INICIO

Bien, antes de empezar me presento, soy MrRidk, y en este curso os tratare de iniciar en el mundillo de la programacion en delphi, espero que el curso os sirva de algo y aprendais algo, ya que para eso estamos aqui, te daras cuenta a lo largo del curso que no soy una persona a la que la guste demasiado enrrollarse, ya que tratare en todo momento de ser claro, conciso y de ir al grano pronto, y ahorrarnos parrafos inutiles que no sirven de nada.

El curso estara dividido en capitulos.

Para localizarme y hacerme cualquier pregunta, podras encontrarme en el canal del irc-hispano #Delphi_Aprendices y #Delphi. o bien puedes optar por mandarme un e-mail a esta direccion: ridk@hotmail.com.

Bien dicho esto espero que disfruteis del curso y que aprendais.

INDICE

TEORIA 1ª Parte.

En esta sección del curso vamos a conocer un poco el lenguaje de programación, lo básico para empezar a genera aplicaciones fáciles.

¿Que es un lenguaje de alto nivel y que es la orientacion a objetos?

En programación, nos encontraremos con 2 tipos de lenguajes, el de "Bajo" y el de "Alto" Nivel, se diferencian en lo proximos que esten al lenguaje maquina, entre los lenguajes de bajo nivel encontramos al Ensamblador, Delphi realmente no es un lenguaje, sino la IDE (Integrated Development Environment), osea el espacio de trabajo para Object Pascal, el cual es un lenguaje de programacion de alto nivel, Object Pascal como su nombre indica es Pascal orientado a objetos, un objeto es una coleccion independiente de estructuras, una clase alberga las estructuras de datos y rutinas de un objeto, en el caso de Delphi, los objetos se comunican mediante mensajes, asi pues cuando pulsamos con el raton sobre un boton, se produce un mensaje el cual desencadena un evento. Además delphi utiliza la RAD (Rapid Application Development), que nos permite desarrollar nuestros programas de forma rápida y visual. Delphi controla por si solo el sistema de mensajes de windows, con lo cual no necesitaremos preocuparnos de capturar los mensajes que dicta windows para desencadenar un procedimiento.

Los programas hechos en delphi se dividen en Formularios "Forms" que se corresponden con las distintas ventanas que tendra nuestro programa, para mostrar un formulario desde otro se ha de poner "Nombredelform.ShowModal" si quieres que tome el control y si quieres utilizar los dos indistintamente "Nombredelform.Show", ademas de los forms un programa esta compuesto por distintos objetos (Etiquetas, botones...) todos ellos estan incluidos en la VCL...

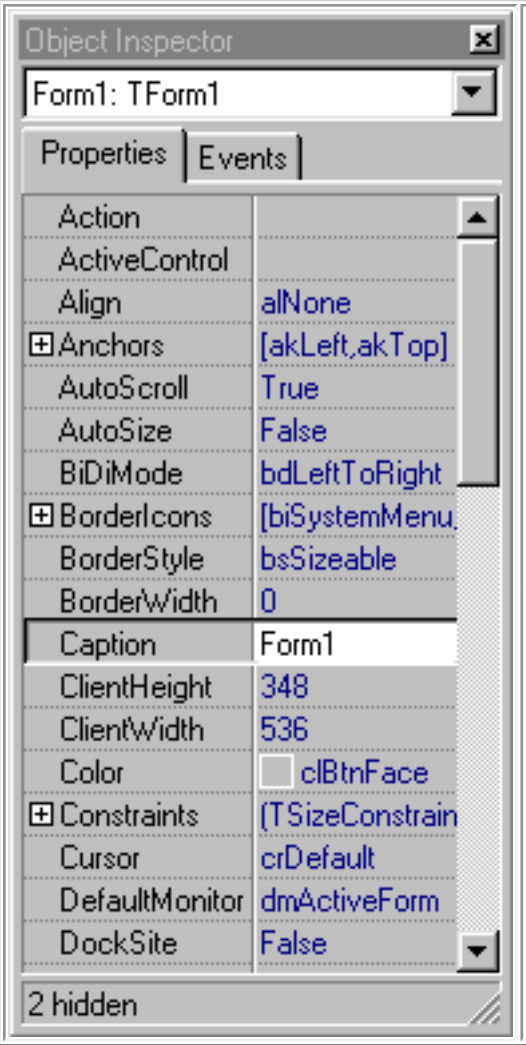
VCL

Delphi utiliza una libreria de componentes llamada VCL (Visual Components Library) que nos brinda una serie de objetos (por ejemplo una combobox) y que nos da la posibilidad de utilizar los objetos sin necesidad de llamar a las distintas API de windows para que dibujen en pantalla el componente, podremos ademas manipular a nuestro antojo el componente, cambiandole las propiedades como el nombre, el caption o cosas asi..., además de poder controlar de una manera muy facil los distintos eventos que podria generar. A la hora de trabajar con los componentes mediante codigo, la sintaxis es la siguiente:

1º El nombre del componente || 2º La propiedad con la que trabajaremos

Ej: Combobox1.Text; // Seleccionamos la propiedad Text del combo, que especifica el texto que contiene.

