

CONSIDERACIONES.

Diversas consideraciones a tener en cuenta, para que el compilador y nuestros programas funcionen correctamente.

INSTALACION:

Cuando instalamos Fujitsu PowerCobol, éste se instala en el directorio FSC, debemos de asegurarnos que C:\FSC\PCOBOL32 (directorio que contiene el compilador y los archivos necesarios para ejecutar) están en el PATH. Os aconsejo para trabajar mas cómodo que os creéis una carpeta dentro de FSC y ahí desarrolléis todo, por ejemplo C:\FSC\PROGRAMAS. Luego a su vez para cada proyecto abrimos otra carpeta dentro de ésta y así lo tendremos todo mas organizado.

No utilizéis espacios en blanco ni para el nombre ni para el directorio donde se encuentre, es decir que "Mis Documentos" o "Archivos de Programa" no nos valdrán para guardar nuestros proyectos.

PROYECTO:

El hecho de crear una ventana no significa que tengamos un programa. Siempre hay que empezar por el proyecto o bien crear las ventanas y luego añadirlas a un proyecto. Pero una ventana por separado si no forma parte de un proyecto no se puede ejecutar, puesto que PowerCobol no la va a reconocer. Tened en cuenta que siempre una de las ventanas debe de estar marcada como ventana principal, para que el proyecto cuando se ejecute empiece por algún sitio.

PROGRAMACION:

Prácticamente todo lo que hemos aprendido de programación Cobol se puede utilizar con PowerCobol.

Todos los procedimientos, algoritmos, descripciones de archivos, tablas, todo es totalmente reutilizable. La nueva programación será la que haga referencia a los aspectos gráficos y de representación de nuestros programas.

ARCHIVOS BTRIEVE (RMCOBOL):

Se pueden leer directamente los archivos Btrieve, es decir los que utiliza RM/Cobol, para ello cuando se defina el fichero en la Select, bastará con ponerle después del nombre de archivo las letras BTRV. Cuando vayamos viendo los ejemplos lo veréis mas claramente, pero desde aquí os aseguro que se utilizan perfectamente. Tengo que hacer constancia, después de muchas pruebas, que los campos con signo y computational no son legibles. Leyendo sobre lo que pone en los manuales, dice que power es capaz de leer la estructura de un fichero Btrieve, pero no de sus características.

COMPILACION:

Los pasos necesarios para una compilación y creación de ejecutable, son los siguientes:

Hacer Compile, el Link y el Make, pero ten en cuenta que esto siempre será sobre la ventana que tengamos activa, si deseamos compilar todo el proyecto que tenemos abierto, lo mejor es hacer un Build y entonces hará lo mismo, pero con toda la aplicación. Si nos presenta algún error, éste suele estar bien detallado, incluida la línea y el evento donde se ha producido, con lo que su localización es muy sencilla.

EJECUCION:

Cuando tengamos una aplicación terminada y queramos ejecutarla en otro equipo, será necesario copiar lo siguiente:

- Archivo ejecutable.
- Imágenes e iconos que vayamos a utilizar.
- Los archivos de datos o bases de datos que tengamos creadas.
- Todas las DLL que se encuentran en el directorio C:\FSC\PCOBOL32. Esto es una salvajada, pero como no sabemos cuales de ellas utiliza nuestro programa lo mejor es copiarlas todas y así no hay problemas.
- Luego todo eso lo copiáis en el nuevo equipo en un directorio y lo introducís en el PATH.
- Y nuestra aplicación estará funcionando a la perfección.

RUNTIME ENVIRONMENT SETUP:

Siempre que ejecutamos nos sale una ventana con las variables de entorno que Power puede necesitar, aquí definiremos el tipo de letra que queramos para la impresora o cualquier otra variable de entorno. Si no queremos que aparezca esta ventana, solo tendremos que modificar la línea:

```
@EnvSetWindow=USE
```

Cambiaremos el USE por UNUSE. Esto hará que no vuelva a aparecer mas. Si por el contrario queremos activarla en otro momento, en el directorio del programa tendremos un archivo que se llamar Cobol85.cbr, lo editamos y vemos que es lo mismo que nos sale en la ventana, volvemos a poner USE y la próxima vez que ejecutemos volverá a salir la ventana.

Todo lo que vaya viendo que pueda considerarse como una consideración lo iré implementando en ésta sección.

EL MENU DE POWERCOBOL.

Os daréis cuenta que al iniciar PowerCobol, solo os aparece una ventana ajustada en la parte superior de vuestro escritorio, que dispone de una barra de menú y otra de herramientas y una ventana vacía preparada para empezar nuestro primer programa.

Veamos una explicación rápida de las opciones que nos ofrece el Menú:

Para PowerCobol, las pantallas se llaman **Sheet**, debemos de tenerlo en cuenta, puesto que será uno de los términos que mas utilizemos. Al lado de las opciones que tengan representación en la barra de herramientas os coloco el icono que corresponda.

- **File:** Opción para el manejo de las ventanas.
 - **New:** Crear una ventana nueva para nuestro proyecto.
 - **Open:** Abrir una ventana que hayamos creado.
 Icono en la barra de herramientas.
 - **Save - Save as:** Guardar o guardar como, la ventana con la que estemos trabajando es ese momento.
 Icono en la barra de herramientas.
 - **Exit:** Abandonar PowerCobol.
 - **Lista:** Ultimas ventanas utilizadas.

- **Edit:** Opción para funciones básicos de edición.
 - **Undo:** Deshacer la última acción realizada.
 - **Redo:** Rehacer lo último deshecho.
 - **Cut:** Eliminar el control de la ventana.
 - **Copy:** Copiar el control o parte de texto.
 - **Paste:** Pegar el control o parte de texto.
 - **Delete:** Igual que Cut.
 - **Move:** Mover el control, se hace mas fácil con el ratón arrastrando.
 - **Size:** Modifica el tamaño del control, se hace mas fácil con el ratón.
 - **Item Order:** Especifica el orden que tendrán los controles cuando pulsemos tabular o enter, es muy importante.
 - **Title:** Título del control, en algunos casos tiene poco o ningún sentido. Se accede con el botón derecho del ratón estando situados sobre un control.
 - **Style:** Para definir las propiedades de un control. También se accede desde el ratón con el botón derecho.
 Icono en la barra de herramientas.
 - **Procedure:** Para programar los eventos disponibles para cada control. También se accede desde el ratón.
 Icono en la barra de herramientas.
 - **Menubar:** Aquí definimos el menú para nuestra ventana, con sus opciones y eventos.
 Icono en la barra de herramientas.
 - **Find - Next - Before - Replace:** Para encontrar una determinada palabra o frase dentro del código de uno o todos los eventos de nuestra ventana.
 Icono en la barra de herramientas.
 Icono en la barra de herramientas.

- **Project:** Opción para controlar nuestros proyectos.
 - **Compile:** Compila todo el código de la ventana activa.
 -  Icono en la barra de herramientas.
 - **Link:** Linkea todas las ventanas de nuestra aplicación con las DLL necesarias para su ejecución.
 - **Make:** Crea el ejecutable del proyecto, generará un archivo con el mismo nombre que el proyecto y extensión .EXE.
 -  Icono en la barra de herramientas.
 - **Build:** Lo hace todo, incluido la compilación de todas las ventanas del proyecto, hace el link y el make. Es lo mejor y mas seguro.
 - **Run:** Ejecuta la aplicación.
 -  Icono en la barra de herramientas.
 - **Debug:** Depuración del código mientras se ejecuta.
 - **New:** Creación de un nuevo proyecto.
 - **Open:** Abrir un proyecto ya existente.
 - **Edit:** Añadir o eliminar ventanas, DLL's o iconos a nuestro proyecto.
 - **Close:** Cerrar el proyecto.
 - **Lista:** Ultimos proyectos a los que hemos accedido.

- **Tool:** Para mostrar las distintas ventanas que nos ofrece el PowerCobol:
 - **Status Box:** Ventana que nos mostrará el tamaño y posición de cada uno de los objetos o controles de nuestra ventana.
 - **Item Box:** Sin duda la mas importante y en ella encontraremos todos los controles que podremos poner en nuestras ventanas.
 - **Color Box:** Ventana donde determinaremos el color de fondo y de primer plano de nuestros controles.
 - **Font Box:** Ventana donde seleccionaremos la fuente y el tamaño de nuestros controles.

- **Option:** Definición del entorno de trabajo.
 - **Compile:** Definimos algunas variables para el compilador.
 - **Run:** Definimos algunas variables para el Runtime. (Nunca lo he utilizado).
 - **Sheet:** Definimos las opciones para nuestra ventana de programa.
 - Grid-Width: Ancho de la cuadrícula de ayuda, lo normal 4.
 - Grid-Height: Alto de la cuadrícula de ayuda, lo normal 4.
 - Display Grid: Que se vea o no la cuadrícula, cuando estamos diseñando. Es conveniente.
 - Snap to Grid: Para ajustar los controles que pongamos a la cuadrícula. Es muy conveniente.
 - Auto Size: Para ajustar el ancho de los controles a la rejilla. Es muy conveniente.
 - Status-Box: Si queremos activar la ventana de status.
 - Item-Box: Si queremos activar la ventana de controles.
 - Color-Box: Si queremos activar la ventana de colores.
 - Font-Box: Si queremos activar la ventana de fuentes.

- Keep on top: Si la marcamos las ventanas citadas antes se superpondrán a la ventana de diseño.
- **Procedure:** Se definen los valores para el editor de código.
 - Tab: Marca los tabuladores cuando los pulsamos en nuestras procedures.
 - Return: Marca la pulsación del Return-Enter.
 - Line number: Enumera las líneas.
 - Guide: Nos indica con una guía la posición de las columnas.
 - Indent: Hace que al pulsar Enter, en la siguiente línea se sitúe en la columna donde empezaba la línea anterior.
 - Prompt save: Para guardar dicha procedure antes de compilar.
 - Tab value: Valor en columnas de la tecla Tab.
- **Test:** Previsualización de como va quedando la ventana, tal y como la veremos al ejecutar el programa, solo diseño.



Icono en la barra de herramientas.

- **Window:** Desde aquí cogemos de entre todas, la ventana que queramos como activa o verlas en cascada, similar a la opción que traen casi todos los programas Windows.
- **Help:** Encontraremos una muy valiosa ayuda sobre lo mas importante de PowerCobol.

Siempre que pulsemos el botón derecho de nuestro ratón sobre cualquiera de los controles que tengamos en la ventana o sobre la propia ventana, nos aparecerá un menú flotante como este:

Title...	Title: Accedemos al título del control, en los controles que no tengan título, al pulsar estará desactivada.
Style...	Style: Para modificar las propiedades del control.
Procedure...	Procedure: Para programar los eventos de cada uno de los controles.
Move	Move: Mover el control. Se hace mejor con el ratón.
Resize	Resize: Cambiar el tamaño del control. Se hace mejor con el ratón.
Menubar...	Menubar: Accedemos al control del menú de nuestra ventana, donde podremos añadir, borrar, modificar, programar nuestras opciones.
Item Order...	Item Order: Definimos el orden que tendrán nuestros controles cuando se pase de uno a otro.

AGENDA.EXE EL RESULTADO.

En ésta página os presento el ejecutable de la aplicación que vamos a realizar paso a paso desde el capítulo 6 en adelante.

Antes de bajarla podéis seguir unos consejos para unificar criterios.

Crearos dentro del directorio C:\FSC (donde se encuentra el Power), dos carpetas, una la llamáis: **Agenda** y otra **Agendaex**. En **Agenda** crearemos el proyecto y todo lo que se vaya generando a partir de ahí, hasta conseguir el programa completo. En **Agendaex**, guardáis y descomprimis el archivo que se encuentra aquí, el cual nos servirá para ver como va quedando nuestro proyecto.

Contenido del archivo agenda.zip que vais a descargar:

- Agenda.exe: el programa ejecutable.
- Cobol85.cbr: archivo de configuración de PowerCobol.
- Bloc.bmp: archivo de imagen utilizado en el programa.
- cobol.bmp: archivo de imagen utilizado en el programa.

Descarga  **47 Kb**

- Una vez que lo hayáis bajado, podéis empezar a utilizarlo, simplemente pinchando en el archivo AGENDA.EXE. Debe funcionar a la primera, si a alguno no le funciona que me lo comunique.

Al ser la primera vez que lo ejecutáis, como el archivo de datos no existe, os saldrá una ventana indicando que se va a crear.

El programa consta de 3 pantallas, la principal, donde hacemos el mantenimiento, una de consulta y otra de Ayuda. A la de consulta y la de ayuda, podremos acceder mediante botones o bien por el menú.

Espero que os guste y os animo a seguir el curso donde detallaré todos los pasos para realizarlo. Todo irá explicado desde el capítulo 6 en adelante. Os deseo suerte y espero que os guste.

CAPTITULO 1. PRIMER CAMBIO.

Nuestros programas Cobol, tienen todos un aspecto muy similar, las divisiones, los párrafos y sobre todo un desarrollo secuencial de las acciones que debe de realizar. Ahora todo eso cambia un poco y el peso fuerte de la programación lo llevan los llamados eventos, que no son otra cosa que los procesos a realizar cuando se actúa cualquiera de los componentes u objetos que tengamos definidos en nuestra pantalla. Vamos a empezar por dar una pequeña explicación de todo lo que nos vamos a encontrar en adelante y con lo cual deberemos de empezar a familiarizarnos.

- **Proyecto**: Es la base para empezar a desarrollar nuestras aplicaciones y en ellos estarán todas las pantallas, las imágenes e iconos.
- **Objeto**: Será todo aquello que definamos gráficamente en nuestro programa, serán objetos un Push-Button, un Label, un Combobox, una Tabla, una Imagen, etc Cada objeto deberá de tener un nombre como si de una variable se tratara. En principio el programa asignará un nombre por defecto pero nosotros podremos variarlo y darle el que mas nos apetezca, ese nombre será el que utilicemos para todo lo que haga referencia a ese objeto. Las pantallas que diseñemos serán los contenedores de objetos, es decir donde definamos todos los objetos que deseemos en nuestra aplicación. Pero a su vez también será un objeto con sus propiedades, sus métodos y sus eventos. En cada aplicación puede haber mas de una pantalla, pero una será siempre la principal.
- **Propiedades**: Serán las distintas opciones que puede tener un objeto y pueden ser comunes o distintas según el tipo de objeto. Serán propiedades, el color, la altura, la anchura, el título, el tipo de letra, si está o no disponible, si está o no visible, etc ... Las propiedades suelen tener un nombre pre-definido por el lenguaje que lo haya designado. Muchas de las propiedades tendrán un valor de tipo SI-NO. Estas propiedades normalmente se podrán establecer tanto en tiempo de diseño como en tiempo de ejecución, es decir, cuando se diseña la pantalla y desde el control del programa, esto es una gran ventaja como veremos mas adelante.
- **Métodos**: Son procedimientos que ya vienen programados por el lenguaje y nosotros solo tendremos que llamarlos para que actúen. Son métodos, añadir campos a un Combobox, enviar el foco a cualquier objeto, abrir una ventana, cerrarla, etc ... Como veremos cada uno de los objetos puede tener sus propios métodos.
- **Eventos**: Serán las acciones del usuario sobre el programa. Como el click sobre un Push-Button, pulsar Return o Tab en un Edit, etc Al producirse el evento, se ejecutará todo lo que le hayamos programado y actuará en consecuencia.

Todo esto, lo que nos va a suponer, es que vamos a perder un poco el "control" sobre nuestro programa, puesto que ya no lo vamos a ver como siempre en un editor y de una manera secuencial, sino que cada evento y propiedades tendrán que ser vistas por separado. Pero os aseguro que eso no es un inconveniente, digamos que al principio es un poco chocante. En los próximos capítulos iré explicando el funcionamiento de PowerCobol en su versión 3, la compilación, la ejecución, los objetos o controles, las propiedades y todo lo necesario para generar nuestras aplicaciones en éste entorno de desarrollo.

CAPTITULO 2. LA PRIMERA VENTANA.

Al abrir el Fujitsu PowerCobol (en adelante Power), nos sale automáticamente una ventana denominada SHEET en blanco y las ventanas de control que tengamos por defecto, pueden ser la de Status, la de Font, la de Color y la de Item, Si alguna no se abre o bien otra queremos cerrarla, bastará con activarla en el menu Tool.

Sobre ésta primera ventana ya podemos empezar a poner nuestros controles u objetos y darles vida. Pero antes de nada aprendamos mas acerca de la ventana, que en realidad es otro objeto mas de nuestra aplicación.

Veamos cuales son sus propiedades.

- **Sheet name:** Será el nombre que la identificará en nuestra aplicación.
- **Title:** Será lo que aparezca en ella cuando ejecutemos el programa.
- **Icon Name y Cursor Name:** Tendremos ocasión de escoger un tipo de icono para mostrar en la barra de título, puede ser uno de los que trae por defecto o cualquiera que nosotros hayamos creado o vinculado. Y el cursor igualmente.
- **Window, Frame, Style:** Con ellas podremos variar la apariencia visual de nuestra ventana. Lo mas recomendable es que probéis y os quedéis con la que mas os guste.

Con la ventana de Status, obtenemos información acerca de la posición relativa de nuestra ventana con respecto a la pantalla y el tamaño de anchura y altura que tiene.

