        ON STARTPAGE ....;
        ON ENDPAGE ... ;
        ON STARTGROUP .... ;
        ON ENDGROUP .... ;
        ON STARTLINE ... ;
        ON ENDLINE .... ;
        ON CHANGE ...
```

La función INIT se llama una sola vez, al inicio de la primera página una vez que los títulos de columna han sido impresos.

La función END se llama una sola vez, en la última página y tras imprimir los totales generales.

La función STARTPAGE se llama al inicio de cada página. Es de las más usadas, puesto que con ella se pueden añadir bitmaps, líneas, cajas, etc...

La función ENDP AGE se llama al final de cada página, cuando todo el texto se ha impreso (no es muy útil).

La función STARTGROUP se llama al inicio de cada grupo y antes de imprimir la cabecera de grupo (si se definió).

La función ENDGROUP se llama al final de un grupo.

La función STARTLINE se llama al inicio de cada línea del cuerpo del informe. Cuando STARTLINE se evalúa, el motor controla si la siguiente línea cabe en la página actual. Si no es así, se expulsa la página y se inicia una nueva. Tenga cuidado al usar STARTLINE, ya que si existen columnas multilínea se producen al menos dos STARTLINE por cada registro procesado.

La función ENDLINE se llama al final de cada línea del cuerpo del informe. Cuando se evalúa ENDLINE, toda la línea se ha impreso, y el número de línea actual del informe se incrementa.

La función CHANGE se llama en cada "oCursor:DbSkip()" si está listando un cursor de datos. Por razones prácticas, esta función se llama antes del oCursor:DbSkip() y no después. Es una cláusula que se usa con frecuencia.

En el código fuente, se ha incluido una línea en blanco en StartGroup y la emisión de un bip al finalizar el informe.

## EL OBJETO REPORT

Es posible acceder a cualquier parte del informe si conoce cómo funciona el objeto Report y de qué forma ha sido diseñado.

El objeto Report tiene muchos datos y métodos que debe conocer y utilizar.

Como mínimo, incluye un objeto Column, un objeto Line (para el título), y un objeto Device. Si existen grupos, también tiene un objeto Group.

Los objetos TRColumn, TRLine y TRGroup son la base del motor de informes.

TRColumn contiene todas las columnas del informe, incluyendo su título, los datos, los totales, etc.. (objeto vertical)

TRLine contiene todas las líneas, que aparecen en el informe de izquierda a derecha. Puede considerarlas como objetos horizontales. Aquí se incluyen los títulos, cabeceras y pies.

TRGroup contiene toda la información sobre cada grupo definido en el informe.

Puede manipular el comportamiento de cualquier objeto cambiando sus datos o utilizando algunos de sus métodos.

Examine los datos y métodos de las clases. Quedará impresionado de todo lo que puede hacer. El único límite es su imaginación.

Ejemplos:

1. Cambiar el tipo de letra del título de la columna 3 para usar el segundo tipo. Dado que la columna es multilínea, cambiar el picture de la segunda línea a "999,999":

```
LOCAL oColumn

oColumn := oReport:aColumns[3]
oColumn:bTitleFont := {|| 2} // Es un codeblock
oColumn:aPicture [2] : = "999,999"
```

2. Justificar a la izquierda la primera línea del título y usar el tipo de letra 2. Justificar a la derecha la segunda línea y usar el tipo de letra 3:

```
#define LINE_LEFT 1
#define LINE_RIGHT 2

LOCAL oLine

oLine := oReport:oTitle
oLine:aPad[l]:= LINE_LEFT// justificar línea 1
oLine:aPad[2]:= LINE_RIGHT // justificar línea 2
oLine:aFont[l]:= {|| 2} // tipo de letra línea 1
oLine:aFont[2]:= {|| 3} // tipo de letra línea 2
```

Como puede ver, el objeto TRLine contiene múltiples líneas, por lo que la mayoría de datos son arrays.

3. En el pie de cada grupo, imprimir el valor actual y el número de registros incluidos en el grupo:

```
GROUP ON oCursor:COD_DPTO ;
   FOOTER "Total Departamento"+;
   oReport:aGroups[l] :cValue+ ;
   str(oReport:aGroups[l] :nCounter)
```

**RECORRIDO HACIA ATRÁS**

Basta con ejecutar oCursor:GoBottom(), cambiar la forma en que el informe efectúa los saltos, y modificar la condición While:

```
oCursor:=Opentable("SELECT * FROM NMDPTO",.T.)
oCursor:GoBottom()

REPORT oReport ....
COLUMN ...
END REPORT

oReport:bSkip := {| |oCursor:DbSkip(-I)}

ACTIVATE oReport WHILE !oCursor:Bof()
```

**IMPRESIÓN DE ARRAYS**

Use una variable estática para mantener el elemento actual del array y cambie los codeblocks Skip y While.