Object Inspector



Se trata del Inspector de Objetos, gracias a él podemos cambiar de una manera rapida facil y visual las propiedades de los distintos objetos que vayamos insertando en el programa. Ademas y gracias a él podemos controlar de forma fácil los distintos eventos, para ver los posibles eventos de un objeto selecciona la pestaña "Events", y para poder modificar el evento haz doble click sobre el, ahora apareceras en una especie de editor de textos, se trata de la ventana de edicion, y es donde se escribe el codigo de nuestro programa, ahora veras que estas situado entre dos palabras clave "Begin" y "End", y justo encima "procedure TForm1.FormCreate(Sender: TObject);". En delphi todas las lineas de comando acaban en ";" salvo algunas excepciones.

Edit Window

Tiene forma de editor de textos, en ella editaremos el codigo de nuestro programa, cuando editamos el codigo de un programa editamos el codigo de cada formulario, si nos centramos en un formulario veremos que en la edit window aparecen distintas secciones, cada una de ellas con distintos valores, estas secciones son:

Uses:

En ella especificamos los distintos modulos que cargara nuestro programa y que se incluiran en el ejecutable una vez compilado para que esté tenga una independencia total del compilador, cuando insertamos un componente se añade a ella la clave para que se carge el modulo que permite acceder a ese componente, asi por ejemplo si añadimos un componente de la pestaña "Estandar" al ejecutar el

programa se añade automaticamente la palabra "StdCtrls", pero hay otros casos en los que no se añade automaticamente, por ejemplo cuando queremos utilizar una determinada API de windows como puede ser ShellExecute(self.handle,'open','programa','parametros',nil,SW_ShowNormal); con la que conseguimos ejecutar una aplicacion externa desde delphi, deberemos pues añadir la palabra "ShellApi" para que nuestro programa pueda utilizar esa API

Type

En esta seccion se especifican los distintos objetos con su correspondiente clase (Label1:TLabel), y los distintos procedimientos que se ejecutaran en nuestro programa. Al final de ella veras 2 palabras clave "Private" y "Public", que sirven para definir variables privadas o publicas

A continuacion veras otra palabra clave "Var" que sirve para definir variables en cualquier sitio, si se pone en ese lugar, definiremos una variable que puede ser utilizada en todos los procedimientos del programa, si la definimos dentro de un procedimiento solo se podra usar en él.

Operadores aritmeticos, logicos y relacionales..

En este apartado del capitulo aprenderemos como identifica los operadores, delphi en relacion a otros lenguajes (VB Y C++), un operador sirve por ejemplo para hacer que el programa divida un nº entre otro.

Operador	Delphi	C++	Visual Basic
Menor que...	a < b	a < b	a < b
Mayor que...	a > b	a > b	a > b
Igual que...	a = b	a = b	a = b
Desigual...	a <> b	a != b	a <> b
Menor o igual...	a <= b	a <= b	a <= b
Mayor o igual...	a => b	a => b	a => b
Asignacion...	a := b	a == b	a = b
Comparacion...	a = b	a = b	a = b
Division...	a / b	a / b	a / b
Multiplicacion...	a * b	a * b	a * b

Comentarios.

En delphi como en cualquier lenguaje de programacion existe los comentarios, estos proporcionan la posibilidad de añadir a las lineas del programa, pequeñas anotaciones sobre como va el diseño del programa, por ejemplo podemos poner en una linea un comentario diciendo que es lo que sigue fallando para que al retomar el trabajo al dia siguiente sepamos por donde empezar, en delphi existen 2 tipos principales de comentarios. Y son:

Los que empiezan por "//" sirven para comentar una linea. y los que estan delimitados asi: "{}" Sirven

para poner un parrafo de comentario.

Bueno en el siguiente capitulo seguiremos con la teoria, pero no te preocupes si te parece demasiada, es que sino tienes unos conocimientos basicos no podras empezar a programar por ti mismo.

TEORIA 2ª Parte.

En este capitulo continuaremos con la parte de teoria del curso, asi pues sigamos conociendo un poco mas sobre el lenguaje de programación Object Pascal (Delphi).

Variables

Una pieza clave en todas las aplicaciones son las variables y el buen control de estas. Una variable puede contener varios tipos de datos y pueden ser modificadas en todo el programa si las declara como globales, o bien por todos los formularios (Si las declara publicas) o en un determinado evento (Si las declara en ese evento).

Para definir una variable publica has de remitirte a la seccion "Public" que encontraras abajo de "Type" y utilizar la siguiente sintaxis:

Variable : Tipodevariable;

Si lo que quiere es definir una constante que no vaya a variar en todo el programa utilice esta sintaxis:

Const Variable: Tipo = Valor;

Pero si lo que quiere es definir variables que puedan ser modificadas a lo largo del programa, deberá saber que en delphi existen varios tipos de variables, las más importantes son:

String	Cadena de texto
Integer	Número
Byte	Byte de un archivo
Boolean	Es una afirmación o una negación (True o False)
Date	Una Fecha
Time	Una hora
Char	Carácter.
Variant	La variable comodín, puede almacenar todo tipo de datos.