Así mismo, con la ventana de color podemos modificar el color de fondo y de texto que tendrá.

En cuanto a los eventos que tiene un objeto de tipo "sheet" o ventana, son los siguientes:

- **SPECIAL-NAME:** Actuará igual que en un programa normal, es decir aquí definiremos lo mismo que haríamos en un programa normal en nuestra CONFIGURATION SECTION. También es posible definir tipos de impresión y otras posibilidades que nos ofrece el propio compilador de Fujitsu.
- **FILE-CONTROL:** Definición de archivos que vamos a utilizar en nuestra ventana Exactamente igual que se explican en los manuales con la excepción de que no hay que especificar el tipo de dispositivo excepto en la definición de la impresora que si se pondrá. Además aquí es donde si queremos especificar que vamos a trabajar con un fichero Btrieve (RM) se le indica con las letras BTRV, después del ASSIGN TO. Un ejemplo sería el siguiente:

```
SELECT SOCIOS ASSIGN TO "SOCIOS.DAT" BTRV  
ORGANIZATION INDEXES ACCESS DYNAMIC  
RECORD KEY KEYSOC  
ALTERNATE RECORD KEY KEYSOC1  
FILE STATUS STASOC.  
  
SELECT IMPRE ASSIGN TO PRINTER.
```

- **BASED:** Es algo propio del Fujitsu y no se para que sirve, además en lo que he podido leer no he conseguido nada, puede ser que sea por compatibilidad con sus productos anteriores, de todas formas no lo he necesitado para realizar ninguna aplicación.

- **FILE:** Aquí será donde se definan las descripciones de los archivos que vayamos a utilizar. Quiero hacer incapié en una cosa. Tened siempre en cuenta que cuando trabajemos con Power, las variables serán por defecto locales y por lo tanto no se extenderán al resto de ventanas que utilice nuestra aplicación, para ello es necesario utilizar la opción GLOBAL y EXTERNAL. De tal modo que una FD quedaría:

```
FD SOCIOS GLOBAL EXTERNAL LABEL RECORD STANDARD.
01 REGSOC.
  02 KEYSOC.
  03 .....
```

- **WORKING:** Que os voy a contar, aquí definiremos las variables a utilizar en el programa. Recordad lo que os he dicho antes, podéis utilizar GLOBAL y EXTERNAL si lo creéis conveniente, para usarlas en otras ventanas de la aplicación. Además quiero añadir que por ejemplo, si luego algún componente determinado va a utilizar alguna variable exclusiva la podremos definir para ese componente y no para el resto de programa. Recordad que la ventana sobre la que estamos hablando es otro componente de nuestro programa para POWER. Eso significa también que si definimos una variable como: 01 SALUDO PIC X(30). Esa variable no la podremos utilizar en un campo de nuestra ventana, ya que al no definirla como GLOBAL, le indicamos al compilador que solo se utilizará en la ventana como componente, pero no en el resto del programa, por lo cual lo mas razonable es definirlas todas como GLOBAL. El ponerle EXTERNAL, hará la misma función que conseguimos con la LINKAGE SECTION.
- **CONSTANT:** Para definir constantes, pero se pueden definir igualmente en la Working como siempre hemos hecho con VALUE.
- **PROCEDURE:** ¡¡¡ OJO !!! Atención, aquí se van a definir las rutinas que luego podremos llamar desde nuestro programa, en ningún caso, lo que aquí se ponga, se ejecutará por si solo. Yo me tiré dos días preguntándome porque no se ejecutaban las sentencias que ponía aquí. Imaginaros las rutinas que hacéis para luego llamarlas con CALL, pues bien eso mismo es ésto,, hasta hay que definirlas completamente es decir con IDENTIFICATION, PROGRAM-ID, etc .. Un ejemplo:

```
IDENTIFICATION DIVISION.
PROGRAM-ID. LIMPIAR IS COMMON.
ENVIRONMENT DIVISION.
DATA DIVISION.
PROCEDURE DIVISION.
INICIO SECTION.
MOVE 0 TO POW-NUMERIC OF CODIGO.
MOVE SPACES TO POW-TEXT OF NOMBRE.
MOVE SPACES TO POW-TEXT OF DIRECCION.
MOVE SPACES TO POW-TEXT OF POBLACION.
EXIT PROGRAM.
END PROGRAM LIMPIAR.
```

Si veis la estructura es similar a un pequeño programa. Para luego llamarlo desde cualquier parte, simplemente CALL "LIMPIAR" en este caso porque ese es su nombre. En este caso lo que hará será mover a espacios y a ceros el contenido de los controles que se indican (CODIGO, NOMBRE, DIRECCION y POBLACION).

- **OPENED:** Aquí si pondremos realmente lo que queramos que el programa realice justo al mostrar la ventana, por ejemplo, rellenar los combobox o

grid que tengamos, o abrir fichero o cualquier acción previa a la visualización de la ventana. Podremos tener aquí nuestra propia Environment o Data Division si fuese necesario.

- **CLOSE:** Aquí se darán las ordenes que el programa ejecute, justo al cerrar la ventana, lo mas normal será cerrar los ficheros, pero podremos dar cualquier comando.
- **CLOSECHILD:** En este apartado, programaremos las acciones que el compilador ejecutará siempre que cerremos una ventana que haya sido abierta desde ésta. Es decir si desde ésta ventana principal llamamos a otra. Justo al cerrarse esta nueva y antes de pasar el control a la principal se ejecutará este apartado.

Una vez explicado esto podemos empezar a colocar los objetos que deseemos en nuestra ventana. Recordad todo bien, en la PROCEDURE irán las rutinas que queramos llamar luego desde el programa, en OPENED, irá lo que deseemos que el programa realice antes de mostrar la ventana. Y en toda la DATA, cualquier dato que queramos que sea portable a cualquiera de los componentes del programa, lo declararemos con GLOBAL y si además queremos que sea portable a otras ventanas, además de GLOBAL, le pondremos EXTERNAL. Esto es algo muy importante.

Tenéis que tener en cuenta que cada ventana es un programa independiente (por llamarlo de alguna forma) es decir que los datos y variables si no las definís con EXTERNAL, no se corresponderán. En Acucobol o RM, siempre que abrimos una ventana desde un programa, éste sigue teniendo el control y no hay porque definir nada nuevo, aquí NO, aquí cada ventana es un programa.

Solo los mensajes que podremos displayar en ventanas pequeñas, del tipo (SI / NO), (ACEPTAR / CANCELAR) formarán parte de nuestro programa o ventana.

Una vez creada la ventana, lo siguiente que debemos de hacer es incorporarla a nuestro Proyecto y estará lista para ser ejecutada.

En el siguiente curso, voy a explicar como se llaman las variables que hacen referencia a las propiedades de los controles mas usuales, para después realizar un pequeño programa y ver como funciona.

CAPTITULO 3. OBJETOS O CONTROLES.

Objeto: Será todo aquello que definamos gráficamente en nuestro programa, serán objetos un Push-Button, un Label, un Combobox, una Tabla, una Imagen, etc Cada objeto deberá de tener un nombre como si de una variable se tratara. En principio el programa asignará un nombre por defecto pero nosotros podremos variarlo y darle el que mas nos apetezca, ese nombre será el que utilizemos para todo lo que haga referencia a ese objeto. Las pantallas que diseñemos serán los contenedores de objetos, es decir donde definamos todos los objetos que deseemos en nuestra aplicación. Pero a su vez también será un objeto con sus propiedades, sus métodos y sus eventos. En cada aplicación puede haber mas de una pantalla, pero una será siempre la principal.

Veamos una lista de los objetos que nos presenta PowerCobol, en ésta versión 3 y una pequeña explicación de para que sirven.

Al final de la página explicaré para que sirven las diferentes opciones de estilo que tienen los objetos, para casi todos son las mismas. Y cuando actúa cada evento

OBJETO	DESCRIPCION	EVENTOS
	LABEL: Un Label, es una etiqueta, un comentario o cualquier cosa que queramos poner en nuestro programa en texto. Sustituye a los DISPLAY (entre comillados) de nuestros programas.	- CLICK - CHANGE
	EDIT: Un Edit nos sirve para aceptar cualquier tipo de campo. Sustituye a los ACCEPT de nuestro programa.	- CHANGE - EDIT - RETURN
	PICTURE EDIT: Un Picture Edit, es igual a un Edit, pero con la ventaja de poder aceptar el campo con la edición que le definamos mediante un PIC. Como desventaja, perdemos el evento CHANGE.	- RETURN - EDIT
	PUSH BUTTON: Un Push Button, es un botón que se puede pinchar para ejecutar cualquier cosa. No se podría decir exactamente a que sustituye en nuestros programas, pero casi siempre a las preguntas de aceptación.	- CLICK
	CHECK BUTTON: Un Check Button, nos sirve para escoger o no una opción. Puede sustituir a las típicas preguntas de Si o No, sobre algunas opciones de nuestros programas.	- CLICK
	RADIO BUTTON: Un Radio Button suele ir acompañado de otros y nos sirve para escoger entre varias opciones, una posible. Siempre la usaremos cuando las opciones estén claramente delimitadas.	- CLICK
	GROUP BOX: Un Group Box, es un rectángulo que puede contener un título. Nos sirve para agrupar controles que mas o menos tienen algo que ver entre si. Por ejemplo un grupo de Radio Button.	
	LIST BOX: Un List Box, es una lista desplegable con opciones de entre las cuales podemos escoger una. Por ejemplo, para escoger una de las provincias del país.	- DBLCLICK - SELCHANGE
	COMBO BOX: Un Combo Box, es una mezcla de Edit y List Box. Aunque se puede presentar de varias formas. Yo la he utilizado mas que el List Box.	- CHANGE - SELCHANGE - EDIT - RETURN
	HORIZONTAL SCROLL BAR: Una Barra de Scroll horizontal. La podemos utilizar para presentar el	- CHANGE - ENDSCROLL

	progreso de algo que estemos realizando, siempre que sepamos los valores de inicio y fin.	
	VERTICAL SCROLL BAR: Igual que la anterior, solo que la presentación es vertical, en vez de horizontal.	- CHANGE - ENDSCROLL
	TIMER: Un control para asignar espacios de tiempo programables.	- TIMER
	TABLE: Magnifico control con el cual creamos una tabla dimensionable. Las barras de desplazamiento se ponen automaticamente dependiendo del tamaño que le hayamos dado.	- CLICK - DBLCLICK - EDIT - RETURN
	GRAPH: Nos permite presentar un gráfico de barras sobre unos valores dados previamente. En este caso es de una dimension unicamente.	
	DDE: Dyanmic Data Exchange, para comunicaciones con otras aplicaciones.	- DDECHANGE
	SOUND: Permite incluir un fichero de sonido con formato WAV.	
	FUNCTION KEY: Igual que un Push Button, pero a la que se le puede asignar una tecla de función.	- CLICK
	DATE: Un control que nos sirve para poner la fecha actual en diferentes formatos.	- CLICK - CHANGE
	METAFILE: Podemos colocar archivos de tipo .emf, .wmf, .clp. No he podido comprobar su uso.	- CLICK - CHANGE
	SIMPLE ANIMATION: Podemos hacer una animacion, indicando varios archivos graficos tipo BMP e indicando el intervalo entre cada uno.	- CLICK - DBCLICK - STARTANIME - ENDANIME
	MCI: Control utilizado para exponer videos AVI, o poder hacer un reproductor de CD-Audio.	
	RECTANGLE: Para dibujar un rectangulo en nuestras pantallas. Puede tener su sentido para dar vistosidad o crearnos barras de herramientas.	
	DRIVE LIST: Control para mostrarnos y utilizar las unidades que tengamos en nuestro disco.	- SELCHANGE
	DIRECTORY LIST: Para mostrarnos los directorios o carpetas de una determinada unidad.	- CHANGE - SELCHANGE
	FILE LIST: Para mostrar los archivos de un directorio seleccionado, podemos escoger entre sus atributos.	- DBCLICK - SELCHANGE

	BITMAP BUTTON: Un Push Button, al cual podemos incorporar una imagen para mayor vistosidad.	- CLICK
	SELECTION BOX: Es igual que un ComboBox, en el que las opciones se cargan directamente desde un archivo.	- SELCHANGE - EDIT - RETURN
	EXCEL CONNECTION: Permite enlazar y obtener datos de una hoja de calculo en Excel. En esta versión falla, porque la familia Office, a la que pertenece Excel se instala en el directorio Archivos de Programa, y ya sabeis los problemas de esta version con los directorios que contienen espacios en blanco.	
	PRINT: Control con el que podemos imprimir los campos de la pantalla actual que tengan activada la casilla de printable. Muy bueno. Digamos que es como un Imprimir Pantalla bastante mejorado.	
	EXTEND IMAGE: Igual que el Image, pero que admite mas versiones de ficheros de imagenes.	- CLICK - DBCLICK - CHANGE
	DB ACCESS: Control que nos permite enlazar con una base de datos y manejar las tablas en nuestro programa de Cobol.	

Ya sabéis que pulsando sobre el botón derecho del ratón y siempre que estemos sobre un objeto, nos aparecerá un menú con el cual nos podremos dirigir tanto al estilo del objeto como a las procedures de cada evento.

Los campos que tenemos dentro de Style, son casi iguales en todos los casos, voy a explicar los mas comunes:

- **Item Name:** Indica el nombre con el que el programa se referirá a dicho componente, ese nombre aunque sale por defecto, podemos modificarlo a nuestro antojo.
- **Text:** Hace referencia al texto que saldrá por defecto puesto en el componente.
- **Visible:** Indica si el objeto será visible en el momento de la ejecución.
- **Enable:** Indica si el objeto estará disponible en el momento de la ejecución.
- **3-D:** Indica si el objeto tendrá una presentación en 3-D, es decir con bordes resaltados y creando una sensación de efecto de 3 dimensiones.
- **Border:** Indica si el campo tendrá un borde rodeándole cuando aparezca.
- **Tab Stop:** Indica si cuando pulsemos Tab, el cursor se posicionará en alguna ocasión en este objeto o por el contrario nunca lo hará.
- **Print:** Indica si este objeto será impreso cuando se seleccione la orden de imprimir con el objeto Print.

- Además encontraremos en la mayoría la alineación tanto vertical como horizontal del componente.

Por supuesto cada uno de estos estilos son modificables en tiempo de ejecución, lo que le definamos aquí será para la primera vez que aparezcan, porque de alguna manera tienen que aparecer, pero nosotros podemos modificar tantas veces como queramos. En el siguiente capítulo explicaré como se hace referencia a los estilos y como se actúa con ellos.

Los eventos, serán las acciones que se produzcan cuando se actúe sobre alguno de ellos, por ejemplo el evento de un Push-Button, será el Click, es decir, cuando hagamos click con el ratón sobre el. Al programar un evento, lo que le decimos al programa es lo que tiene que hacer cuando se produzca dicho evento.

- **CLICK:** Este evento se produce al pinchar con el botón izquierdo del ratón sobre un objeto.
- **DBCLICK:** Este evento se produce al hacer doble-click con el botón izquierdo del ratón sobre un objeto.
- **CHANGE:** Este evento se produce cuando cambia el contenido de un objeto.
- **SELCHANGE:** Este evento se produce cuando cambia la selección de un objeto de tipo lista, es decir cuando cambiamos en un ListBox, o en un Combobox.
- **EDIT:** Se produce cuando se entra en edición en un campo de recogida de datos, Edit, Picture-Edit, Table, etc...
- **RETURN:** Se produce cuando pulsamos Return sobre un objeto. Hay que tener en cuenta que en muchos de los objetos podemos definir que otra acción puede hacer que se produzca el evento RETURN, por ejemplo al perder el foco.

Respecto a versiones posteriores (yo ahora tengo la 5), hay que hacer notar que los eventos aumentan en número considerable y que los objetos son mas y algunos se han integrado con otros.

CAPTITULO 4. PROPIEDADES.

Propiedad: Cada una de las opciones que puede tener un objeto y pueden ser comunes o distintas según el tipo de objeto. Serán propiedades, el color, la altura, la anchura, el título, el tipo de letra, si está o no disponible, si está o no visible, etc ... Las propiedades suelen tener un nombre pre-definido por el lenguaje que lo haya designado. Muchas de las propiedades tendrán un valor de tipo SI-NO.

Las propiedades se ajustan normalmente en tiempo de diseño, pero por muchos motivos será necesario cambiarlas también en tiempo de ejecución, para cambiar estas propiedades utilizaremos nuestro comando de COBOL, MOVE. De tal manera que lo haremos igual que cuando le damos un valor a una variable.

El formato para referirnos a las propiedades de un objeto será el siguiente:

MOVE *valor* **TO** *propiedad* **OF** *control*.

En el caso de PowerCobol, los nombres de las propiedades están predefinidas y todas empiezan por la palabra POW- seguida del nombre de la propiedad. En las propiedades de tipo SI-NO, podemos utilizar para el "NO" el valor "0" o también la palabra reservada de PowerCobol, POW-OFF, para el caso del "SI" podremos utilizar "1" o también POW-ON.

A continuación voy a explicar y poner algunos ejemplos de las mas utilizadas.