```
STATIC nField

Function Main ( )

nField := 1

REPORT oReport ....
COLUMN ...
COLUMN ...
END REPORT

oReport:bSkip := {| |nField++}

ACTIVATE REPORT oReport ;
WHILE nField <= len(aStructure)
```

**SOMBREADOS Y REJILLAS**

Puede usar dos cláusulas adicionales al crear sus columnas:

- Shadow: Imprime en gris claro el fondo de la columna.
- Grid: Añade dos líneas verticales a ambos lados de la columna.

  Por ejemplo:

  COLUMN TITLE ... SHADOW GRID

  Puede modificar el tipo, anchura y color de la rejilla. Para ello, debe crear las plumas necesarias:

  COLUMN TITLE ... GRID <nPen>

  Por defecto, la pluma usada es negra, sólida y con un grosor 1.

  La creación de plumas es idéntica a la de tipos de letra. (Consulte la Guía Norton para más información sobre plumas.)

  Por ejemplo:

```
DEFINE PEN oPen WIDTH 5

REPORT oReport PEN oPen
COLUMN TITLE .... GRID 1
END REPORT

ACTIVATE REPORT
```

**CAMBIO DEL ASPECT**

Ciertos datos de los objetos de un informe le permiten cambiar el aspecto del mismo. Por ejemplo:

- nTotalLine: Es el aspecto para las líneas de total. Por defecto es RPT_DOUBLELINE (2)
- nGroupLine: Es el aspecto para las líneas de grupo. Por defecto es RPT_SINGLELINE (1)
- nTitleUpLine: Es el aspecto para las líneas de los títulos de columna. Por defecto es RPT_DOUBLELINE (2)
- nTitleDnLine: Es el aspecto para las líneas inferiores de los títulos de columna. Por defecto es RPT_DOUBLELINE (2)
- cPageTotal: Es una descripción para el total de página. Por defecto es ""

- cGrandTotal: Es una descripción para el total general. Por defecto es ""

y también algunos métodos como:

- Margin(nValue, nType, nScale): Este método le permite cambiar los márgenes superior, inferior, izquierdo y derecho del informe, en pulgadas o centímetros.

- Say(nCol, xText, nFont, nPad, nRow): Este método le permite escribir el texto que desee en el informe, indicando la columna (donde el objeto columna empieza), el texto, el número de tipo de letra a usar, el número de alineación (izquierda, derecha, centro) y la fila (la actual por defecto ).

**BITMAPS**

Añadir un bitmap a su informe es muy simple. Mediante STARTPAGE, llame a una función propia que dibuje el bitmap empleando el método SayBitmap de TReport.

Por ejemplo:

```
ACTIVATE REPORT oReport ;
ON STARTPAGE Mybitmap()

Function Mybitmap()

        oReport:SayBitmap(.3, .3,"BITMAPS\LOGO.BMP", .5, .5)

RETURN NIL
```

Estos son los parámetros del método Saybitmap:

- nRow
- nCol
- cBitmap (fichero)
- nWidth
- nHeight
- nScale

Si nScale es INCHES (1), entonces 1, 2, 4 Y 5 están en pulgadas; si nScale es CMETERS (2), 1,2, 4 Y 5 están en centímetros.

## CUADROS Y LÍNEAS

Añadir un cuadro o línea al informe es sencillo. Mediante STARTPAGE, llame a una función propia que dibuje una línea o cuadro empleando los métodos Line o Box de TReport.

Por ejemplo:

```
ACTIVATE REPORT oReport ;
ON STARTPAGE Mybox()