Convertir Variables

En delphi al contrario que en visual basic necesitaremos definir las variables y ademas utilizarlas solo con los datos apropiados, asi por ejemplo en una variable tipo integer (Nº Entero) no podemos situar el texto de un edit, aunque este edit solo contenga números, para permitir esto, deberemos convertir el contenido del edit de string a integer, y eso se hace con la estructura: StrToInt('Cadena');. A continuacion os pongo una tabla con las conversiones mas utilizadas...

IntToStr	Convierte un Integer a Texto
IntToHex	Convierte un Integer a Hexadecimal
StrToInt	Convierte una cadena de texto a Integer
StrToDate	Convierte texto en fecha
StrToTime	Convierte texto en fecha
DateToStr	Convierte una fecha a Texto
TimeToStr	Convierte la hora en texto

Propiedades de los objetos

Es importante conocer tanto los eventos propios de cada objeto como sus propiedades, por eso aquí os voy a mostrar una serie de propiedades generales que creo se adaptan a todos los objetos posibles.

Caption	Especifica el titulo del objeto o el texto que contendra
Cursor	Especifica el cursor que se mostrara cuando el ratón este sobre ese objeto
Default(Botones)	Especifica si ese botón Será el que se ejecute solo con dar un enter o no.
Enabled	Especifica si el objeto estará accesible al usuario
Font	Pos eso la fuente del texto de ese objeto
Height	El alto del objeto
Width	El ancho del objeto
Hint	El texto de explicación que mostrara cuando el ratón este sobre él
ShowHint	¿Mostramos o no la hint?
Visible	Especifica si Será visible o no.

Eventos

Un evento es la accion que se desencadena tras por ejemplo pulsar un boton, los eventos estan controlados por los mensajes que emite windows, asi por ejemplo cuando pulsamos un boton, se produce un mensaje, el cual es recibido e interpretado por nuestro programa que hace que se desencadene el evento al que hace referencia ese mensaje.

Algunos de los eventos mas comunes y presentes en casi todos los objetos son:

OnCreate	Es el que se produce al crear el objeto
OnCloseQuery	Es el que se produce al cerrar la aplicación
OnDestroy	Se produce al destruir el objeto
OnKeyPress	Se produce al presionar una tecla sobre el objeto en el que estemos

OnChange	Se produce al cambiar alguna cosa del objeto (propiedades, contenido....)
OnClick	Se produce al hacer click sobre el objeto.
OnEnter	Se produce al situar el foco en un objeto
OnExit	Se produce cuando el objeto pierde el foco

Aunque luego cada objeto contiene sus propias propiedades y sus propios eventos caracteristicos de las acciones que el componente realize.

Procedures y Funciones

Los procedimientos y las funciones son rutinas que se encargan de ejecutar una determinada accion, asi por ejemplo podemos crear un procedimiento que se encarge de elevar a mayusculas un texto, y luego llamarle para que convierta el texto que queramos.

```
Procedure ElevaMayusculas(Texto:String); // Definimos el procedimientoElevaMayusculas, con el argumento texto.
begin
Uppercase(texto); // Elevamos a mayusculas el argumento de la funcion (texto)
end;

{Y para llamarla en cualquier parte del programa haremos algo asi}

ElevaMayusculas ('mrridk'); // Elevamos a Mayusculas el texto 'mrridk'.
```

Este tipo de estructuras es util para realizar determinadas acciones que se ejecutaran una y otra vez a lo largo del programa.

Condiciones

En casi todos los programas se utilizan condiciones, es decir: "Si algo se cumple que haga algo, sino otra cosa";Su sintaxis es realmente sencilla (casi intuitiva):

```
If Condición then begin // La condición
Código // El código que ejecutara si se cumple la condición
end else // Sino se cumple la condición
Alternativo // Código que hace si la condición no se cumple
```

Otra forma de expresar las condiciones es: 'case of', lo que hace es seleccionar un identificador y ver que valores toma y en relacion a ello optar por hacer algo o hacer otra cosa, este tipo de condicion no acepta como identificador ni como valores a cadenas de texto, su sintaxis es la siguiente:

CASE OF

```
case Identificador of // Empezamos el bucle
1 : begin Showmessage ('Hola'); end; // En caso de que tome el valor 1, muestra un mensaje diciendo hola.
2 : begin Showmessage ('Adios'); end; // En caso de que tome el valor 2, muestra un mensaje diciendo adios.
end; // Acabamos
```

Bucles

Un bucle sirve para decirle al programa que ejecute cierta operación mientras una condición se cumple. Los tipos de bucles más importantes son: While, For, Repeat-UntilSus sintaxis son estas:

FOR

```
For I:= 0 to 100 do begin // Desde el contador (i) igual a 0 to 100 (si fuera downto iría para abajo)
Código; // El código que se ejecutara...
End; // Acabamos el bucle.
```

WHILE

```
While Condición do begin // Mientras se cumpla la condición...
Código; // El codigo que se ejecutara...
End; // Acabamos el bucle.
```

REPEAT-UNTIL

```
Repeat // Repite...
V := 1 + v; // Este codigo
Until // Hasta
v > 100; // Hasta que v > 100
```

Mensajes de informacion y/o Error.