PROPIEDAD	DESCRIPCION	VALORES
POW-ACTIVATE	Solo se aplica al objeto timer y sirve para iniciarlo o pararlo. MOVE POW-ON TO POW-ACTIVATE OF RELOJ.	- POW-ON - POW-OFF
POW-BACKCOLOR	Para indicar el color de fondo de un objeto, aplicable a la mayoría de ellos. Podemos utilizar POW-RED (rojo), POW-BLUE (azul), etc.. También para colorear celdas de una tabla. MOVE POW-RED TO POW-BACKCOLOR OF NOMBRE. MOVE POW-BLUE TO POW-BACKCOLOR (1,1) OF TABLA.	- POW-RED - POW-BLUE - POW-BLACK - etc
POW-BORDER	Para que aparezca el objeto con borde, Label, Edit, Image, y algún otro. MOVE POW-ON TO POW-BORDERE OF CAMPO.	- POW-ON - POW-OFF
POW-CHECK	En los objetos RadioButton, CheckButton y Menu, indica si esta o no chequeado. MOVE POW-ON TO POW-CHECK OF ACTIVO.	- POW-ON - POW-OFF
POW-COLS	Contiene el número de columnas de una tabla. MOVE POW-COLS OF TABLA TO COLUMNAS.	- Valor numérico. (devuelve valor)
POW-COUNT	En los ListBox, ComboBox y demás objetos con listas, nos devuelve el número de elementos que tiene. MOVE POW-COUNT OF LISTA1 TO ELEMENTOS.	- Valor numérico. (devuelve valor)
POW-DATA	En los Graph, el valor de cada una de los elementos del gráfico. MOVE 30 TO POW-DATA (3) OF GRAFICO.	- Valor numérico.
POW-DATACOLOR	En los Graph, el color de cada una de los elementos del gráfico. MOVE POW-RED TO POW-DATACOLOR (3) OF GRAFICO.	- POW-RED (resto de colores)
POW-ENABLE	En todos los objetos, para ponerlos en activos o pasivos. MOVE POW-ON TO POW-ENABLE OF GRAFICO.	- POW-ON - POW-OFF
POW-NUMERIC	Introduce un valor numérico en los objetos PictureEdit y Table. MOVE 14 TO POW-NUMERIC(3,3) OF TABLA.	- Valor numérico.
POW-PRNENABLE	En todos los objetos, indican si van a ser imprimibles con el objeto Print. MOVE 1 TO POW-PRNENABLE OF NOMBRE.	- POW-ON - POW-OFF
POW-ROW	Nos devuelve el número de línea que tiene el foco en una tabla. MOVE POW-ROW OF TABLA TO LINEA.	- Valor numérico. (devuelve valor)
POW-ROWS	Le indicamos el número de líneas que tiene una tabla. MOVE 130 TO POW-ROWS OF TABLA.	- Valor numérico.
POW-SELECT	En los objetos de listas ListBox, ComboBox, nos indica el elemento que está seleccionado. MOVE POW-SELECT OF LISTA TO ELEMENTO. MOVE 1 TO POW-SELECT OF LISTA.	- Valor numérico.
POW-TEXT	En muchos controles, nos indica el título o el contenido del mismo. MOVE "TITULO" OF POW-TEXT OF ETIQUETA.	- POW-RED (resto de colores)
POW-TEXTCOLOR	En muchos controles, nos indica el color del texto. MOVE POW-GREEN TO POW-TEXTCOLOR OF ETIQUETA.	- Valor alfanumérico.
POW-VISIBLE	En casi todos los objetos, nos sirve para indicar si están o no visibles en nuestras ventanas. MOVE 1 TO POW-VISIBLE OF GRAFICO.	- POW-ON - POW-OFF

Existen muchas mas propiedades, pero se irán viendo en el ejemplo que empezaremos a desarrollar. Este capítulo está mas enfocado hacia el modo de introducción de datos en las propiedades que a la explicación de todas ellas.

Con estas propiedades además de hacer el MOVE, se puede programar, es decir, por ejemplo podemos preguntar por ellas:

IF POW-SELECT OF LISTA = 1 MOVE "HA SELECCIONADO LA OPCION 1" TO POW-TEXT OF ETIQUETA.
IF POW-SELECT OF LISTA = 2 MOVE "HA SELECCIONADO LA OPCION " TO POW-TEXT OF ETIQUETA.

Del mismo modo que también podemos coger los datos de esas propiedades para en nuestro programa hacer con ellos lo que deseemos. Por ejemplo:

MOVE POW-TEXT OF ETIQUETA TO NOMBRE.

Con estos ejemplos simples, tendríamos que al final la variable NOMBRE definida en nuestra WORKING, como alfanumérica tendría el valor dependiendo de la opción seleccionada en el ListBox llamado LISTA. Y también el objeto Label llamado Etiqueta contendría ese mismo valor.

El próximo capítulo estará dedicado a la utilización de los métodos y luego empezaré con el desarrollo de un programa completo, igual que se hizo en la sección programando, es decir, tendremos la misma agenda, pero ahora en formato gráfico de Windows.

CAPTITULO 5. METODOS.

Método: Son procedimientos que ya vienen programados por el lenguaje y nosotros solo tendremos que llamarlos para que actúen. Son métodos, añadir campos a un Combobox, enviar el foco a cualquier objeto, abrir una ventana, cerrarla, etc ... Como veremos cada uno de los objetos puede tener sus propios métodos.

Para acceder a todos los métodos se utilizará la sentencia CALL, quedando su formato de la siguiente manera:

CALL método OF objeto USING parámetros.

A continuación vamos a explicar algunos de los métodos mas habituales. Pero como siempre os digo, en los ejemplos prácticos es donde mejor se comprende todo.

METODO	DESCRIPCION
ADDSTRING ADDSTRING256	Para añadir elementos a objetos ListBox y ComboBox. CALL ADDSTRING OF LISTA USING "PRIMER ELEMENTO".
ALARM	Aplicable solo en los Sheet y produce un sonido, podemos escoger entre varios predefinidos.POW-MBOK, POW-MBASTERISK, POW-MBQUESTION, POW-MBEXCLAMATION, POW-MBHAND. CALL ALARM OF SHEET1 USING POW-MBOK.
CLEARLIST	Aplicable a ListBox y ComboBox, conseguimos reiniciar los objetos de tal manera que quedan vacios completamente. CALL CLEARLIST OF LISTA.
OPENPRINTER WHITESHEET CLOSEPRINTER	Solo para el objeto Print. Abrimos y cerramos el objeto Print, para provocar una salida por impresora de los objetos de nuestra pantalla que tengan la opción PrmEnable activada. CALL OPENPRINTER OF IMPRESORA. CALL WRITESHEET OF VENTANA1. CALL CLOSEPRINTER OF IMPRESORA.
CLOSESHEET	Aplicable solo a las ventanas, con ello las cerramos. CALL CLOSESHEET OF VENTANA1.
DELETESTRING	Aplicable a los controles ListBox y ComboBox, y conseguimos borrar un elemento de la lista. CALL DELETESTRING OF LISTA USING 1.
DISPLAYMESSAGE	Se va a explicar en la parte final. Por ser un método muy utilizado y util.
GETCELLNUMERIC GETCELLTEXT	Aplicable a las tablas y conseguimos extraer el contenido de las celdas, ya sean numéricas o alfanuméricas. En el ejemplo moveremos a la variable VALOR, el contenido de la celda situada en la columna 1 línea 1 de la tabla llamada TABLA. CALL GETCELLTEXT OF TABLA USING VALOR 1 1.
OPENSHEET	Aplicable solo a las Sheet y con ella las abrimos. Las ventanas siempre dependen de una padre. CALL OPENSHEET OF PRINCIPAL USING "SEGUNDA".
SETCELLNUMERIC SETCELLTEXT	Aplicable a las tablas y conseguimos introducir un valor en las celdas, ya sean numéricas o alfanuméricas. En el ejemplo moveremos el valor "HOLA" a la celda situada en la columna 1 línea 1 de la tabla llamada TABLA.

	CALL SETCELLTEXT OF TABLA USING "HOLA" 1 1.
SETFOCUS	Aplicable a la mayoría de los controles. Con el conseguimos pasar el foco de un objeto a otro. CALL SETFOCUS OF LISTA.

Existen muchos mas métodos, pero estos son los mas utilizados. Veamos como un ejemplo sencillo como rellenaríamos un ListBox a partir de una tabla definida en la Working. Siendo LISTA l nombre que tiene nuestro ComboBox o ListBox. Y además tenemos una tabla con dos columnas, en la primera introducimos el número y en la segunda su nombre.

WORKING-STORAGE SECTION.

01 TABLA.

02 FILLER PIC X(10) VALUE "PRIMERO".

02 FILLER PIC X(10) VALUE "SEGUNDO".

02 FILLER PIC X(10) VALUE "TERCERO".

02 FILLER PIC X(10) VALUE "CUARTO".

02 FILLER PIC X(10) VALUE "QUINTO".

01 TABLITA REDEFINES TABLA.

02 ELEMEN PIC X(10) OCCURS 5 TIMES.

01 CONTA PIC 9.

01 CAMPO PIC X(10).

...

PROCEDURE DIVISION.

INICIO.

MOVE 0 TO CONTA.

UNO.

ADD 1 TO CONTA IF CONTA > 5 GO DOS.

* aqui cargamos el ComboBox.

CALL ADDSTRING OF LISTA USING ELEMEN (CONTA).

* aqui cargamos la tabla.

CALL SETCELLNUMERIC OF TABLA USING CONTA CONTA 1.

MOVE ELEMEN (CONTA) TO CAMPO.

* en la version 3 de PowerCobol da error si introducimos un campo

* con subindice directamente en una tabla, por eso primero lo

* muevo a un campo.

CALL SETCELLTEXT OF TABLA USING CAMPO CONTA 2.

GO UNO.

DOS.

EXIT.

Después de ejecutar el programa:

el ComboBox nos hubiera quedado asi:

Y la tabla hubiera quedado asi:

1	PRIMERO
2	SEGUNDO
3	TERCERO
4	CUARTO
5	QUINTO

DISPLAYMESSAGE

Con este método de PowerCobol, conseguimos que se nos muestre una ventana independiente pero sin perder el control sobre la que tenemos activa. Son las típicas ventanas de confirmación de Windows y tienen unos parámetros pre-

asignados. A continuación vamos a ver algunos ejemplos y como quedan en la práctica.

Su formato, como cualquier otro método es el siguiente:

CALL DISPLAYMESSAGE OF nombreventana USING texto título estilo.

- **Texto:** el texto hace referencia al texto que nos saldrá al mostrar el mensaje en la ventana.
- **Título:** el título hace referencia al título que tendrá la venta que nos saldrá.
- **Estilo:** El estilo se refiere al icono que tendrá la ventana y a las posibles opciones que se nos presenten, en cuanto a los ICONOS:
 - POW-DMNOICON, sin icono.
 - POW-DMICONSTOP, icono con el símbolo Stop.
 - POW-DMICONQUESTION, icono de interrogación.
 - POW-DMICONEXCLAMATION, icono con el signo de exclamación.
 - POW-DMICONINFORMATION, icono con una (i).
- **Estilo:** En cuanto a los botones:
 - POW-DMOK, solo el botón de Ok.
 - POW-DMOKCANCEL, botones de Ok y Cancelar.
 - POW-DMABORTRETRYIGNORE, botones de Abortar, Reintentar e Ignorar.
 - POW-DMYESNOCANCEL, botones de Si, No y Cancelar.
 - POW-DMYESNO, botones de Si y No.
 - POW-DMRETRYCANCEL, botones de Reintentar y Cancelar.
- **Respuestas:** las posibles respuestas al pulsar los distintos botones.
 - POW-DMROK, si hemos pulsado el boton de Ok.
 - POW-DMRCANCEL, si pulsamos Cancelar.
 - POW-DMRABORT, si pulsamos Abortar.
 - POW-DMRRETRY, si pulsamos Reintentar.
 - POW-DMRIGNORE, si pulsamos ignorar.
 - POW-DMRYES, si pulsamos Si.
 - POW-DMRNO, si pulsamos No.

El estilo se debe de guardar en una variable con el formato: 01 ESTILO PIC S9(4) COMP-5. Y para introducir los valores que deseemos lo haremos de la siguiente manera:

ADD POW-DMYESNO POW-DMICONQUESTION GIVING ESTILO.

De esta manera asignamos al estilo el icono de Interrogación y los botones de Si y No, quedando nuestra orden completa de la siguiente manera, teniendo la ventana donde saldrá el nombre de VENTANA1:

WORKING-STORAGE SECTION.

01 ESTILO PIC S9(4) COMP-5.

01 TITULO PIC X(20).

01 TEXTO PIC X(40).

...

PROCEDURE DIVISION.

INICIO.

ADD POW-DMYESNO POW-DMICONQUESTION GIVING ESTILO.

MOVE "Mensaje" TO TITULO.

MOVE "Salir de la aplicación ?" TO TEXTO.



CALL DISPLAYMESSAGE OF VENTANA1 USING TITULO TEXTO ESTILO.

....

Ahora nos saldrá la correspondiente ventana con el mensaje y entonces puede que le demos al botón del Si o al del No, para controlarlo utilizamos la siguiente instrucción:

```
IF PROGRAM-STATUS = POW-DMRYES
  MOVE "HAS PULSADO SI" TO POW-TEXT
  OF TEXTODEEXPLICACION.
IF PROGRAM-STATUS = POW-DMRNO
  MOVE "HAS PULSADO NO" TO POW-TEXT
  OF TEXTODEEXPLICACION.
```

CAPTITULO 6. COMIENZA LA APLICACION.

INTRODUCCION.

Ya tenemos creado nuestro directorio llamado Agenda, dentro de FSC. Esta versión de Power, no nos permite darle con un número el tamaño de la ventana, sino que lo conseguimos con el ratón agrandando o disminuyendo como cualquier ventana de Windows. Os digo esto porque para conseguir el tamaño de las ventanas que os habéis bajado en el ejecutable, tendréis que hacerlo como podáis.

Las palabras que os encontréis en rojo, harán referencia a puntos del menú de Power.

Lo primero que hacemos es crearnos un proyecto: **Project-New** lo guardáis en c:\fsc\agenda\agenda.prj.

Si no os habéis bajado todos los iconos que vamos a utilizar en la aplicación, este es un buen momento para hacerlo. Podéis encontrar el enlace al final de la pantalla, debéis descomprimir el archivo totico.zip en la misma carpeta.

Ahora creamos nuestra primera ventana: **File-New**, una vez abierta le dáis al botón derecho del ratón sobre ella, pinchamos sobre **Style** y ponemos lo siguiente:

- **Sheet name:** PRIMERA
- **Title:** Agenda v1.0
- **Icon name y Cursor name:** Por ahora no se tocan
- **Window:** Popup Window
- **Frame:** Thin
- **Style:** Title, Minimize Box y Control Menú
- **3D:** sin marcar

Pulsamos Ok y como color de fondo seleccionamos el que mas nos guste, si queréis respetar el que tiene, lo agregáis a la paleta de colores es: 198,198,255.

Una vez realizado hacemos: **File-Save as**, y le damos el nombre de agenda.win. A continuación nos vamos a **Project-Edit**, pinchamos sobre **Add** y añadimos la pantalla al proyecto, como va a ser la pantalla con la que arranque la aplicación marcamos la casilla, Starting Sheet.

PRIMERA COMPILACION.

Ahora ya, podríamos por ejemplo, compilar y ejecutar, aunque obviamente no nos iba a hacer nada, pero Power ya reconoce un proyecto completo. Podéis hacer una prueba, le dáis **Project - Build** y luego **Project - Run**.

A partir de ahora para ejecutar bastará solo con dar a **Run**, si ha habido alguna modificación se compilará automáticamente.

Tenéis que tener en cuenta que **Project-Compile**, actuará sobre la ventana activa, mientras que **Project-Build**, compilará todas las ventanas del proyecto.

Project-Link y **Project-Make**, se encargan de generar el ejecutable para su ejecución. Para un proyecto pequeño podemos hacer siempre **Build**, pero si es muy largo, para depurar errores iremos compilando ventana a ventana.

Quizás haya sido un poco lioso lo que he explicado o quizás algo demasiado simple que ya todos conocéis, pero creo que siempre es bueno refrescar las ideas y dar la oportunidad de ayudar al que empieza desde la nada.

LAS IMAGENES.

Cuando utilizamos imagenes en PowerCobol, podemos usarlas como imagenes independientes, o bien como un recurso propio de la aplicación, me explico: Una imagen puede ir con la aplicación en un archivo mas y llamarla así en nuestra aplicación o bien que las imagenes se integren en el ejecutable de la aplicación y no vayan como archivos independientes. La diferencia estará en el tamaño del ejecutable, pero en cambio, nadie podrá modificar las imagenes que incluyamos. En la aplicación utilizaremos ambas para que veáis como se implementan.

Vamos a colocar el icono para la aplicación siguiendo los siguientes pasos:

- **Project-Edit, Add** , en tipo de archivo seleccionamos Icon (*.ICO) y nos aparecerá el que he creado para ello: Agenda.ico.
- En la casilla Resource name, le ponemos AGENDA. Que será el nombre con el que Power reconocerá dicho icono.
- A continuación nos vamos al Style de nuestra ventana principal y en la casilla Icon Name, ponemos AGENDA. Cuando compilemos y ejecutemos de nuevo ya tendrá nuestra aplicación dicho icono y si miráis con el Explorador de Windows en la carpeta veremos que el archivo AGENDA.EXE, también se identifica con él.