Function Mybox ()
        oReport:Box(.l, .1, 11.1, 7.6)
        RETURN NIL
```

Estos son los parámetros del método BOX:

- nRow
- nCol
- nBottom
- nRight
- nPen
- nScale

Estos son los parámetros del método LINE:

- nTop
- nLef
- nBottom
- nRight
- nPen
- nScale

## IMPRESIÓN DE CAMPOS MEMO

Imprimir campos memo resulta difícil en muchos generadores de informes, pero es bastante sencillo con el motor de informes de FiveWin.

Vea cómo se hace:

- Cree una columna con la cláusula DATA vacía, pero indique el tamaño (anchura de línea en caracteres) que desea para el campo memo:

```
COLUMN TITLE "Comentarios" DATA" " SIZE 50
```

- Cree una función que será llamada por la cláusula ON CHANGE, como ésta:

```
Function SayMemo()

LOCAL nLines, nFor
nLines := Mlcount(oCursor:Coment,50)

/*
Puesto que ON CHANGE hace que la línea se incremente, debemos decrementar la línea
actual en uno.
*/

oReport:BackLine(l)

/*
Ahora podemos empezar a imprimir. Recuerde que antes de imprimir algo, debe llamar a
StartLine, y que después debe llamar a EndLine. De esta forma, el programa controla los
saltos de página.
*/

FOR nFor := 1 to nLines
oReport:StartLine()
oReport:Say(nMemoColumn , ;
MemoLine(oCursor:Coment,50,nFor))
Report:EndLine()
        NEXT

RETURN NIL
```

También es posible convertir el memo en arreglos e imprimirlo:

```
Local aLines
aLines:=_VECTOR(oCursor:cMemo,CHR(10))
FOR nFor := 1 to LEN(aLines)
oReport:StartLine()
oReport:Say(nMemoColumn,aLines[nFor])
Report:EndLine()
        NEXT
```

Y esto es todo lo que debe hacer.

## COLORES

Puede cambiar el color de cada tipo de letra usando el método SetTxtColor.

SetTxtColor() recibe dos parámetros. El primero indica el nColor a usar (consulte Colors.ch), y el segundo, el número de tipo de letra en el informe.

Por ejemplo:

oReport:SetTxtColor(CLR_HRED,I)

También puede controlar el color de cualquier pluma indicando sus colores al crearla. Por ejemplo:

DEFINE PEN oPen COLOR CLR BLUE

Si desea cambiar el color de la pluma horizontal usada en títulos y totales, use el método SetPenColor.

Por ejemplo:

oReport:SetPenColor(CLR_RED)

También puede cambiar el color de las sombras mediante el método SetShdColor(nNewColor), que, por defecto, es LIGHTGRAY.

## IMPRESIÓN DE UN FICHERO TXT

En Rept23.PRG se utiliza la técnica de impresión de memos para imprimir un fichero de texto. Resulta muy útil para convertir rápidamente informes de Clipper para DOS a Windows.

Por ejemplo, si su aplicación DOS crea informes mediante R&R Code Generator o Bandit, use la opción de informes DOS para enviar los informes a un archivo .TXT. Luego, adapte el código de este ejemplo para que FiveWin imprima los archivos .TXT con los informes. Asegúrese de que el tipo de letra especificado en el código se corresponde con el usado para la creación del informe. Imagine lo que ello supone: puede convertir sus actuales informes DOS a Windows en menos de 20 minutos, sin importar su complejidad.

Por supuesto, si existe código para solicitar o responder a las diferentes opciones del informe en los .prg de DOS, deberá convertido a FiveWin.

**Nota:** Al previsualizar este informe, puede aparecer borroso debido al uso del tipo de letra de paso fijo. Al imprimirse, no aparecerá borroso.

## INCLUSIÓN DE FORMATOS ELABORADOS EN CRYSTAL REPORT

Facilita la asociación de uno varios archivos creados a través del excelente diseñador de Reportes "Crystal Report".

Tiene como finalidad utilizar diseños Crystal para presentar con alta calidad los cursores de datos diseñados con el generador de reportes Datapro. Esta combinación logra integrar en un sólo producto: La generación del cursor de datos empleando léxico SQL y las innumerables posibilidades que brinda la emisión de reportes de Crystal.

Para facilitar el diseño de reportes Crystal, Datapro genera dos tablas de datos en formato DBF contentivas del cursor de datos SQL y los parámetros del reporte permitiendo un rápido y fácil diseño del reporte Crystal, básicamente Datapro se encarga de Generar la consulta SQL según los parámetros de Rango y Criterio del Reporte, convierte el cursor de datos en una tabla DBF para conformar el cuerpo del reporte y transfiere todos los parámetros del sistema y reporte hacia una segunda tabla DBF para ser utilizada como encabezado. El usuario debe emplear estas dos tablas para el diseño del reporte. También puede crear diversos diseños con la misma tabla DBF y asociarlas al reporte brindándole al usuario la posibilidad de escoger una de ellas cuando solicite la emisión del reporte con salida hacia Crystal Report.

Los nombres de los archivos DBF creados para el Diseño Crystal se conforman con el mismo código del reporte y se encuentran en la carpeta c:**aplicaciónDatapro**\crystal, donde **aplicaciónDatapro** puede ser: DpNmwin, DpAdmWin, etc. Sólo se generan cuando se solicita la emisión del reporte hacia Crystal Report, bajo la siguiente nomenclatura:

- "t"+Codigo+".dbf" para el cuerpo, detalle o columnas del reporte quedando así: t0000000001.dbf.
- "t"+Código+"_.dbf" para el encabezado y parámetros" quedando así t t0000000001_.dbf

El objetivo de agregar como prefijo la letra "t" evita posibles errores de ALIAS() cuando el nombre del archivo no se crea con una letra.

El archivo del cuerpo, contiene el cursor de datos de la consulta compuesto por registros que representan las líneas de las consultas y los campos de la tabla representan las columnas de la consulta.

El archivo de "Encabezado o Título", contiene información que permite describir al reporte, aquí se encuentran los siguientes datos:

| | |
|---|---|
| USUARIO | Código del Usuario |