En delphi existen basicamente 2 tipos de mensajes, el messagebox , que da una informacion y luego nos permite elegir que hacer, y el ShowMessage que unicamente muestra la informacion. Su sintaxis es la siguiente:

MessageBox:

```
Var
Button: Integer; // Definimos una variable que será la que albergue la respuesta del usuario
Begin // Comenzamos
Button:= Application.MessageBox ('Texto', 'Titulo', MB_YesNo + MB_IconInformation); // Creamos el mensaje...
If button = IDYes then begin // Si ha pulsado Yes tonces....
Codigo // El codigo...
end else // Sino...
If Button = IDNo then begin // SI ha pulsado NO tonces...
Codigo
end; // Acabamos.
```

ShowMessage:

```
ShowMessage('Texto');
```

Bien y con esto se acaba la seccion de teoria pura y dura, a partir de ahora trataremos de aprender mediante ejemplos practicos en los que pondre el codigo fuente con comentarios al margen para que queden bien explicados.

PRIMER PROGRAMA (¡¡¡ HOLA MUNDO !!!)

Como es costumbre, todos los cursos comienzan con un simple programa que muestra en pantalla el mensaje 'Hola Mundo', y este curso tambien hara algo de eso. Lo que trataremos sera de mostrar un mensaje en pantalla que diga Hola Mundo al pular un boton, como ves nada del otro mundo para un autentico genio como tú. :-D.

Como ya se comento en el 2º Capitulo de teoria en delphi existen basicamente 2 tipos de mensajes el Messagebox y el Showmessage, en este programa utilizaremos el ShowMessage.

Ahora has de pensar un poco, ¿Donde coloco el codigo?, bien, queremos que se muestre al pulsar un boton, y si te has leido la teoria, sabras que eso lo controla un evento el OnClick, asi pues vamos a editar el codigo de ese evento, vete al Object Inspector a la pestaña events, localiza el OnClick y da doble click sobre él, ahora estas en la Edit Window, entre un Begin y un End, como sabras todos los eventos y casi todas las estrucutras estan delimitadas entre un Begin y un End, cada begin requieree un end y cada end un begin, ahora vuelve a pensar ¿Cual es el codigo que debere poner? bien si te has hecho esa pregunta vuelve al capitulo de teoria donde se esplicaban los mensajes, el codigo es muy simple, la estructura de un ShowMessage solo que como argumento pondremos ¡¡¡ Hola, Mundo !!!, asi pues nuestro primer programa se reduce a esta linea:

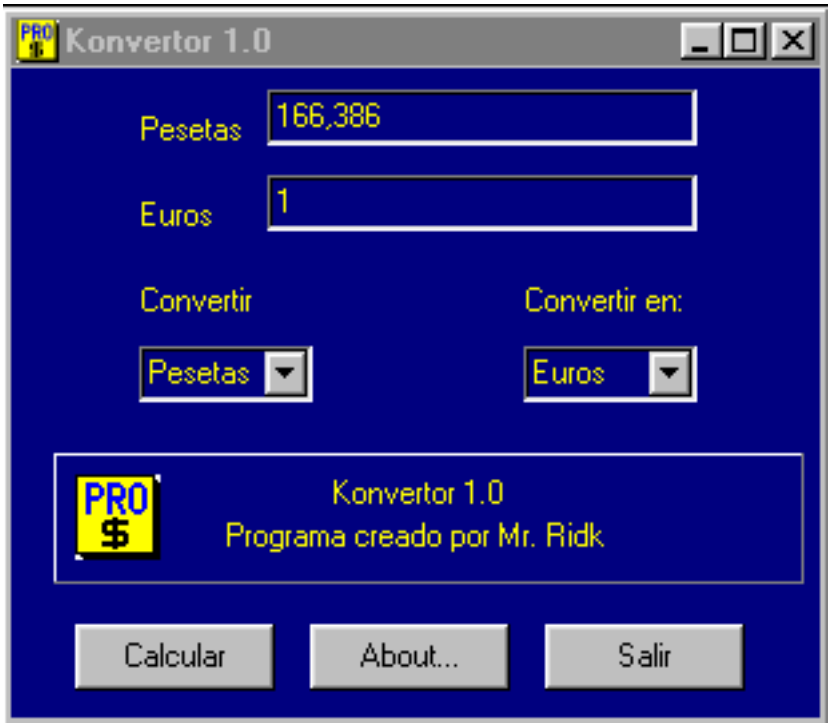
```
Showmessage ('¡¡¡ Hola, Mundo !!!');
```

Y con eso ya hemos acabado, ahora ejecuta el programa, dale al boton y observa el mensaje. Ala!!! ya eres un programador de verdad, ahora si quieres mejorar continua con el curso...

KONVERTOR

En este capitulo haremos nuestro primer programa serio, se tratara de un konvertor de monedas, que trabajar con 3 monedas entre si, utilizaremos la condicion case of, ya que para este tipo de cosas es lo más facil y rapido.

Antes que nada diseña una interface como esta:



Bien como ya especifique en el 1er capitulo de la teoria, la condicion case of es facil y rapida, ya que hay que escoger un identificador en este caso el index del combobox, que especifica que item esta seleccionado y ver que valor tiene en un determinado momento y en base a ese valor, elaborar una respuesta u otra.