Ahora vamos a colocar la primera imagen en nuestra ventana, para ellos seleccionamos el icono Extend Image de la ventana de controles y lo arrastramos hasta nuestra ventana.

Una vez colocado el icono de la imagen en la ventana, nos vamos a Style y en Image File, ponemos bloc.bmp. Podríamos utilizar también el botón Image File para localizarlo, pero eso nos daría el path completo de donde se encuentra, con lo que si luego cambiamos la aplicación de carpeta, la imagen no saldría, por eso es mejor, siempre que esté en la misma carpeta del proyecto, poner solo su nombre.

Además deshabilitaremos la casilla de Border, para que se quede perfectamente enclavado en nuestra pantalla, ahora solo nos queda ir probando para asignar el

tamaño preciso, tanto a la ventana como a la imagen, en cualquier momento podremos pulsar el botón de Test y comprobar como va quedando.

CREANDO EL MENU.

Abrimos la ventana de menubar y en title colocamos lo que se verá en la barra de menú: &Opciones.

Pulsamos Enter y ahora escribimos &Consulta y le damos al botón que tiene los signos => para implementarlo dentro de Opciones.

Pulsamos sobre el botón de Extend, para que nos muestre mas opciones para el menú y pinchamos donde indica Separator: Ins Under, que significa que vamos a introducir un separador del menú debajo de la opción de Consultas.

Ahora pinchamos sobre Ins Under en la ventana de la izquierda para introducir otra opción en el menú a la que llamamos &Acerca de ...

A continuación volvemos a introducir un separador igual que lo hemos hecho antes y para concluir volvemos a pulsar sobre Ins Under en la ventana de la izquierda para introducir otra opción, en esta caso la última a la que vamos a llamar &Salir.

A continuación cerramos la ventana de MenuBar y ya podemos darle al test para comprobar como nos ha quedado el menú.



CAPTITULO 7. DISEÑANDO LA PANTALLA.

CONTROLES

Este es el momento de empezar a colocar los objetos en nuestra pantalla. Esta misión de diseño no es la mas importante, pero si va a ser la mas atractiva para nuestro usuario o cliente, que va a querer que la información se presente de manera clara y ordenada, por eso nada mejor que utilizar la utilidad del Grid, para que los controles se alineen con mayor facilidad en la pantalla.

Los controles que vamos a utilizar en esta pantalla son:

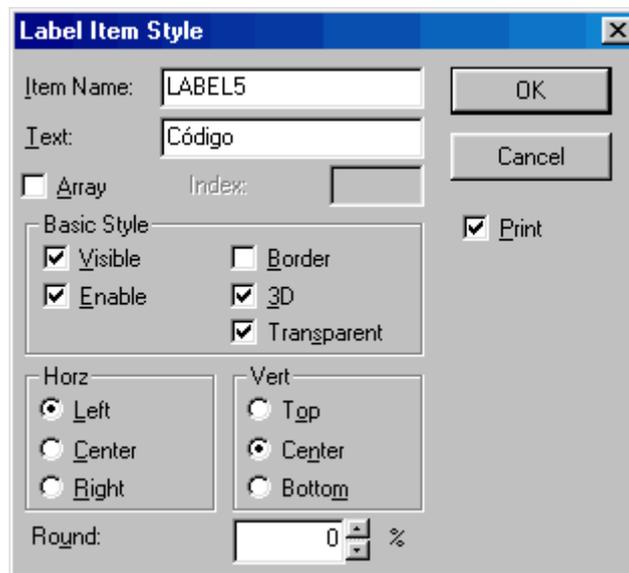
- **Label:** Serán lo que siempre poníamos con **DISPLAY**. Con ellas pondremos todos los nombres de los campos que vamos a introducir, Código, Nombre, Domicilio, Población, etc Podéis ponerlo donde queráis, sin dejaros ninguno o ponerlo como yo lo he hecho en el ejemplo.
- **Group Box:** Son las cajas que facilitan la agrupación de controles, aunque en esta versión solo sirven para eso, para decorar, clarifica aún más la información.

- **Picture Edit:** Son las cajas donde vamos a aceptar los datos, es decir lo que sustituye a los **ACCEPT**. Sin duda lo mas importante y el mas utilizado.
- **Radio Button:** Botones de opción para elegir entre varios, por eso he añadido el campo género.
- **Combo Box:** Caja de opciones, lo he utilizado para señalar el tipo de contacto.
- **Bitmap Button:** Son los botones que harán los procesos.

Una vez los tengáis colocados, es el momento de asignarles un nombre, ese nombre es importante y hará referencia al control. Los nombres si debemos de darlos igual, para que luego en la programación no haya problemas.

LABEL

A los Label con el nombre del campo se le puede dejar el nombre que se le asigne PowerCobol. Mejor veamos esta imagen que corresponde al Label de Código.

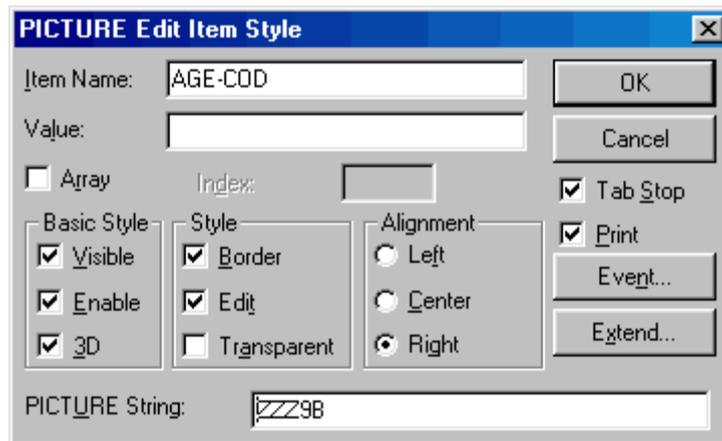


Así es como deben de ir todos los Labels, solo cambiando el Text, que hará referencia a cada uno de nuestros campos.

PICTURE EDIT

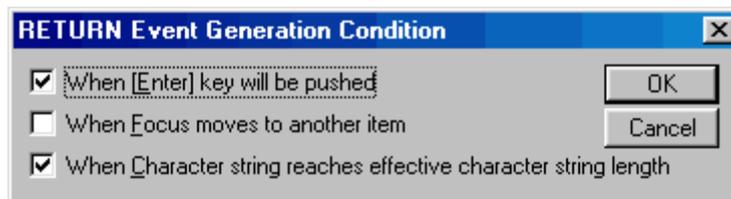
En los Picture Edit, donde introduciremos los campos, su nombre será el mismo que el nombre del campo del registro pero con un guión enmedio.

Por ejemplo el Picture Edit del Código se llamará AGE-COD. Veamos otra imagen.



Aquí fijaros que algo muy importante es PICTURE String, que nos indicará como se mostrará el dato una vez introducido. Por eso he elegido éste control y no el Edit normal. En este caso PIC ZZZ9B.

Además debéis de pulsar sobre el botón Event y asegurarnos que las casillas estén así:



Con ésta ventana le indicamos cuando se efectuará el evento Return, es decir cuando el campo se dará por aceptado. La primera casilla indica que lo hará al pulsar ENTER (activada). La segunda que lo hará cuando el foco esté en otro control (desactivada). La tercera indica que se producirá cuando se rellene todo el campo con su longitud (activada).

Con esto conseguimos que los Picture Edit, funcionen igual que los ACCEPT tradicionales.

A continuación pongo los valores de todos los estilos de los Picture Edit:

Campo	Nombre	Picture String
Código	AGE-COD	ZZZ9B
Nombre	AGE-NOM	X(30)
Domicilio	AGE-DOM	X(30)
Población	AGE-POB	X(20)
Código Postal	AGE-POS	9(5)B
Provincia	AGE-PRO	X(15)
Teléfono	AGE-TEL	X(20)
Teléfono Móvil	AGE-MOV	X(20)
e-mail	AGE-MAI	X(30)
Página Web	AGE-WEB	X(40)

OTROS

Para el Género creamos dos Radio Button, al que identifica al hombre lo llamamos AGE-GEH y el que identifica mujer lo llamamos AGE-GEM. Uno de los dos debe de llevar activa la casilla de Check en su estilo, que nos indica cual estará seleccionado por defecto.

Para el Tipo creamos un ComboBox y lo llamamos AGE-TIP. Tendremos en cuenta de marcar la casilla Dropdown List, dentro del estilo del Combo Box.

Los Group Box, como os dije antes son aclaratorios y solo tienen diseño, así que podéis ponerlos como en el ejemplo agrupando campos mas o menos comunes.

ICONOS

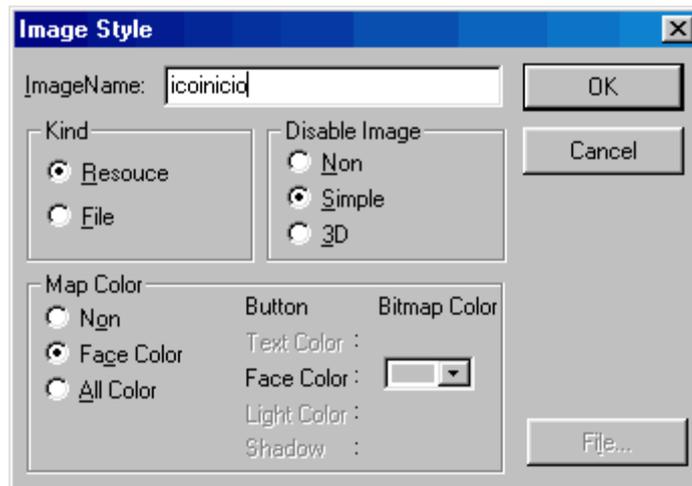
Para simular el hueco donde van los iconos, he dibujado un rectángulo (Rectangle) al que le he dado el color de fondo azul marino. Y luego encima he colocado todos los iconos y label de los iconos.

Vamos a incluir en nuestro proyecto todos los iconos que os habéis bajado y que forman parte de la aplicación. Van a ser recursos y por lo tanto van a formar parte de nuestro proyecto, recordad lo que os comenté en el capítulo anterior sobre como tratar las imágenes.

Por lo tanto abrimos nuestro menú Project, nos vamos a Edit y pulsamos sobre Add. Una vez allí en tipo de archivos marcamos Bitmap (*.bmp) y vamos añadiendo uno a uno. Ya sabéis que cada vez que se añade uno debeis de darle un nombre de recurso. Veamos los que hay que añadir.

Imagen	Nombre recurso
borrar.bmp	icoborrar
consul.bmp	icoconsul
fin.bmp	icofin
grabar.bmp	icograbar
inicio.bmp	icoinicio.bmp
listar.bmp	icolistar
mail	icomail
mas.bmp	icomas
menos.bmp	icomenos
ok.bmp	icook
salir.bmp	icosalir

Ahora es momento para ir colocando los Bitmap Image e ir asignando sus nombres. Primero creamos los cuatro referentes a la búsqueda de registros y los llamamos: INICIO, MENOS, MAS y FIN, asignándoles los recursos apropiados de icoinicio, icomenos, icomas e icofin. Para asignar el nombre del recurso hay que pinchar sobre Image en la caja de estilo de cada uno de los Bitmap Button: Lo haremos igual con todos.



A continuación colocamos el icono de borrar al que llamamos BORRAR. Luego Ok, al que llamamos OK. Después listar, al que llamamos LISTAR. Seguido de e-mail al que llamamos EMAIL. Luego Grabar (GRABAR), consulta (CONSULTA) y salir (SALIR).

Ahora solo nos quedan los nombres de los Botones que se colocan también con label. Los cuatro primeros, los de las flechas de búsqueda, los podemos poner juntos y con el nombre que se le asigne. Pero para los demás lo deberán llevar y será el siguiente, cada uno irá debajo del botón:

texto de Label	Nombre label
Borrar	L-BORRAR
Ok	L-CANCELAR
Listar	L-LISTAR
e-mail	L-EMAIL
Grabar	L-GRABAR
Consulta	L-CONSULTA
Salir	L-SALIR

PRINT

Ahora colocamos el control Print en un lugar de la pantalla que no nos estorbe, este tipo de controles no se ven en tiempo de ejecución, pero es necesario que estén integrados en nuestra ventana, para poder asignarle los valores. Una vez colocado, por ejemplo en la esquina superior derecha nos vamos a su cuadro de estilo y ponemos como nombre PRINT1 y como Title, FICHA DE LA AGENDA. Lo demás lo podéis dejar por defecto.

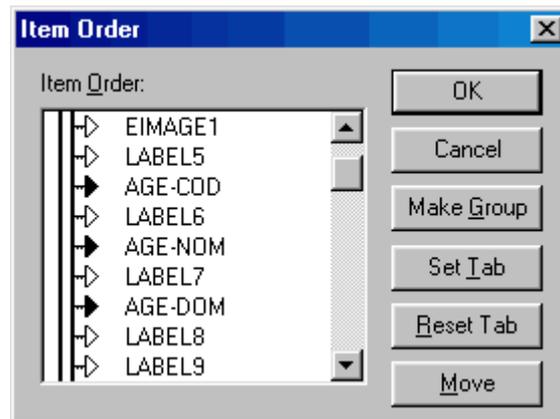
DDE

Vamos a colocar también un DDE, que es un control que nos sirve para conectar con otras aplicaciones externas y que utilizaremos para enviar e-mail. Igual que el anterior no se ve en tiempo de ejecución, pero tenemos que darle propiedades, que serán muy simples. Tenéis que abrir su caja de estilo y simplemente poner en ItemName, DDE1.

ITEM ORDER

Llegados a este punto solo quedar explicar una pequeña cosa. Todos los componentes que hemos ido poniendo llevan un orden y en algunos casos pudiera suceder que unos no se vean, simplemente porque se han quedado debajo, bien, eso se soluciona, sacando el menu pop-up sobre cualquier punto de la pantalla y dirigiendonos a la opción que pone Item Order. Ahí debéis de llevar todo lo que se debe de quedar debajo, es decir los Group Box y el Rectangle a una posición mas alta que lo que va a contener.

También debéis de tener en cuenta en marcar los tabuladores sobre los campos de nuestro fichero y no sobre lo demás, además en el mismo orden en que los hemos colocado. Una muestra de como debe quedar se muestra en ésta imagen.



Los que tienen el triángulo en negro, son los controles sobre los que nos moveremos al pulsar la tecla TAB. Se ponen con Set Tab y se quitar con Reset Tab. Dejad solo los que hacen referencia a los Picture Edit.

CONCLUSION

Llegados aquí os hago una pregunta: ¿Vamos bien ?. Es que no es fácil explicar en la distancia y sobre papel y no quisiera que os perdierais o se quedara algo sin explicar con claridad. Si una vez leído todo lo aquí expuesto, probáis a compilar y ejecutar y algo no funciona, me lo comunicais y lo modifico al instante.

CAPTITULO 8. EVENTOS DE LA VENTANA

EMPEZANDO A PROGRAMAR

Una vez hemos aprendido a diseñar y colocar componentes en nuestras ventanas, ha llegado el momento de empezar a introducir el código fuente necesario para que nuestra aplicación sea algo mas que diseño.

Os vuelvo a recordar que esta programación orientada a eventos, es algo diferente de la tradicional y por lo tanto, habrá parte de código que pongamos en nuestra ventana y parte que irá dentro de los eventos de los componentes.

Para empezar vamos a programar la ventana principal. Pinchamos sobre el botón derecho del ratón sobre cualquier parte de la ventana teniendo en cuenta que no estemos sobre ningún componente. Se nos desplegará un menú con las distintas opciones y nosotros pinchamos sobre Procedure.