| USNOMBRE | Nombre del Usuario |
| FECHASIS | Fecha del Sistema |

| | |
|---|---|
| EMPRESA | Nombre de la Empresa |
| TITULO | Título del Reporte |
| RANGOnn | Datos del Rango (nn) es la posición desde 01.. |
| CRITERIOnn | Datos del Criterio (nn) es la posición desde 01.. |

Para asociar un archivo Crystal Report con el reporte Datapro, el sistema provee un editor de registros que muestra la lista de los archivos RPT ya asociados y una breve descripción de su uso.

Este se encuentra en el modo diseño de cada reporte representado por el icono .

Presenta la ventana de dialogo para incluir un nuevo vinculo con un archivo de extensión .RPT de Crystal Report que solicita la descripción y el nombre del archivo. Preferiblemente debe estar ubicado en la carpeta "CRYSTAL\ " y el sistema validará que exista el archivo indicado. Presione el botón para aceptar los datos introducidos.

Presenta la ventana de dialogo para cambiar la descripción y el nombre del archivo.

Elimina de la lista la línea (registro) representada por la descripción y el nombre del archivo. El archivo de crystal no será eliminado del computador.

Acepta los contenidos de la lista, los graba para el reporte que está en diseño, cierra la ventana de dialogo y regresa hacia el editor de registros

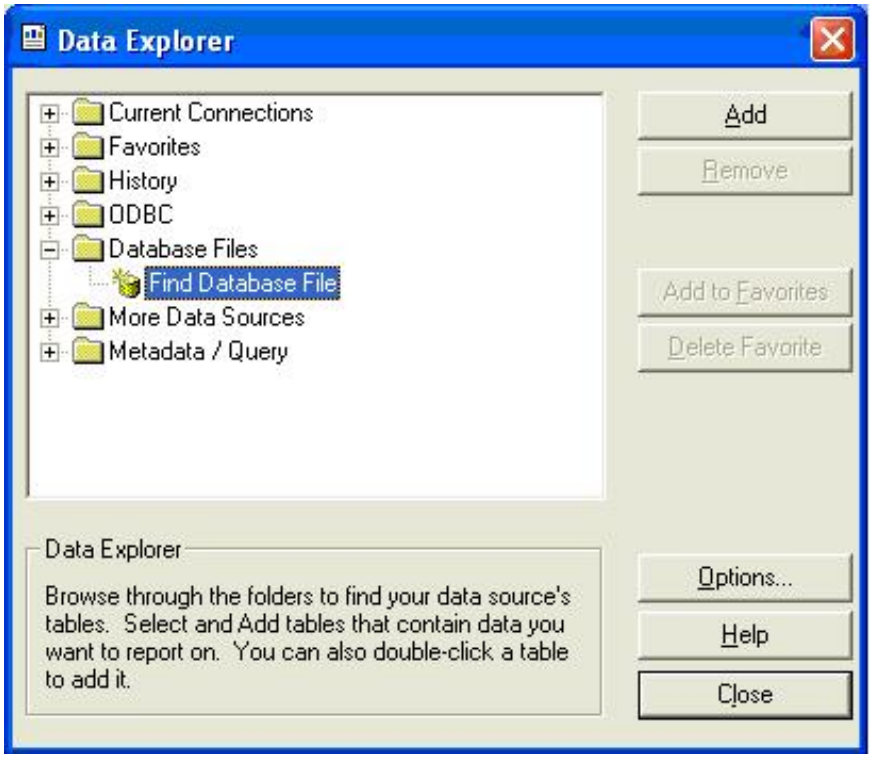
No efectúa ningún cambio, cierra la ventana de dialogo y regresa hacia el editor de registros.

**Consideraciones:**

Si la estructura de la consulta SQL del reporte posteriormente es modificada o alterada debe rediseñar los formatos en Crystal, ya que éstas estarán apuntando a consultas desactualizadas definidas en la versión anterior del reporte.

En ningún momento el sistema validará si el contenido del reporte Crystal es válido, cualquier error que genere Crystal Report deberá ser corregido en su propio editor.

Antes de crear el diseño en Crystal es necesario construir previamente el reporte Datapro y solicitar su ejecución, seleccionar salida para Crystal "Preview", para que el sistema construya las tablas "DBF" que empleara el archivo "RPT" de Crystal que aun no existe. Inmediatamente acceda a Crystal Report e inicie el diseño del reporte, seleccione las dos tablas "Datos" y "Títulos", "t0000000001.dbf" y "t0000000001_.dbf" desde la carpeta "c:\dpaplicacion\crystal". Recuerde que en Crystal debe seleccionar la opción de datos " Database Files " para emplear tablas "DBF", luego indique la ruta de los datos y finalmente Add para adicionar la tabla seleccionada en la lista de tablas que serán empleadas en la construcción del reporte.




**CONDICIONES PARA EL USO DE CRYSTAL REPORT CON DATAPRO**

1. Para construir reportes en Crystal el usuario debe poseer el entorno de diseño Crystal Report, el cual "**NO"** forma parte de los Sistemas Datapro y "**NO**"se encuentra incluido en el contrato de licencia de los Sistemas Datapro.

2. Es necesario instalar los Dll's de Crystal Report en la ruta windows/system32 necesarios para ejecutar los reportes construidos bajo dicho entorno, el proceso de instalación del sistema los incluye y registra automáticamente en el equipo por el hecho de tener formatos ya prediseñados con extensión RPT y por razones de comodidad para el usuario, sin compromiso

por parte de Datapro sobre su utilización y/o funcionamiento.

3. Para la creación de los reportes Crystal "*.rpt" incluidos en este producto ha sido empleada la versión 8.x en español. Otras versiones aun no han sido probadas con Datapro.

4. Cada archivo de diseño Crystal debe ser distribuido en todos los PC donde se ejecuta la aplicación Datapro sobre la ruta c:\dpaplicacion\crystal. Datapro no se encarga de esta función.

5. Los diseños incluidos han sido creados para visualizar y/o emplear el contenido de tablas DBF generadas por Datapro, obtenidas de cursores de datos suministrados por el gestor de base de datos.

6. Este manual no incluye información o explicación sobre la construcción o diseño de reportes en Crystal Report.

# EJECUCIÓN DEL REPORTE

## FIJAR PARÁMETROS

Almacena total o parcial todos los valores incorporados como: rango, criterio y parámetros necesarios para la ejecución del reporte. Estos valores serán recuperados para futuras ejecuciones presentando el formulario de impresión con los mismos valores encontrados en el momento de ser fijado. estos valores son almacenados en un componente dataset que permitirá también recuperar los parámetros fijados desde otro PC.