Asi pues un posible codigo fuente podria ser el siguiente:

```
var
valor : variant; // Definimos la variable que contendra el valor del edit
begin
valor := Edit1.text; // Asignamos el valor que el usuario a metido para convertir.
case Combobox1.ItemIndex of // Empezamos a ver que queremos convertir
0 : Begin // SI es pesetas, ya que el item pesetas tiene el valor index = 0, pq es el
1er item de la lista
Case Combobox2.ItemIndex of // Empezamos a ver en que quermos convertir
0 : Begin Showmessage('Escoga Monedas Distintas'); end; // Ya que intentamos
convertir pesetas en pesetas damos error
1 : begin Edit2.text := valor * '0,00601'; end; // Convertimos en euros (1)
2 : begin edit2.text := valor * '0,00532'; end; // Convertimos en Dolares (2)
```

```
end; // Acabamos con el de convertir pesetas
1 : begin // Empezamos con el de convertir euros
case combobox2.itemindex of // Empezamos a ver en que queremos convertir
0 : begin .... end; // Convertimos en pesetas
Y bueno lo demas lo dejo de tu cuenta, la estructura supongo que estar bien
esplicada asi que el resto tomatelo como un ejercicio el acabarlo
```

Bueno, nuestro programa convierte a la perfeccion, pero que ocurre en el momento de cambiar el item seleccionado de una comobox? Nada !!, ahi esta la cuestion, que no ocurre nada, deberia cambiar las edit que muestran que convertir y en que convertir, pero de momento eso no ocurre. Bien, ahora piensa un poco ¿Cuando deben de cambiar? efectivamente al cambiar el contenido del combobox, asi pues en el evento OnChange del Combobox, y ¿Que codigo debemos poner?, un codigo que ponga como caption del label el contenido del item seleccionado y eso se consigue de la siguiente manera:

```
Label1.Caption:= ComboBox1.Items[ComboBox1.ItemIndex]; // El texto
de la label1 será el del item seleccionado de la combobox1
```

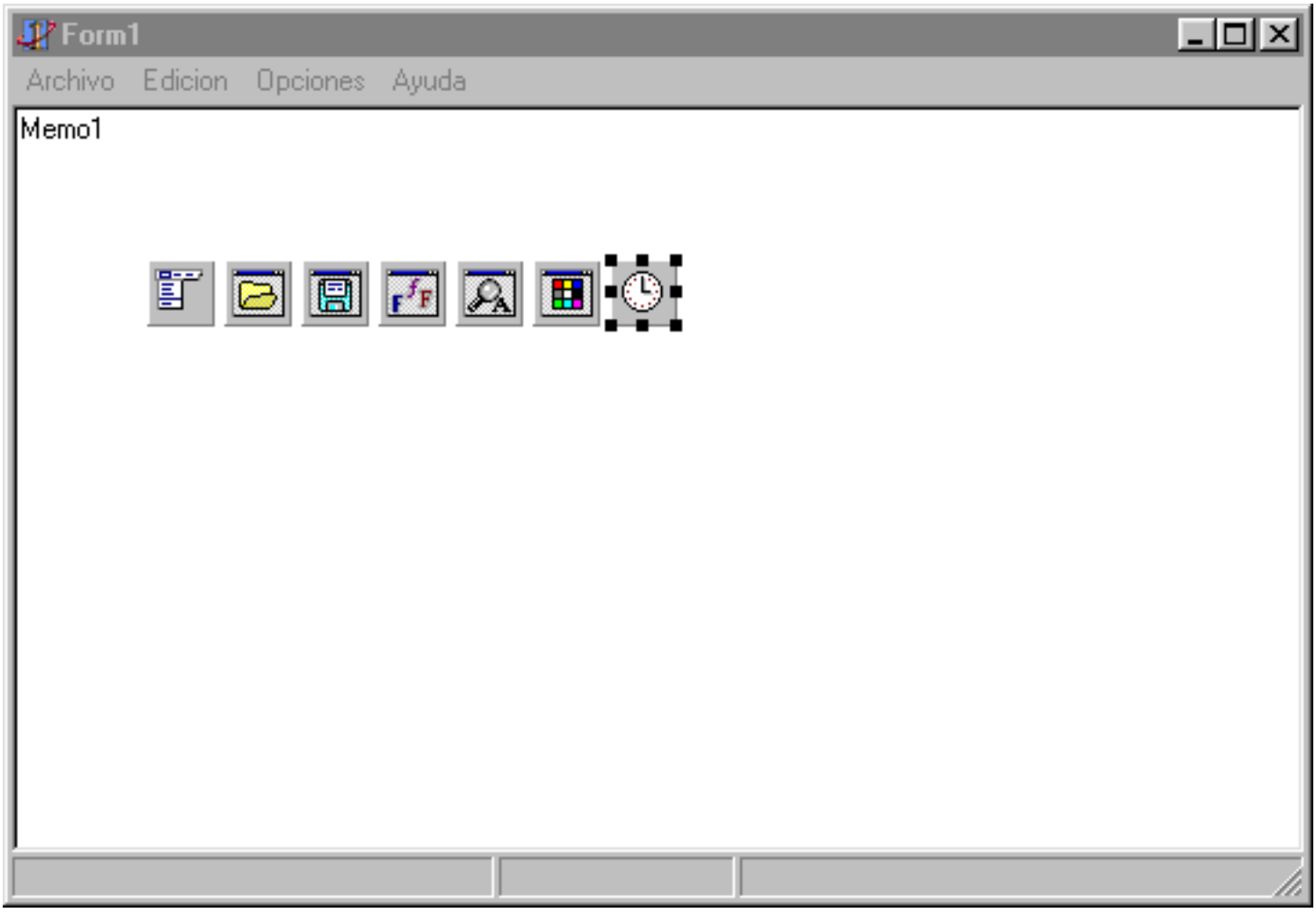
Y ya esta, ahora sigamos con los siguientes botones, en el de salir, vas a poner "Close" que lo que hace es llamar al evento OnCloseQuery del form, osea al que cierra la aplicacion.