A continuación se nos muestra una lista de todos los eventos disponibles para la ventana. Vamos a programar. El primer evento en programar será FILE-CONTROL, así que nos situamos sobre el y pinchamos en OK. Para aclararnos un poco, voy a hacer una tabla con los eventos y su programación:

Evento	Programación	Ayuda
FILE-CONTROL	SELECT AGENDA ASSIGN TO "AGENDA.DAT" ORGANIZATION INDEXED ACCESS DYNAMIC RECORD KEY KEYAGE FILE STATUS STAAGE.	Definición de nuestros archivos.
FILE	FD AGENDA GLOBAL EXTERNAL LABEL RECORD STANDARD. 01 REGAGE. 02 KEYAGE. 03 AGECD PIC 9999. 02 AGENOM PIC X(30). 02 AGEDOM PIC X(30). 02 AGEPOS PIC 99999. 02 AGEPOB PIC X(20). 02 AGEPRO PIC X(15). 02 AGEGEN PIC X. 02 AGETIP PIC 9. 02 AGETEL PIC X(20). 02 AGEMOV PIC X(20). 02 AGEMAI PIC X(30). 02 AGEWEB PIC X(40).	Descripción de los ficheros a utilizar en nuestra aplicación. Al ponerle GLOBAL EXTERNAL conseguimos que los valores de los campos se pasen de una ventana a otra y por todos los controles de la ventana.
WORKING	01 STAAGE GLOBAL PIC XX. 01 MENSAJE IS GLOBAL. 02 FILLER PIC X(7) VALUE "ERROR: ". 02 NUMSTA PIC 99BB. 02 FILLER PIC X(15) VALUE "POR EL MOTIVO: ". 02 NOMSTA PIC X(15). 01 TABLA GLOBAL. 02 TCOB PIC 9(4) OCCURS 3000 TIMES. 01 CCC GLOBAL PIC 9999. 01 TOPE GLOBAL PIC 9999. 01 AHIVA PIC 9(4) GLOBAL EXTERNAL.	Variable de estado. Mensaje en caso de error al acceder al fichero. Tabla las con claves de nuestro fichero. Contador. Tope de contador. Variable.
OPENED	ENVIRONMENT DIVISION. DATA DIVISION. WORKING-STORAGE SECTION. 01 MENSAJE. 02 FILLER PIC X(30) VALUE "SE VA CREAR EL ARCHIVO AGENDA". 01 TITULO. 02 FILLER PIC X(6) VALUE "ERROR". 01 ESTILO PIC S9(4) COMP-5. PROCEDURE DIVISION. DECLARATIVES. INICIO SECTION. USE AFTER ERROR PROCEDURE ON AGENDA. END DECLARATIVES. ADD POW-DMICONEXCLAMATION POW-DMOK GIVING ESTILO. OPEN INPUT AGENDA IF STAAGE = "35" CALL DISPLAYMESSAGE OF PRIMERA USING MENSAJE TITULO ESTILO OPEN OUTPUT AGENDA. CLOSE AGENDA. OPEN I-O AGENDA. CALL ADDSTRING OF AGE-TIP USING "PERSONAL ". CALL ADDSTRING OF AGE-TIP USING "PROFESIONAL". CALL ADDSTRING OF AGE-TIP USING "FAMILIAR ". CALL ADDSTRING OF AGE-TIP USING "EMPRESAS ". CALL ADDSTRING OF AGE-TIP USING "SANITARIO ". CALL "CARGAR".	Lo que se ejecutará justo al abrir la ventana. Como véis lleva su propia Working independiente de la otra. Usamos las Declaratives para controlar el status del fichero. Si el status es 35 significa que no existe y nos saca una ventana indicándonoslo. A continuación abrimos el archivo como I-O y cargamos el combobox con los tipos de contacto que vamos a tener. Hacemos una llamada a la rutina CARGAR.
CLOSE	ENVIRONMENT DIVISION. DATA DIVISION. PROCEDURE DIVISION.	Al cerrar la ventana, también cerramos nuestro fichero. Lo

	CLOSE AGENDA.	último que se hace.
CLOSECHILD	ENVIRONMENT DIVISION. DATA DIVISION. WORKING-STORAGE SECTION. 01 ESTILO PIC S9(4) COMP-5. PROCEDURE DIVISION. MOVE POW-ON TO POW-ENABLE OF PRIMERA. IF AHIVA > 0 GO SUERTE. CALL SETFOCUS OF AGE-COD. EXIT PROGRAM. SUERTE. MOVE AHIVA TO POW-NUMERIC OF AGE-COD. MOVE AHIVA TO AGE COD. READ AGENDA NOT INVALID KEY CALL "EXISTE". CALL "PONCCC". EXIT PROGRAM.	Esto se realizará cada vez que se cierre una ventana hija de ésta. La ponemos activa. Comprobamos si nos viene con un código y si es así procedemos a mostrarlo y sino se va de nuevo a pedir el código.

Bien, espero que hayais entendido mas o menos todo lo que hemos introducido. Recordad el evento CLOSE se realiza cuando se cierra la aplicación y el CLOSECHILD, cuando se cierra una de las ventanas hija.

Fijaros como controlamos si el estatus del fichero es 35 para hacer un OPEN OUTPUT y por lo tanto crear el fichero siempre que no exista.

Lo mas complicado de entender debe de ser el evento CLOSECHILD. Si recordáis tenemos una ventana que es de consulta en la cual salen todos nuestros contactos en una tabla y pinchando sobre uno de ellos se va y nos lo muestra en la ventana principal, lo podéis ver en el ejecutable directamente. Pues bien, la varibale ahiva, lo que nos da es el número de contacto sobre el que hemos pinchado dos veces en la consulta, por eso al cerrar la ventana hija comprubo si el valor de ahiva es mayor que 0 y entonces leo el registro seleccionado.

El valor de ahiva puede ser 0 si cerramos la ventana de consulta sin pinchar sobre ningún contacto o bien si hemos abieto la ventana de Acerca de.. ya que el tratamiento será el mismo porque ambas ventanas son hijas de la principal.

Si seguís teniendo alguna duda, antes de continuar me lo decís y lo explico mejor. La idea es que vosotros mismo podáis realizar una aplicación similar a ésta y para ello es mejor que comprendáis todo.

CAPTITULO 9. LA PROCEDURE

PROGRAMANDO LAS RUTINAS

Ya sabéis que cuando programamos con PowerCobol, en realidad estamos desarrollando una programación orientada a eventos, es decir los procesos se ejecutan cuando algún evento ha entrado en acción, por ejemplo al hacer clic con el ratón sobre algún Push Button, al introducir un texto en un Picture Edit, etc ... Pero también es posible que algunos de esos procesos se vayan a repetir en mas de una ocasión, entonces el hecho de copiarlos en cada evento no nos estaría facilitando la programación.

Para ello utiliza Power la Procedure de cada Sheet para introducir los procesos que no son propiedad de algún evento en concreto. En realidad son como pequeños programas alojados dentro del propio programa, ya que cada uno debe de tener un inicio con IDENTIFICATION DIVISION y un fin con END PROGRAM. Pero todas irán seguidas una debajo de otra dentro de la PROCEDURE DIVISION de la Sheet en que se vayan a utilizar.

En nuestro caso vamos a tener cuatro procedures, a continuación paso a explicaros el porque de cada una y su programación. Quiero hacer desde aquí una reflexión, ya que es un tema que nos puede interesar a todos, yo personalmente programo muy poco en programación estructurada, porque, por mi costumbre, porque nunca he programado en grupo y en la mayoría de los casos he sido autodidacta, por lo tanto me gustaría dejar claro que cada uno puede realizar los procesos como mejor sepa o como mejor se adapte a su manera de programar, yo utilizo el Go y para los ejemplos tengo que hacer uso de el.

RUTINA "EXISTE"

La primera se llama EXISTE y va a ser llamada siempre que introduzcamos un código de contacto que exista en nuestro fichero. Ya habéis que no existe en la aplicación un mantenimiento como tal, es decir con altas, bajas, consultas y modificaciones sino, que simplemente dependiendo del código que introduzcamos haremos una acción u otra. Si al introducir un código, éste no existe, nos permitirá darlo de alta y si por el contrario existe, se nos visualizará en pantalla y podremos modificarlo, consultarlo o borrarlo.

En nuestro caso, además al comenzar todos los campos de nuestro mantenimiento están desactivados para obligar a que solo se pueda introducir el código. Por eso en ésta rutina además de mover el contenido del registro a los controles de la pantalla, los ponemos en enable para poder entrar en ellos y además el botón de Grabar pasa a tener el texto Regrabar para indicar que se va a modificar.

Esta sería su programación.

```
Programación
IDENTIFICATION DIVISION.
PROGRAM-ID. EXISTE IS COMMON.
ENVIRONMENT DIVISION.
DATA DIVISION.
PROCEDURE DIVISION.
INICIO SECTION.
    MOVE 0 TO POW-ENABLE OF AGE-COD.
    MOVE AGECOD TO POW-NUMERIC OF AGE-COD.
    MOVE AGENOM TO POW-TEXT OF AGE-NOM.
    MOVE AGEDOM TO POW-TEXT OF AGE-DOM.
    MOVE AGEPOS TO POW-NUMERIC OF AGE-POS.
    MOVE AGEPOB TO POW-TEXT OF AGE-POB.
    MOVE AGEPRO TO POW-TEXT OF AGE-PRO.
    IF AGEGEN = "H" MOVE POW-ON TO POW-CHECK OF AGE-GEH
    ELSE MOVE POW-ON TO POW-CHECK OF AGE-GEM.
    MOVE AGETIP TO POW-SELECT OF AGE-TIP.
    MOVE AGETEL TO POW-TEXT OF AGE-TEL.
    MOVE AGEMOV TO POW-TEXT OF AGE-MOV.
    MOVE AGEMAI TO POW-TEXT OF AGE-MAI.
    MOVE AGEWEB TO POW-TEXT OF AGE-WEB.
    MOVE 1 TO POW-ENABLE OF GRABAR.
    MOVE 1 TO POW-ENABLE OF L-GRABAR.
    MOVE 1 TO POW-ENABLE OF BORRAR.
    MOVE 1 TO POW-ENABLE OF L-BORRAR.
    MOVE 1 TO POW-ENABLE OF LISTAR.
    MOVE 1 TO POW-ENABLE OF L-LISTAR.
    MOVE 1 TO POW-ENABLE OF EMAIL.
    MOVE 1 TO POW-ENABLE OF L-EMAIL.
    MOVE 1 TO POW-ENABLE OF CANCELAR.
    MOVE 1 TO POW-ENABLE OF L-CANCELAR.
    MOVE 1 TO POW-ENABLE OF AGE-NOM.
    MOVE 1 TO POW-ENABLE OF AGE-DOM.
    MOVE 1 TO POW-ENABLE OF AGE-POB.
    MOVE 1 TO POW-ENABLE OF AGE-POS.
    MOVE 1 TO POW-ENABLE OF AGE-PRO.
    MOVE 1 TO POW-ENABLE OF AGE-GEH.
    MOVE 1 TO POW-ENABLE OF AGE-GEM.
```

```

MOVE 1 TO POW-ENABLE OF AGE-TIP.
MOVE 1 TO POW-ENABLE OF AGE-TEL.
MOVE 1 TO POW-ENABLE OF AGE-MOV.
MOVE 1 TO POW-ENABLE OF AGE-MAI.
MOVE 1 TO POW-ENABLE OF AGE-WEB.
MOVE "Regrabar" TO POW-TEXT OF L-GRABAR.
CALL SETFOCUS OF CANCELAR.
EXIT PROGRAM.
END PROGRAM EXISTE.

```

RUTINA "CARGAR"

Esta segunda es una rutina un poco particular y es que a mi, me gusta siempre que voy a trabajar con un fichero y por supuesto siempre que no sea enormemente grande, me gusta cargarlo en una tabla y así el recorrido a través de él lo hago de manera mas rápida y segura. Con ellos conseguimos saber el número de registros que tenemos, al movernos hacia delante o hacia atrás ir con seguridad a los registros y la lectura siempre se hace directa sobre el registro en cuestión puesto que lo tenemos siempre en la tabla.

Como véis es una simple rutina con un contador y una tabla de un elemento para ir cargando los códigos de nuestra agenda. Al final guardamos en la variable tope el número de registros totales.

Esta rutina será llamada siempre que se haga un cambio en el fichero, es decir cuando demos un nuevo contacto de alta o cuando borremos alguno, para que la tabla siempre sea el reflejo del fichero. Esta acción es sumamente rápida en ningún caso ralentizará el programa.

Esta sería su programación.

Programación

```

IDENTIFICATION DIVISION.
PROGRAM-ID. CARGAR IS COMMON.
ENVIRONMENT DIVISION.
DATA DIVISION.
PROCEDURE DIVISION.
INICIO SECTION.
    MOVE 0 TO CCC.
    CLOSE AGENDA.
    OPEN INPUT AGENDA.
    INITIALIZE TABLA.
UNO.
    READ AGENDA NEXT RECORD AT END GO DOS.
    ADD 1 TO CCC MOVE AGECD TO TCD (CCC).
    GO UNO.
DOS.
    CLOSE AGENDA.
    OPEN I-O AGENDA.
    MOVE CCC TO TOPE MOVE 0 TO CCC.
    MOVE 1 TO POW-ENABLE OF AGE-COD.
    CALL SETFOCUS OF AGE-COD.
    EXIT PROGRAM.
END PROGRAM CARGAR.

```

RUTINA "PARASALIR"

Esta rutina tiene poco que explicar, simplemente se encarga de sacar una ventana en la pantalla en la que nos pregunta si estamos de acuerdo en abandonar la aplicación.

Esta rutina la he puesto porque hay mas de un sitio desde el cual podemos salir, desde un botón en la pantalla principal y desde una opción del menú.

Esta sería su programación.

```
Programación
IDENTIFICATION DIVISION.
PROGRAM-ID. PARASALIR IS COMMON.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 ESTILO PIC S9(4) COMP-5.
PROCEDURE DIVISION.
    ADD POW-DMYESNO POW-DMICONQUESTION GIVING ESTILO
    CALL DISPLAYMESSAGE OF PRIMERA USING "Salir de la aplicación ?"
"Mensaje" ESTILO.
    IF PROGRAM-STATUS = POW-DMRYES GO UNO ELSE GO FINALIZAR.
UNO.
    CALL CLOSESHEET OF PRIMERA.
    EXIT PROGRAM.
FINALIZAR.
    EXIT PROGRAM.
END PROGRAM PARASALIR.
```

-

RUTINA "PONCCC"

Perdonad por los nombres, es la fuerza de la costumbre. Pero en realidad esta rutina lo que hace es tener siempre localizado el elemento de la tabla que estamos utilizando. Es decir siempre que consultemos cualquier código la tabla también debe de posicionarse en ese punto. De esa manera en el momento en que pulsemos el botón de siguiente o anterior registro, lo haga teniendo como referencia el registro último que hemos consultado.

Así que lo único que hace es recorrer la table en busca del registro actual.

Esta sería su programación.

```
Programación
IDENTIFICATION DIVISION.
PROGRAM-ID. PONCCC IS COMMON.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 NADA PIC X.
PROCEDURE DIVISION.
INICIO SECTION.
    PERFORM VARYING CCC FROM 1 BY 1
        UNTIL CCC = TOPE OR TCOD (CCC) = AGECD
        MOVE ' ' TO NADA
    END-PERFORM.
EXIT PROGRAM.
END PROGRAM PONCCC.
```

Finalizadas las explicaciones sobre la Sheet en los siguientes capítulos veremos la programación de los eventos de los controles que haya en pantalla. Como veís hasta ahora siempre hemos estado utilizando código perfectamente familiar para nosotros, por lo que podemos decir que el hecho de programar en Power no nos hace abandonar nuestro lenguaje habitual.

CAPTITULO 10. PROGRAMANDO CONTROLES

PROGRAMANDO LOS CONTROLES (I)

En el capítulo anterior vimos la programación de las rutinas ahora es el momento de ver como se programan los eventos de los controles que tenemos en nuestra pantalla.

Una de las grandes diferencias que al menos yo he encontrado, es que cuando se programa de manera tradicional, vas teniendo el control de cada campo por separado, ahora en cambio el usuario con el ratón se puede ir a cualquier campo y dejar otros en blanco. Para evitar un poco el descontrol, he optado por lo siguiente. Nuestro primer campo a pedir es el código y por lo tanto una vez nos situemos sobre el siempre vamos a deshabilitar todos los demás campos y dejarlos sin contenido, así el usuario estará obligado a entrar un código antes de hacer nada, además aprovechamos y colocamos el número siguiente de código libre, veamos su programación:

Control: **AGE-COD**
Evento : **EDIT**

Programación

```
ENVIRONMENT DIVISION.  
DATA DIVISION.  
WORKING-STORAGE SECTION.  
01 SIGUE PIC 9999.  
PROCEDURE DIVISION.  
    MOVE SPACES TO POW-TEXT OF AGE-NOM  
    MOVE SPACES TO POW-TEXT OF AGE-DOM  
    MOVE SPACES TO POW-TEXT OF AGE-POB  
    MOVE 0 TO POW-NUMERIC OF AGE-POS  
    MOVE SPACES TO POW-TEXT OF AGE-PRO  
    MOVE SPACES TO POW-TEXT OF AGE-MOV  
    MOVE SPACES TO POW-TEXT OF AGE-TEL  
    MOVE SPACES TO POW-TEXT OF AGE-MAI  
    MOVE SPACES TO POW-TEXT OF AGE-WEB  
  
    MOVE 0 TO POW-ENABLE OF AGE-NOM  
    MOVE 0 TO POW-ENABLE OF AGE-DOM  
    MOVE 0 TO POW-ENABLE OF AGE-POB  
    MOVE 0 TO POW-ENABLE OF AGE-POS  
    MOVE 0 TO POW-ENABLE OF AGE-PRO  
    MOVE 0 TO POW-ENABLE OF AGE-GEH  
    MOVE 0 TO POW-ENABLE OF AGE-GEM  
    MOVE 0 TO POW-ENABLE OF AGE-TIP  
    MOVE 0 TO POW-ENABLE OF AGE-MOV  
    MOVE 0 TO POW-ENABLE OF AGE-TEL  
    MOVE 0 TO POW-ENABLE OF AGE-MAI  
    MOVE 0 TO POW-ENABLE OF AGE-WEB  
    MOVE 0 TO POW-ENABLE OF GRABAR  
    MOVE 0 TO POW-ENABLE OF L-GRABAR  
    MOVE 0 TO POW-ENABLE OF BORRAR  
    MOVE 0 TO POW-ENABLE OF L-BORRAR  
    MOVE 0 TO POW-ENABLE OF LISTAR  
    MOVE 0 TO POW-ENABLE OF L-LISTAR  
    MOVE 0 TO POW-ENABLE OF EMAIL  
    MOVE 0 TO POW-ENABLE OF L-EMAIL  
    MOVE 0 TO POW-ENABLE OF CANCELAR  
    MOVE 0 TO POW-ENABLE OF L-CANCELAR  
  
    COMPUTE SIGUE = TOPE + 1
```

MOVE SIGUE TO POW-NUMERIC OF AGE-COD.