Para fijar los valores, se presenta un dialogo con dos secciones: Componentes, que permite seleccionar: Rango, Criterio, Parámetros o Todos y la segunda sección permite seleccionar el modo de almacenamiento: Para este PC, para el usuario que lo ejecuta o para todos los PC o todos los usuarios.

Cuando se almacena el Rango o Criterio, conjuntamente se incluyen los operadores relacionales y lógicos empleados por cada uno de ellos. Los parámetros se refiere a los valores del formulario de impresión distinto a Rango o Criterio, estos pueden ser: Dispositivo de Salida, archivo RPT de Crystal Report, modelo de impresora, etc.

**INICIAR PARÁMETROS**

Vacía o borra todos los valores existentes en las casillas de las secciones Rango y Criterio, además presenta una ventana de dialogo que permite borrar del DataSet los valores previamente almacenados, evitando que en las futuras ejecuciones los valores de rango y criterio contengan algún dato.

Este formulario, presenta tres opciones: Rango, Criterio o Todos, permite indicar que componente será iniciado o blanqueado, si es seleccionado “Todos”, los valores de rangos y

criterios quedaran vacíos.

Borra del DataSet todos los valores almacenados para: Rango, Criterio y Parámetros del reporte, es decir; las nuevas ejecuciones del reporte presentará Rangos y Criterio sin valores en las casillas.

# GLOSARIO DE TÉRMINOS

## REPORTE

Un reportes es, básicamente, un programa diseñado exclusivamente para mostrar información obtenida de la consulta o query de una o varias tablas, constituye una de las salidas de todo sistema. Las instrucciones que posee están diseñadas para proporcionar información distribuida en columnas y cada columna posee características que definen la presentación física del informe. La ejecución de reporte genera la presentación visual y modelada de la información solicitada. Las instrucciones del reporte definen: Encabezado, cuerpo compuesto de columnas, grupos o rupturas de control, totales y pié de página. Cada columna puede representar un campo de la consulta SQL o una expresión.

## DEFINICIÓN DE BASES DE DATOS

Conjunto de datos almacenados de manera estructural a través de una o varias tablas en un medio magnético (disco) haciéndose posible el acceso de manera sencilla, con la posibilidad de relacionarlos, ordenarlos en base a diferentes criterios, etc. Cada unidad de información almacenada se denomina registro; y cada uno de estos se compone por uno o varios subconjuntos de información denominados campos. Los campos están compuestos por cadenas de dígitos o caracteres y se clasifican según el tipo de información que almacenan.

## TABLA

Una tabla es una estructura de columnas y filas que almacena datos. Donde cada columna representa un campo, y el conjunto de campos componen un registro y cada registro obedece a datos específicos o únicos como por ejemplo: Trabajador, Departamento, Cliente, Proveedor, entre otros.

## REGISTRO

Es una fila, o un conjunto de columnas. La serie de campos o datos que forman una información es lo que se denomina un registro. Los datos como nombre, apellido, cédula, edad, cargo, profesión y sexo de un trabajador, por citar sólo un ejemplo, conformarían la información pertinente al mismo. Cada dato se une para formar la información. Así pues, cada campo se une para formar un registro.

## CAMPO

Los campos son las columnas de una tabla, y la unión de ellos conforman un registro. Almacenan diferentes tipos de datos, cada campo se identifica a través de un nombre y posee parámetros como: Tipo de información que almacena, longitud del campo, cantidad de decimales y en algunos motores de datos contiene reglas o condiciones. La longitud del registro lo conforma la sumatoria de todas las capacidades de cada campo.

## RELACIÓN ENTRE TABLAS

Es el vinculo entre dos o más tablas, cada vinculo tiene una regla de deben cumplir las tablas involucradas "Identidad y relación", es decir; todas las tablas enlazadas deben poseer elementos similares que facilite establecer la regla de enlace. Ejemplo: Relación entre trabajadores y departamentos.