Y con esto y un bizcocho hasta el proximo capitulo (jeje, que mal se me dan las rimas :-?).

EDITOR DE TEXTOS 1ª PARTE

En esta serie de 2 capitulos, aprenderemos como crear paso a paso un editor de textos facilito, como el notepad, y con las opciones basicas requeridas en un editor de textos. Nuestro editor de textos utilizara los cuadros de dialogo estandares de windows, es decir el cuadro predeterminado de abrir un archivo o el de buscar un texto, que estan presentes en casi todos los programas. Los encontraras en la paleta "Dialogs", y para ejecutarlos se utiliza la propiedad ".Execute" aunque ya veremos como se utiliza cada uno...

Bien, dicho esto diseña una interface como esta:



El TMemo lo queremos por que es él el encargado de contener la parte del programa donde editaremos el texto y demás..El MainMenu lo queremos para dar la posibilidad de tener un menu donde estarán las opciones que puede hacer el usuario.Antes de comenzar con el codigo he de decirte que existen una serie de propiedades en el Memo que indican la posicion en la que esta el cursor (SelStart), la cantidad de texto seleccionado (SelLength) la cantidad de texto (LenText).Bien empecemos con la opción de abrir un archivo, para ello utilizaremos el dialog OpenDialog y el código es el siguiente

```
If OpenDialog1.execute then begin // Si se ha ejecutado correctamente
tonces
Memo1.Lines.LoadFromFile (Opendialog1.Filename); // El texto del
Memo lo carga del archivo del Opendialog
end; // Se acabo
```

Ahora con la opción de guardar el archivo, como veras dentro un rato es prácticamente igual.

```
If SaveDialog1.execute then begin // Si se ha ejecutado correctamente
tonces
Memo1.Lines.SaveToFile (Savedialog1.Filename); // El texto del Memo lo
guarda al archivo del Savedialog
end; // Se acabo
```

Sigamos con el menu del Archivo, en todo Editor de textos ha de haber la propiedad de crear uno nuevo ¿no? bueno pos con delphi es así de fácil:

```
Memo1.Clear;
```

Que fácil ¿Verdad? bueno sigamos ahora con la opción de imprimir,

```
Memo1.Print ('My
Document');
```

Sigamos con el código del programa la opción de deshacer:

```
Memo1.Undo;
```

Ahora con las opciones de cortar, copiar, pegar y seleccionar todo:

```
Memo1.CutToClipBoard; //
Corta
Memo1.CopyToClipBoard; //
Copia
Memo1.PasteFromFile; // Pega
Memo1.SelectAll; // Selecciona
Todo
```

Muy bien ya tenemos el menu de Edicion al completo, pero claro los botones de Copiar Cortar y Pegar siempre están asequibles aunque no haya texto seleccionado, ¿Cómo evitarlo? En el OnClick del Item Edicion del Menu pones este código..

```
var
selección : Boolean; // Definimos una variable booleana (si o no)
begin
if Memo1.Sellength <> 0 then begin // Si hay algo seleccionado
Seleccion := true; // La variable se convierte en true
end else // sino
Seleccion := false; // La variable es false
Cortar1.Enabled := Seleccion = true; // Los items estan disponibles cuando la variable es true
Copiar1.Enabled := Seleccion = true; // Los items estan disponibles cuando la variable es true
Eliminar.Enabled := Seleccion = true; // Los items estan disponibles cuando la variable es true
end; // Acabamos
```

Bueno y con esto doy por terminado este capitulo, ¿Impaciente por saber como acabara? pos eso lo sabremos en el siguiente capitulo...

EDITOR DE TEXTOS 2ª PARTE

En esta segunda parte acabaremos de rematar el editor de textos, aunque casi todo el trabajo ya esta hecho... Empecemos con el menú de opciones. En el Onclick del item Buscar pon esto:

```
FindDialog1.Execute;
```