Ahora, una vez pulsado Return sobre el código introducido generamos otro evento en el que se comprueba si existe o no y dependiendo se realizará una función u otra. Si existe el código llamamos a la rutina existe, que se encarga de colocarnos todos los contenidos de los campos y podremos, modificarlos, borrar, regrabar, etc En caso de no existir pone de nuevo los demás campos habilitados y nos envía al siguiente que en este caso es AGE-NOM. Veamos su programación:

Control: **AGE-COD**

Evento : **RETURN**

Programación

```
ENVIRONMENT DIVISION.  
DATA DIVISION.  
PROCEDURE DIVISION.  
    MOVE POW-NUMERIC OF AGE-COD TO AGECOD  
    IF AGECOD = 0 GO DOS  
    END-IF  
    READ AGENDA INVALID KEY GO TRES  
    NOT INVALID KEY CALL "EXISTE"  
    END-READ  
    CALL "PONCCC"  
    EXIT PROGRAM.  
  
DOS.  
    CALL SETFOCUS OF AGE-COD  
    EXIT PROGRAM.  
  
TRES.  
    MOVE 1 TO POW-ENABLE OF AGE-NOM  
    MOVE 1 TO POW-ENABLE OF AGE-DOM  
    MOVE 1 TO POW-ENABLE OF AGE-POB  
    MOVE 1 TO POW-ENABLE OF AGE-POS  
    MOVE 1 TO POW-ENABLE OF AGE-PRO  
    MOVE 1 TO POW-ENABLE OF AGE-GEH  
    MOVE 1 TO POW-ENABLE OF AGE-GEM  
    MOVE 1 TO POW-ENABLE OF AGE-TIP  
    MOVE 1 TO POW-ENABLE OF AGE-MOV  
    MOVE 1 TO POW-ENABLE OF AGE-TEL  
    MOVE 1 TO POW-ENABLE OF AGE-MAI  
    MOVE 1 TO POW-ENABLE OF AGE-WEB  
    MOVE 1 TO POW-SELECT OF AGE-TIP  
    INITIALIZE REGAGE  
    MOVE POW-NUMERIC OF AGE-COD TO AGECOD  
    CALL SETFOCUS OF AGE-NOM  
    MOVE "Grabar" TO POW-TEXT OF L-GRABAR  
    MOVE 1 TO POW-ENABLE OF GRABAR  
    MOVE 1 TO POW-ENABLE OF L-GRABAR  
    MOVE 1 TO POW-ENABLE OF CANCELAR  
    MOVE 1 TO POW-ENABLE OF L-CANCELAR  
    EXIT PROGRAM.
```

A partir de aquí y para los demás controles normalmente lo que haremos será ir pasando el foco al siguiente campo cada vez que pulsemos RETURN, como lo haríamos en un programa convencional. Por supuesto el usuario podrá ir libremente por cada campo. Cuando se pulse el botón de Grabar será cuando se realicen los controles en los campos que consideremos como obligatorios. Veamos toda la programación aunque sea muy simple.

Control: **AGE-NOM**

Evento : **RETURN**

Programación

ENVIRONMENT DIVISION.
DATA DIVISION.
PROCEDURE DIVISION.
CALL SETFOCUS OF AGE-DOM

Control: **AGE-DOM**
Evento : **RETURN**

Programación

ENVIRONMENT DIVISION.
DATA DIVISION.
PROCEDURE DIVISION.
CALL SETFOCUS OF AGE-POB

Control: **AGE-POB**
Evento : **RETURN**

Programación

ENVIRONMENT DIVISION.
DATA DIVISION.
PROCEDURE DIVISION.
CALL SETFOCUS OF AGE-POS

Control: **AGE-POS**
Evento : **RETURN**

Programación

ENVIRONMENT DIVISION.
DATA DIVISION.
PROCEDURE DIVISION.
CALL SETFOCUS OF AGE-PRO

Control: **AGE-PRO**
Evento : **RETURN**

Programación

ENVIRONMENT DIVISION.
DATA DIVISION.
PROCEDURE DIVISION.
CALL SETFOCUS OF AGE-GEH

Control: **AGE-GEH**
Evento : **CLICK**

Programación

ENVIRONMENT DIVISION.
DATA DIVISION.
PROCEDURE DIVISION.
IF POW-CHECK OF AGE-GEH = POW-ON MOVE "H" TO AGEGEN ELSE
MOVE "M" TO AGEGEN
END-IF
CALL SETFOCUS OF AGE-TIP

Control: **AGE-GEM**
Evento : **CLICK**

Programación

ENVIRONMENT DIVISION.
DATA DIVISION.
PROCEDURE DIVISION.
IF POW-CHECK OF AGE-GEM = POW-ON MOVE "M" TO AGEGEN ELSE
MOVE "H" TO AGEGEN
END-IF
CALL SETFOCUS OF AGE-TIP

Control: **AGE-TIP**
Evento : **CHANGE**

Programación

```
ENVIRONMENT DIVISION.  
DATA DIVISION.  
PROCEDURE DIVISION.  
    MOVE POW-SELECT OF AGE-TIP TO AGETIP  
    CALL SETFOCUS OF AGE-TEL
```

Control: **AGE-TIP**
Evento : **SELCHANGE**

Programación

```
ENVIRONMENT DIVISION.  
DATA DIVISION.  
PROCEDURE DIVISION.  
    MOVE POW-SELECT OF AGE-TIP TO AGETIP  
    CALL SETFOCUS OF AGE-TEL
```

Control: **AGE-TIP**
Evento : **EDIT**

Programación

```
ENVIRONMENT DIVISION.  
DATA DIVISION.  
PROCEDURE DIVISION.  
    IF POW-TEXT OF L-GRABAR = "Grabar"  
        MOVE 1 TO POW-SELECT OF AGE-TIP  
    END-IF
```

Control: **AGE-TEL**
Evento : **RETURN**

Programación

```
ENVIRONMENT DIVISION.  
DATA DIVISION.  
PROCEDURE DIVISION.  
    CALL SETFOCUS OF AGE-MOV
```

Control: **AGE-MOV**
Evento : **RETURN**

Programación

```
ENVIRONMENT DIVISION.  
DATA DIVISION.  
PROCEDURE DIVISION.  
    CALL SETFOCUS OF AGE-MAI
```

Control: **AGE-MAI**
Evento : **RETURN**

Programación

```
ENVIRONMENT DIVISION.  
DATA DIVISION.  
PROCEDURE DIVISION.  
    CALL SETFOCUS OF AGE-WEB
```

Control: **AGE-WEB**
Evento : **RETURN**

Programación

```
ENVIRONMENT DIVISION.  
DATA DIVISION.  
PROCEDURE DIVISION.  
    CALL SETFOCUS OF GRABAR
```

Como véis casi todo ha sido igual, con la excepción de los controles que hacen referencia al género y al tipo de contacto. Si os dais cuenta el evento que se produce cuando editamos AGE-TIP, determinamos que si estamos introduciendo un contacto nuevo (cosa que sabemos por el nombre del control GRABAR), seleccionamos directamente el valor 1.

En el próximo capítulo se explicarán los eventos de los controles que conforman la barra de herramientas, que son los decisivos y sobre los cuales controlaremos nuestro archivo.

CAPTITULO 11. PROGRAMANDO CONTROLES

PROGRAMANDO LOS CONTROLES (II)

Seguimos con la programación de los controles, en este capítulo veremos la programación de los eventos de los controles que conforman la barra de herramientas. Es decir los botones que tenemos abajo en un recuadro azul y que nos permiten controlar el fichero.

Empezando de izquierda a derecha, nos encontramos con cuatro botones con flechas que nos van a servir para desplazarnos rápidamente por el fichero. Al ser todos botones, el evento que tenemos en cada caso es el que hace referencia al CLICK sobre dicho botón, veamos la programación de los cuatro botones.

Control: **INICIO**
Evento : **CLICK**

```
Programación
ENVIRONMENT DIVISION.
DATA      DIVISION.
WORKING-STORAGE SECTION.
PROCEDURE DIVISION.
    IF TOPE = 0 GO DOS.
    MOVE TCOD (1) TO AGECD
    MOVE 1 TO CCC
    READ AGENDA INVALID KEY GO DOS
        NOT INVALID KEY CALL "EXISTE".
    EXIT PROGRAM.
DOS.
    CALL SETFOCUS OF AGE-COD
    EXIT PROGRAM.
```

Como teníamos los códigos de nuestros contactos en una tabla, lo único que hacemos es mover el primer elemento de la tabla y leer. Para irnos al final del fichero hacemos lo mismo, solo que moviendo el último elemento de la tabla, veámoslo:

Control: **FIN**
Evento : **CLICK**

```
Programación
ENVIRONMENT DIVISION.
DATA      DIVISION.
PROCEDURE DIVISION.
    IF TOPE = 0 GO DOS.
    MOVE TCOD (TOPE) TO AGECD
    MOVE 1 TO CCC
    READ AGENDA INVALID KEY GO DOS
        NOT INVALID KEY CALL "EXISTE".
    EXIT PROGRAM.
DOS.
    CALL SETFOCUS OF AGE-COD
    EXIT PROGRAM.
```

Ahora veamos como avanzamos o retrocedemos por el fichero con los botones de las flechas izquierda y derecha:

Control: **MAS**
Evento : **CLICK**

Programación

```
ENVIRONMENT DIVISION.  
DATA DIVISION.  
PROCEDURE DIVISION.  
    IF TOPE = 0 GO DOS.  
    ADD 1 TO CCC IF CCC > TOPE MOVE TOPE TO CCC.  
    MOVE TCODE (CCC) TO AGECOD.  
    READ AGENDA INVALID KEY GO DOS  
        NOT INVALID KEY CALL "EXISTE".  
    EXIT PROGRAM.  
DOS.  
    CALL SETFOCUS OF AGE-COD  
    EXIT PROGRAM.
```

Control: **MENOS**
Evento : **CLICK**

Programación

```
ENVIRONMENT DIVISION.  
DATA DIVISION.  
PROCEDURE DIVISION.  
    IF TOPE = 0 GO DOS.  
    SUBTRACT 1 FROM CCC IF CCC < 1 MOVE 1 TO CCC.  
    MOVE TCODE (CCC) TO AGECOD.  
    READ AGENDA INVALID KEY GO DOS  
        NOT INVALID KEY CALL "EXISTE".  
    EXIT PROGRAM.
```

Una vez vistos los controles para el movimiento por el fichero, seguiremos analizando los demás botones de la barra de herramientas.

El botón que sigue es el de borrar y su objetivo será el de eliminar el registro que en ese momento se encuentre en la pantalla. Como es algo delicado aprovechamos la posibilidad de sacar una ventana de verificación con DISPLAYMESSAGE y si la respuesta es positiva borramos el registro. Este botón estará activo solo cuando tengamos un registro presente. Veamos su programación:

Control: **BORRAR**
Evento : **CLICK**

Programación

```
ENVIRONMENT DIVISION.  
DATA DIVISION.  
WORKING-STORAGE SECTION.  
01 ESTILO PIC S9(4) COMP-5.  
PROCEDURE DIVISION.  
    ADD POW-DMYESNO POW-DMICONQUESTION GIVING ESTILO  
    CALL DISPLAYMESSAGE OF PRIMERA  
        USING "Seguro de borrarlo ?" "Mensaje" ESTILO.  
    IF PROGRAM-STATUS = POW-DMRYES GO UNO ELSE GO FINALIZAR.  
UNO.  
    ADD POW-DMICONEXCLAMATION POW-DMOK GIVING ESTILO.  
    DELETE AGENDA INVALID KEY  
        MOVE STAGE TO NUMSTA MOVE "BORRAR" TO NOMSTA  
        CALL DISPLAYMESSAGE OF PRIMERA  
            USING MENSAJE "ERROR" ESTILO.  
    CALL "CARGAR".  
    EXIT PROGRAM.  
FINALIZAR.  
    EXIT PROGRAM.
```

A continuación el botón cancelar lo único que hace es volver a enviar el foco al control AGE-COD, para introducir un nuevo código.

Control: **CANCELAR**
Evento : **CLICK**

Programación

```

ENVIRONMENT DIVISION.
DATA      DIVISION.
PROCEDURE DIVISION.
    MOVE 1 TO POW-ENABLE OF AGE-COD
    CALL SETFOCUS OF AGE-COD.

```

eL Botón de listar, lo que hará será una impresión de la pantalla, pero claro, solo de los campos que tengamos con la propiedad de Printable seleccionada. Esto viene muy bien y nos puede ayudar mucho por la facilidad de uso. Este control solo estará activo cuando tengamos un registro presente. Como aclaración os digo que los controles de ListBox y Table, aunque su contenido no quepa en la pantalla, en la impresora si se imprimirán por completo. Al igual que en el anterior mostramos previamente una ventana de advertencia.

Control: **LISTAR**
Evento : **CLICK**

Programación
<pre> ENVIRONMENT DIVISION. DATA DIVISION. WORKING-STORAGE SECTION. 01 ESTILO PIC S9(4) COMP-5. PROCEDURE DIVISION. ADD POW-DMYESNO POW-DMICONQUESTION GIVING ESTILO CALL DISPLAYMESSAGE OF PRIMERA USING "Listar la ficha ?" "Mensaje" ESTILO. IF PROGRAM-STATUS = POW-DMRYES GO UNO ELSE GO FINALIZAR. UNO. CALL OPENPRINTER OF PRINT1. CALL WRITESHEET OF PRINT1. CALL CLOSEPRINTER OF PRINT1. EXIT PROGRAM. FINALIZAR. EXIT PROGRAM. </pre>

El botón del e-mail, nos llamará al Outlook Express de Microsoft para enviar un correo como vimos en la sección de trucos. Este control solo estará disponible cuando haya un registro presente. También comprobaremos que el campo del e-mail en nuestro registro tenga datos antes de enviar nada.

Control: **EMAIL**
Evento : **CLICK**

Programación
<pre> ENVIRONMENT DIVISION. DATA DIVISION. WORKING-STORAGE SECTION. 01 LLAMADA. 02 FILLER PIC X(30) VALUE "c:\archiv~1\outloo~1\msimn.exe". 02 FILLER PIC X(17) VALUE " /mailurl:mailto:". 02 LLAMAI PIC X(30). 01 ESTILO PIC S9(4) COMP-5. PROCEDURE DIVISION. IF AGEMAI = SPACES GO SINMAIL. MOVE AGEMAI TO LLAMAI. CALL EXECAPL OF DDE1 USING LLAMADA POW-SWNORMAL. EXIT PROGRAM. SINMAIL. ADD POW-DMOK POW-DMICONSTOP GIVING ESTILO CALL DISPLAYMESSAGE OF PRIMERA USING "Introduzca una dirección de correo" "Error" ESTILO. EXIT PROGRAM. </pre>

Veamos ahora el botón de grabar los datos del registro. Este botón cambia de nombre (GRABAR, REGRABAR) dependiendo si es un alta o una modificación, así también controlaremos si es un WRITE o un REWRITE lo que tenemos que hacer. Es el evento mas largo de todos los que hemos visto. Este será el momento de mover todos los datos que hemos introducido en los controles a nuestros campos del

registro y de comprobar cuales de ellos queremos como obligatorios, en nuestro caso solo el del nombre, pero si queréis poner mas solo tenéis que añadir líneas iguales a las del control del nombre. Veamos su programación:

Control: **GRABAR**
 Evento : **CLICK**

Programación
<pre> ENVIRONMENT DIVISION. DATA DIVISION. WORKING-STORAGE SECTION. 01 ESTILO PIC S9(4) COMP-5. PROCEDURE DIVISION. MOVE POW-TEXT OF AGE-NOM TO AGENOM. MOVE POW-TEXT OF AGE-DOM TO AGEDOM. MOVE POW-TEXT OF AGE-POB TO AGEPOB. MOVE POW-NUMERIC OF AGE-POS TO AGEPOS. MOVE POW-TEXT OF AGE-PRO TO AGEPRO. IF POW-CHECK OF AGE-GEH = POW-ON MOVE "H" TO AGEGEN ELSE MOVE "M" TO AGEGEN END-IF MOVE POW-SELECT OF AGE-TIP TO AGETIP. MOVE POW-TEXT OF AGE-TEL TO AGETEL. MOVE POW-TEXT OF AGE-MOV TO AGEMOV. MOVE POW-TEXT OF AGE-MAI TO AGEMAI. MOVE POW-TEXT OF AGE-WEB TO AGEWEB. IF AGENOM = SPACES GO SINNOM. ADD POW-DMICONEXCLAMATION POW-DMOK GIVING ESTILO. IF POW-TEXT OF L-GRABAR = "Grabar" WRITE REGAGE INVALID KEY MOVE STAAGE TO NUMSTA MOVE "GRABAR" TO NOMSTA CALL DISPLAYMESSAGE OF PRIMERA USING MENSAJE "ERROR" ESTILO. IF POW-TEXT OF L-GRABAR = "Regrabar" REWRITE REGAGE INVALID KEY MOVE STAAGE TO NUMSTA MOVE "REGRABAR" TO NOMSTA CALL DISPLAYMESSAGE OF PRIMERA USING MENSAJE "ERROR" ESTILO. CALL "CARGAR". CALL "PONCCC". EXIT PROGRAM. SINNOM. ADD POW-DMICONEXCLAMATION POW-DMOK GIVING ESTILO. CALL DISPLAYMESSAGE OF PRIMERA USING "Nombre obligatorio" "ERROR" ESTILO. CALL SETFOCUS OF AGE-NOM. EXIT PROGRAM. </pre>

Los botones que nos quedan están repetidos en la aplicación, ya que podremos acceder desde ellos o desde el menú que creamos en el capítulo 4. El código lo pondremos en los dos eventos y así actuaraá igual si se pulsa el botón o si se selecciona la opción desde el menú. Cuando se creo el menú no se le pusieron nombre a las opciones, así que Power le habrá puesto una por defecto del tipo MENU1, MENU2, etc ... Solo tenéis que buscar la correspondiente y asignarle el evento CLICK.