La tabla de trabajador posee el campo COD_DPTO y la tabla departamento posee DEP_CODIGO, estos dos campos son comunes permitiendo la relación.

EL diccionario de datos posee registrada la relación entre todas las tablas de la base de datos, ofreciendo al usuario la posibilidad de seleccionar cualquier tabla con las relaciones disponibles sin necesidad de escribir ningún comando, así puede crear un reporte con tan solo seleccionar con Clic la tabla y las relaciones necesarias. Algunos motores de bases de datos ofrecen integridad referencial para los procesos de actualización o mantenimiento de registros.

## INTEGRIDAD REFERENCIAL

Algunos motores de bases de datos ofrecen esta potente funcionalidad que consiste en establecer en cualquier tabla la relación con otra para así afectarlas directamente en caso que la tabla "Padre" sea actualizada. Esta funcionalidad se llama integridad referencial, ejemplo: Si existe una tabla de padres y esta relacionada con la tabla "Hijos" y la regla entre ambas se establece con el campo

"apellido", en caso de ser cambiado el apellido del algún padre todos los hijos también cambian automáticamente el apellido o también en caso de eliminar algún registro de la tabla de "padres" también son eliminados los registros de las tablas "hijos", esta labor la realiza de manera automática el gestor o motor de la base de datos, igualmente no es posible crear registros en la tabla "Hijos" si no existen padres con quien relacionarlos.

## CLAVE PRIMARIA

Es la condición dada a un campo de la tabla que identifica a cada registro que requiera ser único en toda la tabla es decir que su valor no se repita, como por ejemplo el código del trabajador no puede permitir la creación de dos registros con el mismo código. Los motores de las bases de datos se encargan de hacer cumplir esta regla.

## TIPOS DE DATOS

Todos los datos están representados mediante un identificador que permite acceder (escribir o leer) su contenido. Estos identificadores se denominan Variables y Campos; los cuales presentan los mismos tipos de datos que difieren sólo en el origen. Los campos son subunidades de información no volátiles y ubicados en los registros de las bases de datos que se encuentran en disco. La información de la variable es volátil y su contenido es manipulado directamente desde la memoria del computador.

Se deben conocer los diferentes tipos de datos que se pueden manipular. Estos datos se obtienen directamente de las bases de datos, de la ejecución de expresiones y de las variables creadas directamente por el usuario. Los tipos de datos son:

**Numéricos:** Están compuestos por dígitos (0123456789.), su valor es cuantitativo y se pueden obtener resultados mediante la aplicación de cualquier ecuación matemática.

**Alfanuméricos:** Están compuestos por cadenas de caracteres, no poseen valor cuantitativo, y sus expresiones deben estar encerradas o delimitadas entre comillas dobles ("") o comillas sencillas (''). Ejemplo:

**Fecha:** La longitud de su contenido es fija, se presenta en el formato dd/mm/aaaa; donde **"dd"** se refiere al día, **"mm"** al mes y **"aaaa"** al año. Los datos tipo fecha aceptan la aplicación de sumas o restas.

**Lógicos:** Resultados booleanos obtenidos de una pregunta donde la respuesta es **"Verdadero"** o **"Falso**". El valor verdadero se representa por la expresión .T.; y el valor falso por la expresión .F. El contenido de estas expresiones se emite asociándolas con la función de formato: F() o TRANSFORM().

Cabe destacar que DataPro utiliza datos compatibles con el modo xBase para su entorno operativo y para la gestión de bases de datos utiliza los modelos soportados por SQL. Debido a que algunos modelos de datos no son compatibles, DataPro se encarga de realizar las conversiones de formato xBase hacia el modelo de datos SQL. Como por ejemplo:

| XBASE | SQL | Dato |
|---|---|---|
| Logical | Bit | Lógico |
| Memo | Text | Memos |
| Numeric | Numeric | Numérico Max 20 |
| Date | Date | Fecha |
| Char | Char | Carácter |

**OPERADORES**

Elementos que el usuario debe utilizar para crear las relaciones, fórmulas y operaciones entre los datos contenidos en una expresión. Los operadores se clasifican en:

**Aritméticos:** Símbolos utilizados para efectuar operaciones aritméticas con datos del mismo tipo. Su resultado debe ser asignado sobre un elemento de identificación llamado **"VARIABLE"**.

- **Asignación (=):** Indispensable para asignar hacia una variable el resultado obtenido de la ejecución de una expresión, según la siguiente sintaxis: VARIABLE = Expresión.

- **Adición (+):** Se utiliza para agregar el contenido de uno o varios datos en otro. Puede ser aplicable en los datos de tipo numérico, carácter y de fecha

- **Resta (-):** Debe ser utilizada para sustraer los valores de uno o varios datos con otros. Sólo es aplicable para los datos de tipo **"numérico y fechas"**

- **Multiplicación (*):** Debe ser utilizada para multiplicar los valores de uno o varios datos con otros. Sólo es aplicable para los datos de tipo **"numérico"**.

- **División (/):** Debe ser utilizada para efectuar la división entre el dividendo y el divisor. Cuando el valor del divisor es cero **"0"**, se presenta el error **"División por cero"**. Este error se evita utilizando la función **"DIV(DIVIDENDO,DIVISOR)"**, que genera el mismo resultado. La división sólo permite la utilización de expresiones de tipo numérico.

- **Exponencial(**):** Obtiene el resultado de un valor elevado a un exponente, según la sintaxis: X**n; donde (n) es el exponente.

**Relacionales:** Símbolos utilizados para efectuar comparación entre datos del mismo tipo. Se pueden emplear con expresiones de tipo numérica, alfanumérica, de fecha y lógicas.

- Los símbolos relacionales son:

= IGUAL A
# DIFERENTE
> MAYOR QUE
< MENOR QUE
=> MAYOR O IGUAL
=< MENOR O IGUAL
$ CONTENIDO EN
() AGRUPACION

El operador $ solo es aplicable en las expresiones de tipo alfanumérico **"textos o cadenas"**. Este operador permite determinar si el contenido de una expresión alfanumérica se encuentra exactamente igual dentro del contenido de otra.

El operador de agrupación **"()"** es aplicable en todos los tipos de datos y se utiliza para la agrupación de comparaciones o expresiones aritméticas. También son muy utilizados para agrupar relaciones entre si, mediante operadores relacionales.

**Lógicos:** Se utilizan para relacionar expresiones de tipo lógico **"Verdadero o Falso"**, generando también resultados de tipo lógico. Estos operadores son:

.AND. = Y
.OR. = O
.NOT. = NO (Negación)

La negación también se puede indicar con .!.. Si se asocia con otra expresión lógica, genera los siguientes resultados:

(.!.)(.F.) = .T. No falso es verdadero.
(.!.)(.T.) = .F. No verdades es falso.

## DICCIONARIO DE DATOS

Está compuesto por un conjunto de tablas que poseen información exclusiva de todo lo relacionado con las tablas, campos, índices, menú de acceso, programas fuentes, relaciones entre tablas y opciones para cada campo. El diccionario de datos puede informarnos sobre el uso de un campo y si esta relacionado con la clave de otra tabla o también si se rellena con ceros hacia la izquierda. También puede informar sobre los campos de una tabla, los índices que lo conforman y las posibles relaciones con tablas incluyendo integridad referencial. Prácticamente el Diccionario de datos es el administrador del sistema y todos los componentes de DataPro obedecen a las reglas establecidas en él, como por ejemplo las sugerencias brindadas por el generador de reportes.

# REFERENCIA DE COLORES

## BAJA DENSIDAD

| Color | Definición | Referencia Decimal | Color RGB |
|---|---|---|---|
| Negro | CLR_BLACK | 0 | RGB( 0, 0, 0 ) |
| Azul | CLR_BLUE | 8388608 | RGB( 0, 0, 128 ) |
| Verde | CLR_GREEN | 32768 | RGB( 0, 128, 0 ) |
| Cyan | CLR_CYAN | 8421376 | RGB( 0, 128, 128 ) |
| Rojo | CLR_RED | 128 | RGB( 128, 0, 0 ) |
| Amarillo | CLR_MAGENTA | 8388736 | RGB( 128, 0, 128 ) |
| Marrón | CLR_BROWN | 32896 | RGB( 128, 128, 0 ) |
| Gris | CLR_HGRAY | 12632256 | RGB( 192, 192, 192 ) |
| Gris plateado | CLR_LIGHTGRAY | CLR_HGRAY | |

## ALTA DENSIDAD

| Color | Definición | Referencia Decimal | Color RGB |
|---|---|---|---|
| Gris | CLR_GRAY | 8421504 | RGB( 128, 128, 128 ) |
| Azul | CLR_HBLUE | 16711680 | RGB( 0, 0, 255 ) |
| Verde | CLR_HGREEN | 65280 | RGB( 0, 255, 0 ) |
| Cyan | CLR_HCYAN | 16776960 | RGB( 0, 255, 255 ) |
| Rojo | CLR_HRED | 255 | RGB( 255, 0, 0 ) |
| Amarillo | CLR_HMAGENTA | 16711935 | RGB( 255, 0, 255 ) |
| Amarillo | CLR_YELLOW | 65535 | RGB( 255, 255, 0 ) |
| Negro | CLR_WHITE | 16777215 | RGB( 255, 255, 255 ) |