Demasiado fácil ¿no?, bueno pos ahora tendrás que controlar el evento OnFind del FindDialog y poner este código:

```
var
FoundAt: LongInt; // Definimios la variable de donde se ha encontrado
StartPos, ToEnd: Integer; // Definimos las variables de comienzo y final..
begin
with Memo1 do // Cogemos el Memo para trabajar con el
begin
if SelLength <> 0 then // Si no estamos al principio del texto..

StartPos := SelStart + SelLength; // Comenzamos desde donde estemos
else // sino

StartPos := 0; // Comenzamos al principio

ToEnd := Length(Text) - StartPos; // Definimos la longitud desde el
comienzo hasta el final

FoundAt := FindText(FindDialog1.FindText, StartPos, ToEnd,
[stMatchCase]); // Buscamos el texto con los datos que hemos calculado
if FoundAt <> -1 then // Si hemos encontrado algo...
begin
SetFocus; // Colocamos el cursor alli
SelStart := FoundAt; // Comenzamos a seleccionar donde lo hay
encontrado
SelLength := Length(FindDialog1.FindText); // Y seleccionamos la
longitud del texto a buscar
end; end; end; // Acabamos.
```

Bien una vez solucionado y entendido el comando de buscar pasamos a la opción de cambiar la fuente.Como se trata de trabajar con un dialogo de Windows ya veras que fácil e intuitivo es esto:

```
If FontDialog1.Execute then begin // SI se ha ejecutado correctamente el dialogo

Memo1.Font := FontDialog1.Font; // Cambiamos la fuente

end; // Acabamos
```

Y ahora el del color del fondo del Memo, como vas a ver es casi igual...:

```
If ColorDialog1.Execute then begin // SI se ha ejecutado correctamente el dialogo

Memo1.Color := ColorDialog1.Color; // Cambiamos el color

end; // Acabamos
```

Bueno y con eso acabamos del todo el menu de opciones. ¿Y ahora? pos ahora vamos con la ayuda que nos dará la posibilidad de ver un formulario nuevo(el típico about).Para ello en el OnClick pones esto:

```
AboutBox.ShowModal; // Mostramos de una forma 'modal' la ventana del about
```

Y con esto ya hemos acabado el editor de textos, veras como preferiras usar el tuyo antes que el de micro$oft...

INIFILES (*.INI)

¿Que es un archivo INI?

Un archivo INI es un archivo especial que se caracteriza por estar dividido en secciones, lo cual proporciona la posibilidad de acceder rápida y fácilmente a una sección del archivo.Generalmente se usan para guardar configuraciones de programas, por ejemplo conque fuente fue cerrado el editor de textos, para que al arrancar de nuevo el programa, este se ejecute con esa fuente y no con la predeterminada.Para poder trabajar con archivos INI en delphi deberás añadir "IniFiles" a la cláusula uses, con ello conseguirás que tu programa pueda trabajar con archivos INI. La sintaxis es básicamente esta:

```
var
MiFichero: Tinifile; // Definimos el nombre de la variable...
begin
MiFichero := Tinifile.create ('C:\Configuracion.ini'); // Creamos el fichero y se lo asignamos a la variable.
.... // Aquí iría el código en donde le diríamos al programa que hacer con ese archivo
MiFichero.Free; // Liberamos el Archivo
End; // Acabamos
```

En un archivo INI se puede hacer básicamente dos cosas : Leer y escribir.

```
Para leer:

Variable := MiFichero.ReadInteger('Sección', 'Valor', Valornumerico);

Para escribir

MiFichero.WriteInteger('Sección', 'Valor', ValorNumerico);
```

La sintaxis para leer o escribir depende del dato que vayamos a leer, así por ejemplo os pongo una tabla con los más habituales

MiFichero.ReadBool	Lee una expresión booleana
MiFichero.ReadString	Lee una cadena de texto
MiFichero.ReadInteger	Lee un dato numerico
MiFichero.WriteBool	Escribe una expresión booleana
MiFichero.WriteString	Escribe una cadena de texto
MiFichero.WriteInteger	Escribe un dato numérico

Habitualmente este tipo de código se pone en el OnCloseQuery y en el OnCreate de los formularios, para que el programa se ejecute con la configuración anterior, y se cierre guardando la configuración actual. Así pues os voy a poner un poco de tarea.

Bien, y con esto se acaba este capitulo dedicado a los archivos Ini, en el siguiente aprenderemos un poquito sobre el registro de windows.

REGISTRO DE WINDOWS

Este capitulo esta dedicado a Windows y a su registro, en él aprenderemos que es y para que sirve el registro de windows.

El registro de windows es el lugar donde windows y otros programas que funcionan bajo windows guardan su configuración, o cualquier otra cosa que sea necesaria para el correcto funcionamiento del programa, asi por ejemplo cuando asociamos una extension a un ejecutable, no estamos, nada mas que accediendo al registro y escribiendo en él para que cuando se ejecute un archivo con una determinada extension se ejecute el programa asociado y no otro.

El registro de windows esta dividido en 6 secciones que son:

HKEY_CLASSES_ROOT
HKEY_CURRENT_USER
HKEY_LOCAL_MACHINE
HKEY_USERS
HKEY_CURRENT_CONFIG
HKEY_DYN_DATA

Cada seccion esta devidida en claves y estas a su vez en otras claves, en el registro se pueden almacenar todo tipos de datos, desde la fuente con la que se cerro un editor de textos hasta el tipo de impresora y demas.

Bien para poder manejar el registro en windows, necesitaras añadir "Registry" a la clausula uses.

En el registro se pueden hacer dos cosas, leer y escribir (ademas de borrar).