El botón de CONSULTA, simplemente nos llama a otra ventana en la cual vamos a efectuar la consulta y además pone la pantalla principal deshabilitada para que no se pueda tocar mientras estamos en la consulta.

Control: **CONSULTA - MENUCONSULTA**
 Evento : **CLICK**

Programación
<pre> ENVIRONMENT DIVISION. DATA DIVISION. PROCEDURE DIVISION. MOVE POW-OFF TO POW-ENABLE OF PRIMERA CALL OPENSHEET OF PRIMERA USING "CONSULTA". </pre>

El botón de SALIR, nos sirve para abandonar la aplicación y lo único que hace es llamar a la rutina PARASALIR.

Control: **SALIR** - **MENUSALIR**
Evento : **CLICK**

Programación
ENVIRONMENT DIVISION. DATA DIVISION. PROCEDURE DIVISION. CALL "PARASALIR".

En este caso no es el botón sino, la imagen que tenemos a la izquierda de la ventana, la que pone la dirección de la página, la que nos sirve de botón y al igual que las otras se repite en el menú y con ella sacamos una ventana donde se nos explica sobre el programa.

Control: **IMAGEN** - **MENUIMAGEN**
Evento : **CLICK**

Programación
ENVIRONMENT DIVISION. DATA DIVISION. PROCEDURE DIVISION. MOVE POW-OFF TO POW-ENABLE OF PRIMERA CALL OPENSHEET OF PRIMERA USING "ACERCA".

Llegados a este punto, toda la programación de nuestra ventana principal se da por terminada y para los capítulos restantes, nos queda explicar la ventana de ACERCA que es muy fácil y la que he realizado para la consulta.

Si hasta aquí algo no ha quedado claro, es el momento de las consultas. Ya que lo importante no es que salga bien sino que comprendáis todo lo que se ha explicado.

En este momento se puede compilar todo y debe de funcionar sin problemas, altas, bajas, modificar, todo excepto la consulta.

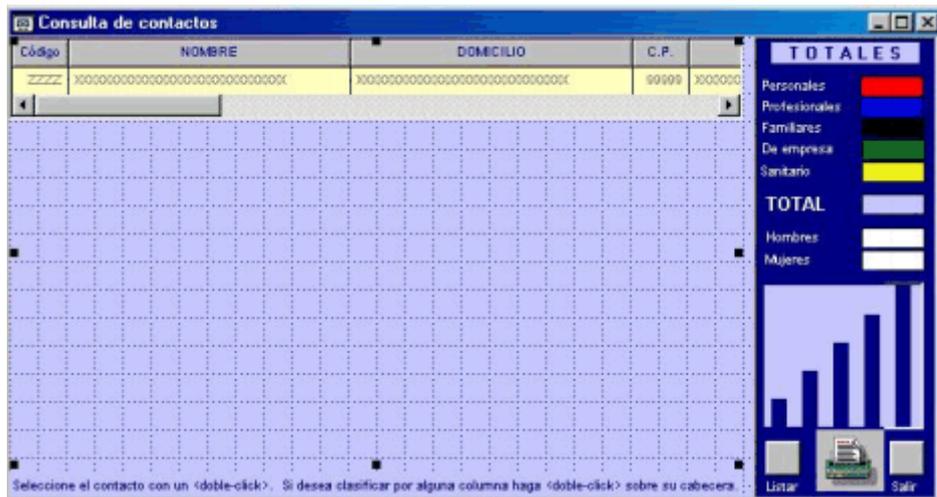
CAPTITULO 12. EMPEZANDO LA CONSULTA

LA PANTALLA DE CONSULTA

Vamos con la segunda pantalla de nuestra aplicación, una pantalla mucho mas compleja y efectiva que la de mantenimiento. Si bien en esa primera nos limitabamos a colocar campos de edición para introducir los datos, en ésta segunda vamos a hacer una consulta completa y en la que aparecerán nuevos controles a los que aplicaremos nuevos eventos.

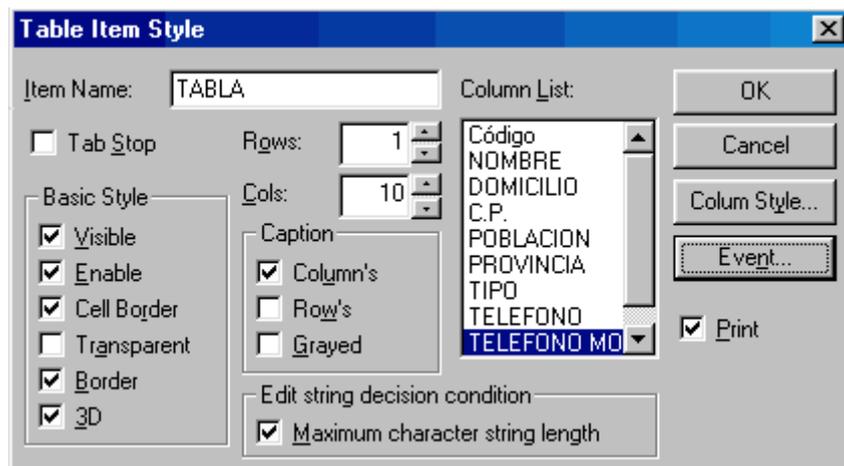
Para crear otra pantalla dentro de nuestro proyecto debemos seleccionar la opción **New**, del menú de **File**. Para el estilo de la ventana marcamos: Pop-up Windows, Thin y Title Bar. La llamamos CONSULTA y en el título de la misma le ponemos Consulta de contactos.

A continuación os pongo una imagen de como debe de quedar la pantalla con todos sus controles, es una imagen en tiempo de desarrollo y seguidamente os explico las características de cada uno. La imagen no está a tamaño real, está exactamente a un 70% para que pudiera entrar en la página.



TABLE

El control mas importante que contiene es la tabla, llamada por PowerCobol Table. La misma nos va a servir para incluir una lista separada por columnas con todos nuestros contactos, la cual podremos clasificar por cada uno de los campos que contiene. Las propiedades son las siguientes:



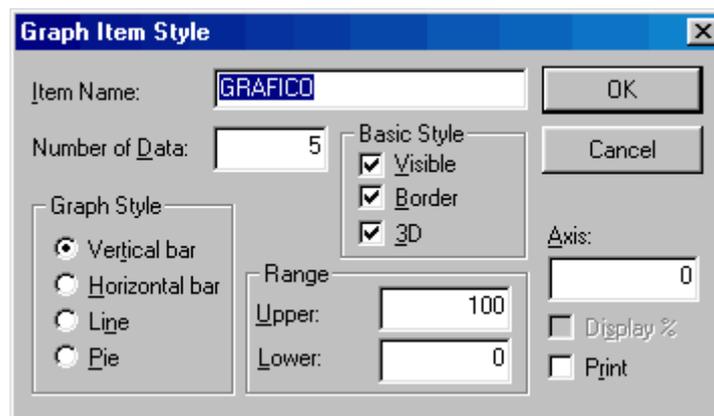
Le llamaremos como véis TABLA y tendrá todas las columnas que véis y una mas que no sale, llamada E-MAIL. Para cada una de las columnas están serán sus características:

CAPTION	PICTURE	WIDTH	ALIGNMENT	SCROLL LOCK
CODIGO	ZZZZ	40	RIGHT	SI
NOMBRE	X(30)	200	LEFT	SI
DOMICILIO	X(30)	200	LEFT	NO
C.P.	99999	40	RIGHT	NO
POBLACION	X(20)	132	LEFT	NO
PROVINCIA	X(15)	100	LEFT	NO
TIPO	X(12)	80	LEFT	NO
TELEFONO	X(20)	132	LEFT	NO
TELEFONO MOVIL	X(20)	132	LEFT	NO
E-MAIL	X(30)	200	LEFT	NO

He respetado los nombres de las propiedades en inglés, porque de esa manera os saldrán a vosotros. Indicaros que la opción Scroll Lock, lo que posibilita es que los campos marcados con esa propiedad no se muevan al desplazarnos horizontalmente por la tabla, con lo que el código y el nombre siempre estarán disponibles a la vista. Si os dais cuenta, al principio le he dicho que solo tiene una línea (Rows), una vez en ejecución se le irán añadiendo las líneas que sean necesarias según registros tengamos.

GRAPH

Para la representación gráfica de los datos, PowerCobol dispone de éste control, que aunque no muy extenso en sus posibilidades si es efectivo en cuanto a su cometido. Gracias a el, vamos a representar de forma gráfica la cantidad de contactos que tenemos en nuestro fichero según su tipo. Las propiedades de dicho gráfico al cual llamaremos desde nuestra aplicación GRAFICO, son las siguientes:

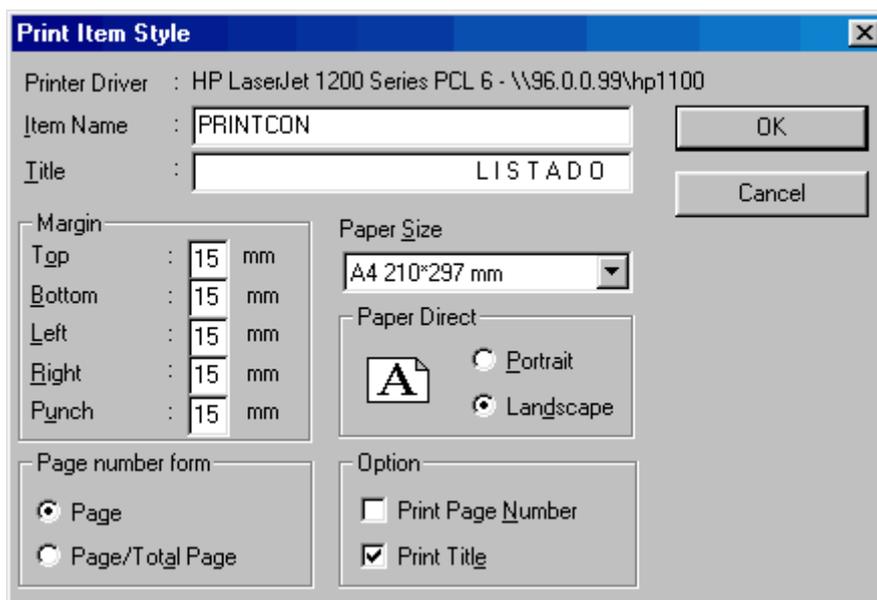


Lo mas reseñable de éste control es indicarle el tipo de gráfico que vamos a representar, vertical, horizontal, de líneas o en forma de queso, además de los límites tanto superior como inferior para la representación.

PRINT

Para evitar tener que confeccionar un listado y puesto que PowerCobol posee un control para imprimir los controles, podemos apoyarnos en él y aprovechar su potencia para imprimir la tabla y así conseguir un listado profesional sin escribir ni una sola línea de código. Para ello y antes de nada tenemos que tener en cuenta que la propiedad PRINT de nuestros controles, tiene que estar desactivada excepto para el control tabla que será el que se imprima. Además PowerCobol con éste control, me refiero a la tabla, tiene el privilegio de imprimirla completamente, aunque su contenido no quepa en la pantalla, con lo que ya veréis que bien queda y que fácil es conseguirlo.

Por supuesto podremos continuar imprimiendo utilizando nuestras definiciones de líneas descritas en la Working-Storage Section, pero para determinados casos podemos aprovecharnos de éste control. Sus propiedades son las siguientes:



Con éstas propiedades os garantizo que el listado queda muy bien en modo apaisado y sobre un papel tipo A4. Este control carece de eventos.

DEMÁS CONTROLES

Como podéis apreciar al resto de controles, son los habituales, LABEL y BUTTON, que nos ayudan a dar forma a la pantalla de consulta.

Antes de seguir os diré como se llaman cada uno de los campos que figuran a la derecha y que rellenaremos con los totales y subtotales. Además el fondo de dichos campos se ha puesto en colores que utilizaremos también para cada una de las barras del gráfico y así situar los datos mas rapidamente. El porque he utilizado LABEL en vez de EDIT es muy sencillo, para evitar que dichos campos se puedan modificar o tomar el foco. Al ponerlos como LABEL el compilador los considera como DISPLAYS normales y no podemos ni modificarlos ni editarlos ni nada similar.

- Para poner el número de contactos de tipo Personal, el LABEL se llama DE1.
- Para poner el número de contactos de tipo Profesional, el LABEL se llama DE2.
- Para poner el número de contactos de tipo Familiar, el LABEL se llama DE3.
- Para poner el número de contactos de tipo de Empresa, el LABEL se llama DE4.
- Para poner el número de contactos de tipo Sanitario, el LABEL se llama DE5.
- Para poner el número total de contactos, el LABEL se llama TOTAL.
- Para poner el número de contactos Hombres, el LABEL se llama DEH.
- Para poner el número de contactos Mujeres, el LABEL se llama DEM.

Veamos los eventos a programar sobre dicho formulario. En éste capítulo veremos solo algunos y dejaremos el resto de eventos, así como la programación del resto de eventos de los demás controles para el siguiente capítulo.

Evento	Programación	Ayuda
SPECIAL-NAMES	DECIMAL-POINT IS COMMA.	Indicamos la coma como punto decimal.

FILE-CONTROL	SELECT AGENDA ASSIGN TO "AGENDA.DAT" ORGANIZATION INDEXED ACCESS DYNAMIC RECORD KEY KEYAGE FILE STATUS STAAGE.	Definición de nuestros archivos.
	SELECT CLASI ASSIGN TO "CLASI.DAT".	Para clasificar.
FILE	FD AGENDA GLOBAL EXTERNAL LABEL RECORD STANDARD. 01 REGAGE. 02 KEYAGE. 03 AGECOD PIC 9999. 02 AGENOM PIC X(30). 02 AGEDOM PIC X(30). 02 AGEPOS PIC 99999. 02 AGEPOB PIC X(20). 02 AGEPRO PIC X(15). 02 AGEGEN PIC X. 02 AGETIP PIC 9. 02 AGETEL PIC X(20). 02 AGEMOV PIC X(20). 02 AGEMAI PIC X(30). 02 AGEWEB PIC X(40). FD CLASI GLOBAL LABEL RECORD STANDARD. 01 REGCLA. 02 CLACOD PIC 9999. 02 CLANOM PIC X(30). 02 CLADOM PIC X(30). 02 CLAPRO PIC 99999. 02 CLAPOB PIC X(20). 02 CLAPRO PIC X(15). 02 CLATIP PIC X(12). 02 CLATEL PIC X(20). 02 CLAMOV PIC X(20). 02 CLAMAI PIC X(30).	Descripción de los ficheros a utilizar en nuestra aplicación. Al ponerle GLOBAL EXTERNAL conseguimos que los valores de los campos se pasen de una ventana a otra y por todos los controles de la ventana. Esta definición corresponde al fichero que vamos a utilizar para la clasificación por las distintas columnas de la tabla.
WORKING	01 STAAGE GLOBAL PIC XX. 01 MENSAJE IS GLOBAL. 02 FILLER PIC X(7) VALUE "ERROR: ". 02 NUMSTA PIC 99BB. 02 FILLER PIC X(15) VALUE "POR EL MOTIVO: ". 02 NOMSTA PIC X(15). 01 CCC PIC 9999 GLOBAL. 01 TOPE PIC 9999 GLOBAL.. 01 AHIVA PIC 9(4) GLOBAL EXTERNAL.	Variable de estado. Mensaje en caso de error al acceder al fichero. Contador. Tope de contador. Variable.

El próximo capítulo abordará el evento OPENED, en el que se cargarán los datos en la tabla y se pondrán los totales por tipo, además de formar el gráfico y así terminaremos la pantalla de consulta.

Una vez concluido el diseño de la ventana, pinchamos sobre **Project**, **Edit** y a continuación pulsamos sobre el botón **Add**, para añadir esta ventana al proyecto y así poder compilar y ver como funciona. Una vez añadida pulsamos sobre el botón **Ok** y ya podemos ver en la pequeña ventana del proyecto que tenemos en la pantalla, como esta ventana de consulta está incluida.

CAPTITULO 13. CLASIFICACION Y VENTANA ACERCA DE (FIN)

EL EVENTO OPENED

En el capítulo anterior nos quedo por el evento que se produce al abrir la ventana, es decir, el que se produce en el OPENED. En el vamos a rellenar la tabla con los registros que tenemos en el fichero de Agenda y además crearemos el gráfico y totalizaremos por tipos de contactos.

En primer lugar nos definiremos una Working propia para trabajar sobre algunas variables, como contadores y la tabla con los tipos de contactos. Para seguir con la

estructura de programación que he utilizado durante todo el curso, voy a utilizar párrafos y sentencias GOTO, pero tengo intención de hacer un especial sobre programación estructurada en la que explicaré las diferencias entre un tipo de programación y otro, con ejemplos, así que ahora seguiré como siempre.

Fijaros que el párrafo UNO, lo utilizamos para leer el fichero de principio a fin e ir rellenando los campos que componen la tabla. El párrafo DOS lo utilizamos para construir el gráfico tomando como valor máximo el del tipo de contacto mayor.

PROGRAMACION EVENTO OPENED

```
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 COLUMNAS PIC 9999.
01 CUENTA PIC 999.
01 TIPO1 PIC 9(4).
01 TIPO2 PIC 9(4).
01 TIPO3 PIC 9(4).
01 TIPO4 PIC 9(4).
01 TIPO5 PIC 9(4).
01 DEBA PIC 9(4).
01 MAX PIC 9(4).
01 GENH PIC 9(4).
01 GENM PIC 9(4).
01 TIPO PIC X(12).
01 LNUMERO.
   02 LNUM PIC ZZ.ZZ9.
01 TABLATIPOS.
   02 FILLER PIC X(15) VALUE "PERSONAL ".
   02 FILLER PIC X(15) VALUE "PROFESIONAL".
   02 FILLER PIC X(15) VALUE "FAMILIAR".
   02 FILLER PIC X(15) VALUE "EMPRESAS".
   02 FILLER PIC X(15) VALUE "SANITARIO".
01 TABLITIPOS REDEFINES TABLATIPOS.
   02 ELETIP PIC X(15) OCCURS 5 TIMES.

PROCEDURE DIVISION.
INICIO SECTION.
   CLOSE AGENDA.
   OPEN INPUT AGENDA.
   OPEN OUTPUT CLASI.
   MOVE 0 TO POW-VISIBLE OF TABLA.
   MOVE 0 TO CCC TIPO1 TIPO2 TIPO3 TIPO4 TIPO5 GENH GENM.
UNO.
   READ AGENDA NEXT RECORD AT END GO DOS.
   ADD 1 TO CCC.
   MOVE CCC TO POW-ROWS OF TABLA.
   CALL SETCELLNUMERIC OF TABLA USING AGECOD CCC 1
   MOVE AGECOD TO CLACOD.
   CALL SETCELLTEXT OF TABLA USING AGENOM CCC 2
   MOVE AGENOM TO CLANOM.
   CALL SETCELLTEXT OF TABLA USING AGEDOM CCC 3
   MOVE AGEDOM TO CLADOM.
   CALL SETCELLNUMERIC OF TABLA USING AGEPOS CCC 4
   MOVE AGEPOS TO CLAPOS.
   CALL SETCELLTEXT OF TABLA USING AGEPOB CCC 5
   MOVE AGEPOB TO CLAPOB.
   CALL SETCELLTEXT OF TABLA USING AGEPRO CCC 6
   MOVE AGEPRO TO CLAPRO.
   MOVE ELETIP (AGETIP) TO TIPO.
   CALL SETCELLTEXT OF TABLA USING TIPO CCC 7
   MOVE TIPO TO CLATIP.
   CALL SETCELLTEXT OF TABLA USING AGETEL CCC 8
   MOVE AGETEL TO CLATEL.
   CALL SETCELLTEXT OF TABLA USING AGEMOV CCC 9
   MOVE AGEMOV TO CLAMOV.
   CALL SETCELLTEXT OF TABLA USING AGEMAI CCC 10
   MOVE AGEMAI TO CLAMAI.
   WRITE REGCLA.

EVALUATE AGETIP
```

```

WHEN 1 ADD 1 TO TIPO1
WHEN 2 ADD 1 TO TIPO2
WHEN 3 ADD 1 TO TIPO3
WHEN 4 ADD 1 TO TIPO4
WHEN 5 ADD 1 TO TIPO5
END-EVALUATE
EVALUATE AGEGEN
  WHEN "H" ADD 1 TO GENH
  WHEN "M" ADD 1 TO GENM
END-EVALUATE
GO UNO.

```

DOS.

```

MOVE 1 TO POW-VISIBLE OF TABLA.
MOVE CCC TO LNUM MOVE LNUMERO TO POW-TEXT OF TOTAL.
MOVE GENH TO LNUM MOVE LNUMERO TO POW-TEXT OF DEH.
MOVE GENM TO LNUM MOVE LNUMERO TO POW-TEXT OF DEM.
IF TIPO1 > TIPO2 AND TIPO1 > TIPO3 AND TIPO1 > TIPO4 AND TIPO1 > TIPO5
  COMPUTE MAX = TIPO1 + 2.
IF TIPO2 > TIPO1 AND TIPO2 > TIPO3 AND TIPO2 > TIPO4 AND TIPO2 > TIPO5
  COMPUTE MAX = TIPO2 + 2.
IF TIPO3 > TIPO1 AND TIPO3 > TIPO2 AND TIPO3 > TIPO4 AND TIPO3 > TIPO5
  COMPUTE MAX = TIPO3 + 2.
IF TIPO4 > TIPO1 AND TIPO4 > TIPO2 AND TIPO4 > TIPO3 AND TIPO4 > TIPO5
  COMPUTE MAX = TIPO4 + 2.
IF TIPO5 > TIPO1 AND TIPO5 > TIPO2 AND TIPO5 > TIPO3 AND TIPO5 > TIPO4
  COMPUTE MAX = TIPO5 + 2.
MOVE MAX TO POW-UPPER OF GRAFICO.
MOVE TIPO1 TO LNUM MOVE LNUMERO TO POW-TEXT OF DE1.
MOVE TIPO1 TO POW-DATA (1) OF GRAFICO.
MOVE POW-RED TO POW-DATACOLOR (1) OF GRAFICO.
MOVE TIPO2 TO LNUM MOVE LNUMERO TO POW-TEXT OF DE2.
MOVE TIPO2 TO POW-DATA (2) OF GRAFICO.
MOVE POW-BLUE TO POW-DATACOLOR (2) OF GRAFICO.
MOVE TIPO3 TO LNUM MOVE LNUMERO TO POW-TEXT OF DE3.
MOVE TIPO3 TO POW-DATA (3) OF GRAFICO.
MOVE POW-BLACK TO POW-DATACOLOR (3) OF GRAFICO.
MOVE TIPO4 TO LNUM MOVE LNUMERO TO POW-TEXT OF DE4.
MOVE TIPO4 TO POW-DATA (4) OF GRAFICO.
MOVE POW-DARKGREEN TO POW-DATACOLOR (4) OF GRAFICO.
MOVE TIPO5 TO LNUM MOVE LNUMERO TO POW-TEXT OF DE5.
MOVE TIPO5 TO POW-DATA (5) OF GRAFICO.
MOVE POW-YELLOW TO POW-DATACOLOR (5) OF GRAFICO.
CLOSE AGENDA CLASI OPEN I-O AGENDA.
EXIT PROGRAM.

```

Con los eventos vistos hasta el momento al llamar a la pantalla de consultas desde la pantalla principal nos saldrá perfectamente ésta pantalla. Pero sigamos con los demás eventos.

Para abandonar la pantalla, pulsaremos sobre el botón salir y producirá el siguiente evento. La variable AHIVA, es la que vamos a utilizar para pasar el valor de un contacto si es que pinchamos sobre el para que nos vuelva a la pantalla de mantenimiento. En este caso al darle a salir, le mandamos 0.

Control: **SALIR**
Evento : **CLICK**

Programación

```

ENVIRONMENT DIVISION.
DATA DIVISION.
PROCEDURE DIVISION.
  MOVE 0 TO AHIVA
  CALL CLOSESHEET OF CONSULTA.

```

Para imprimir la tabla utilizamos el botón listar y se producirá el siguiente evento. En este caso utilizamos una ventana de confirmación previa y fijaros de que forma tan simple conseguimos un listado muy bueno de nuestro fichero con el mínimo

esfuerzo. Aunque no será aplicable a todos los casos, si que en muchos de ellos nos va a servir.

Control: **LISTAR**
Evento : **CLICK**

```
Programación
ENVIRONMENT DIVISION.
DATA DIVISION.
PROCEDURE DIVISION.
    ADD POW-DMYESNO POW-DMICONQUESTION GIVING ESTILO
    CALL DISPLAYMESSAGE OF CONSULTA USING "Listar ?" "Mensaje" ESTILO
    IF PROGRAM-STATUS = POW-DMRYES GO UNO ELSE GO FINALIZAR.
UNO.
    CALL OPENPRINTER OF PRINTCON
    CALL WRITESHEET OF PRINTCON
    CALL CLOSEPRINTER OF PRINTCON
    EXIT PROGRAM.
FINALIZAR.
    EXIT PROGRAM.
```

Ahora vamos a ver la programación del evento DBCLICK en la tabla. Con el podremos hacer dos cosas. Una es seleccionar un contacto para llevarlo a la pantalla de mantenimiento, para lo cual comprobamos que la fila sobre la que se ha hecho el DBCLICK no sea la 0, entonces nos envía al párrafo UNO donde salimos de ésta ventana enviando los datos del código del contacto en la variable AHIVA, de tal manera que al entrar el foco de nuevo en la pantalla de mantenimiento se ejecutará en esa el evento CLOSECHILD y como la variable AHIVA lleva un valor hará lo indicado, que es mostrar dicho contacto.

Si por el contrario hemos pinchado sobre la fila 0, le indicamos que queremos clasificar por alguna columna, entonces capturamos la columna y hacemos un sort por el valor de dicha columna, y volvemos a recargar la tabla con los valores clasificados del fichero de SORT.

Control: **TABLA**
Evento : **DBCLICK**

```
Programación
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT ORDEN ASSIGN TO SORTWK01.
DATA DIVISION.
FILE SECTION.
SD ORDEN.
01 REGORD.
    02 ORDCOD PIC 9999.
    02 ORDNOM PIC X(30).
    02 ORDDOM PIC X(30).
    02 ORDPOS PIC 99999.
    02 ORDPOB PIC X(20).
    02 ORDPRO PIC X(15).
    02 ORDTIP PIC X(12).
    02 ORDTEL PIC X(20).
    02 ORDMOV PIC X(20).
    02 ORDMAI PIC X(30).
WORKING-STORAGE SECTION.
01 INDICE PIC S9(8) COMP-5.
01 FILA PIC 9999.
01 COLUMNA PIC 99.

PROCEDURE DIVISION.
    MOVE POW-CLICKROW OF TABLA TO FILA.
    MOVE POW-CLICKCOL OF TABLA TO COLUMNA.
    IF FILA NOT = 0 GO UNO.
    IF COLUMNA = 1 SORT ORDEN ON ASCENDING KEY ORDCOD
    USING CLASI OUTPUT PROCEDURE RELLENA ELSE
```

```

IF COLUMNA = 2 SORT ORDEN ON ASCENDING KEY ORDNOM
  USING CLASI OUTPUT PROCEDURE RELLENA ELSE
IF COLUMNA = 3 SORT ORDEN ON ASCENDING KEY ORDDOM
  USING CLASI OUTPUT PROCEDURE RELLENA ELSE
IF COLUMNA = 4 SORT ORDEN ON ASCENDING KEY ORDPOS
  USING CLASI OUTPUT PROCEDURE RELLENA ELSE
IF COLUMNA = 5 SORT ORDEN ON ASCENDING KEY ORDPOB
  USING CLASI OUTPUT PROCEDURE RELLENA ELSE
IF COLUMNA = 6 SORT ORDEN ON ASCENDING KEY ORDPRO
  USING CLASI OUTPUT PROCEDURE RELLENA ELSE
IF COLUMNA = 7 SORT ORDEN ON ASCENDING KEY ORDTIP
  USING CLASI OUTPUT PROCEDURE RELLENA ELSE
IF COLUMNA = 8 SORT ORDEN ON ASCENDING KEY ORDTEL
  USING CLASI OUTPUT PROCEDURE RELLENA ELSE
IF COLUMNA = 9 SORT ORDEN ON ASCENDING KEY ORDMOV
  USING CLASI OUTPUT PROCEDURE RELLENA ELSE
IF COLUMNA = 10 SORT ORDEN ON ASCENDING KEY ORDMAI
  USING CLASI OUTPUT PROCEDURE RELLENA ELSE
GO DOS.
EXIT PROGRAM.

UNO.
  MOVE FILA TO CCC.
  CALL GETCELLNUMERIC OF TABLA USING AHIVA FILA 1.
  CALL CLOSESHEET OF CONSULTA.
  EXIT PROGRAM.

DOS.
  EXIT PROGRAM.

RELLENA SECTION.
  MOVE 1 TO INDICE.

VER.
  RETURN ORDEN AT END GO TO FINVER.
  MOVE ORDCOD TO POW-NUMERIC(INDICE 1) OF TABLA.
  MOVE ORDNOM TO POW-TEXT(INDICE 2) OF TABLA.
  MOVE ORDDOM TO POW-TEXT(INDICE 3) OF TABLA.
  MOVE ORDPOS TO POW-NUMERIC(INDICE 4) OF TABLA.
  MOVE ORDPOB TO POW-TEXT(INDICE 5) OF TABLA.
  MOVE ORDPRO TO POW-TEXT(INDICE 6) OF TABLA.
  MOVE ORDTIP TO POW-TEXT(INDICE 7) OF TABLA.
  MOVE ORDTEL TO POW-TEXT(INDICE 8) OF TABLA.
  MOVE ORDMOV TO POW-TEXT(INDICE 9) OF TABLA.
  MOVE ORDMAI TO POW-TEXT(INDICE 10) OF TABLA.
  ADD 1 TO INDICE.
  GO VER.

FINVER.
  EXIT.

```

Con esto se da por concluida la ventana de consultas y su programación, como habéis podido comprobar no ha sido muy complicada, a partir de aquí podéis implementar todo lo que deseéis, mas campos, mas controles, cualquier cosa siempre es mejorable.

VENTANA ACERCA DE

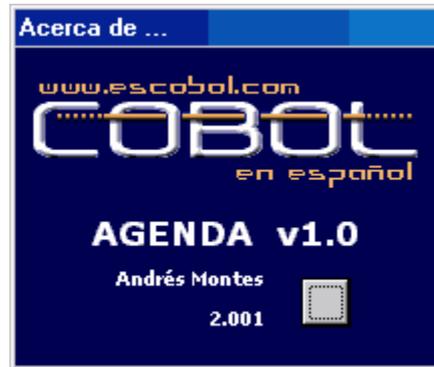
Por último vamos a crear una ventana con información sobre la aplicación, una ventana pequeña que solo nos mostrará datos sobre el programa y un botón para salir de ella, es la típica pantalla Acerca de... Esta pantalla se llamará desde la ventana principal.

Al igual que para crear la ventana de consulta, primero seleccionamos la opción **New**, del menú de **File**. En esta ventana vamos a colocar un Extend Image, donde colocamos el logotipo de la página que lo habréis bajado en el archivo totico.zip, que se llama cobol.bmp. También podéis colocar la imagen que vosotros queráis intentando que quede adaptada al tamaño de la ventana.

Además colocamos 2 labels indicando el nombre del programa, la versión y el autor y para finalizar un Bitmap Button, que será el que utilizemos para salir de la ventana.

Para el estilo de la ventana marcamos: Pop-up Windows, Thin y Title Bar. La llamamos ACERCA y en el título de la misma le ponemos Acerca de

El aspecto que debe de tener una vez la hayais completado debe de ser algo así.



El único evento que tendremos que programar será el del botón de salir y será el siguiente:

Control: **SALIR**
Evento : **CLICK**



El Bitmap Button, tendrá la imagen icook, marcado como resource y no como file. Y nos servirá para salir de ésta ventana.

Una vez concluido el diseño de la ventana, pinchamos sobre **Project**, **Edit** y a continuación pulsamos sobre el botón **Add**, para añadir esta ventana al proyecto y así poder compilar y ver como funciona. Una vez añadida pulsamos sobre el botón **Ok** y ya podemos ver en la pequeña ventana del proyecto que tenemos en la pantalla, como esa ventana está incluida.

FIN DEL CURSO

Con este capítulo se da por finalizado el curso de PowerCobol. Solo espero que os haya servido y que hayais aprendido a utilizarlo y sacarle provecho a la potencia de este compilador para Windows. A partir de ahora nuestras aplicaciones tendrán otra visión, otra perspectiva mas amigable, sencilla y por supuesto bonita.

Tengo que reconocer que el dejar para el final estos capítulos ha implicado que tuviera que retomar de nuevo el PowerCobol 3, ya que en la actualidad programo sobre la versión 5. Esto supongo que ha dado lugar a una explicación menos clara y efectiva que en los anteriores capítulos. Por lo tanto os pido un poco de mas atención para comprenderlo y por fin podáis terminar el proyecto.

Un saludo a todos los que hayais llegado hasta el final y hayais conseguido realizar el proyecto AGENDA en su totalidad. A partir de este momento estais preparados para embarcaros en los desarrollos que deseies con PowerCobol.