Antes de poder hacer nada con el registro se debe de asignar a una variable haciendo lo siguiente:

```
var
reg : Tregistry; // Definimos la variable
begin
reg := Tregistry.create; // Asignamos el registro a la variable
with reg do begin // Cogemos el registro para trabajar con él
... // El codigo
Reg.free; // Liberamos el registro.
```

Para especificarle la seccion a la que queremos que se diriga utilizaremos el comando "RootKey := HKEY_CLASSES_ROOT" por ejemplo.

Una vez alli se puden hacer 2 cosas como ya dije antes, leer o escribir, pero antes de eso hay que abrir

las distintas claves.

un posible codigo de ejemplo podria ser el siguiente:

```
var
reg : Tregistry; // Definimos la variable
begin
reg := Tregistry.create; // Asignamos el registro a la variable
with reg do begin // Cogemos el registro para trabajar con él
RooTKey := HKEY_CLASSES_ROOT; // Nos vamos a
HKEY_CLASSES_ROOT

OpenKey ('.nfo', true); // Abrimos la clave

WriteString('', 'Archivo de texto'); // Escribimos el tipo de fichero que es

CloseKey;

{Antes de liberar el registro debemos notificarle que ha sido modificado}

SHChangeNotify( SHCNE_ASSOCCHANGED,SHCNF_IDLIST, nil,
nil);

{Una vez hecho esto liberamos el registro}
Reg.free; // Liberamos el registro.
```

Bueno y con esto y un bizcocho hasta el siguiente capitulo ;-).

BASES DE DATOS (1ª PARTE - TEORIA)

En este capitulo aprenderemos para que sirven y como se trabaja con bases de datos en delphi.Antes que nada decir que una base de datos es un sistema de almacenamiento de datos de una forma estructurada, se divide en Campos (Columnas) y en registros (Filas). Los Campos son las secciones en las que estará dividida la información que contiene la Base de datos, y los registros son los datos que contiene la base de datos.Para crear una tabla podremos usar el Database Desktop que viene con delphi, y que nos proporciona una manera rápida y sencilla de generar tablas, para luego manipularlas en nuestros programas. Algunos comandos interesentes en el manejo de BD son estos:

Table1.Open;	Abre la tabla
Table1.Insert;	Ponemos a la tabla en modo insercion
Tabel1.Post;	Guarda los cambios que se han producido en una tabla
Table1.Delete;	Borra el registro seleccionado
Table1.Cancel;	Cancela las modificaciones que se hayan producido.
Table1.Flushbuffers;	Libera la tabla de la memoria.

Algunos tipos de campos más usados podrían ser estos:

FtString	Cadena de texto
FtInteger	Numero
FtDateTime	Fecha y Hora
FtBoolean	Campo con una expresion booleana
FtTime	Hora
FtDate	Fecha
FtAutoinc	Campo autoincrementable
FtMemo	Campo en el que puedes situar el texto de un memo
FtGraphic	Imagen
FtVariant	Puedes meter cualquier dato, ya que esta sin determinar.

Bien, para trabajar con registros de determinados campos utilizaremos la sentencia "Table1.FieldbyName('NombreCampo').as(tipodedato) := Dato;"Con FieldByName utilizamos un determinado campo, as(tipodedato) por ejemplo asstring sirve para especificar el tipo de dato que queremos insertar, y el dato es el valor que vamos a insertar.Un ejemplo de esto seria el siguiente:

```
Table1.Insert; // Preparamos la tabla para poder insertar
Table1.FieldbyName('Nombre').Asstring := 'MrRidk'; // Insertamos el dato 'MrRidk' en el ultimo registro del campo 'Nombre'
Table1.FieldbyName('Web').AsString := 'Http://welcome.to/mrridk'; // Insertamos el dato 'Http://welcome.to/mrridk en el ultimo registro del campo 'Web'
Table1.Post; // Guardamos los cambios.
```

Obviamente los campos han de existir ya que sino se producira una exception, y también el dato a meter deberá corresponder con el tipo de campo al que se lo queramos insertar. Por ejemplo no podemos insertar un 'String' en un campo tipo 'Integer'.Otra de las cosas que mas ventajas tiene de trabajar con tablas es la posibilidad de filtrar una tabla en base a unos datos, para que solo se muestren los registros con esas características.Para filtrar una tabla la propiedad de 'Filtered' ha de estar en 'True', y para que se vuelvan a mostrar todos los registros pondremos esa misma propiedad a 'False'. Antes de filtrar o no filtrar la tabla, deberemos darle los datos en los que se va a basar para filtrar, eso se hace con la propiedad 'Filter', de esta manera.

```
Table1.Filter := 'Campo = '" + Dato +"';; // Recuerda que el dato debera de estar acorde con el tipo de campo...
Table1.Filtered := True; // Filtramos
```

Bien con eso conseguimos filtrar la tabla en base a un dato que nosotros especificamos en tiempo de diseño (es decir cuando programamos). Pero verdad que eso tiene escasa utilidad, la ventaja de filtrar una tabla es poder elegir en cada momento en base a que filtrar, por ejemplo filtrar en base al texto de un Edit. Bien pues para conseguir esto se utiliza esto:

```
Table1.Filter := 'Campo = '" + Edit1.text + "'; // Filtramos en base a lo que hay en el edit.

Table1.Filtered := True; // Filtramos
```

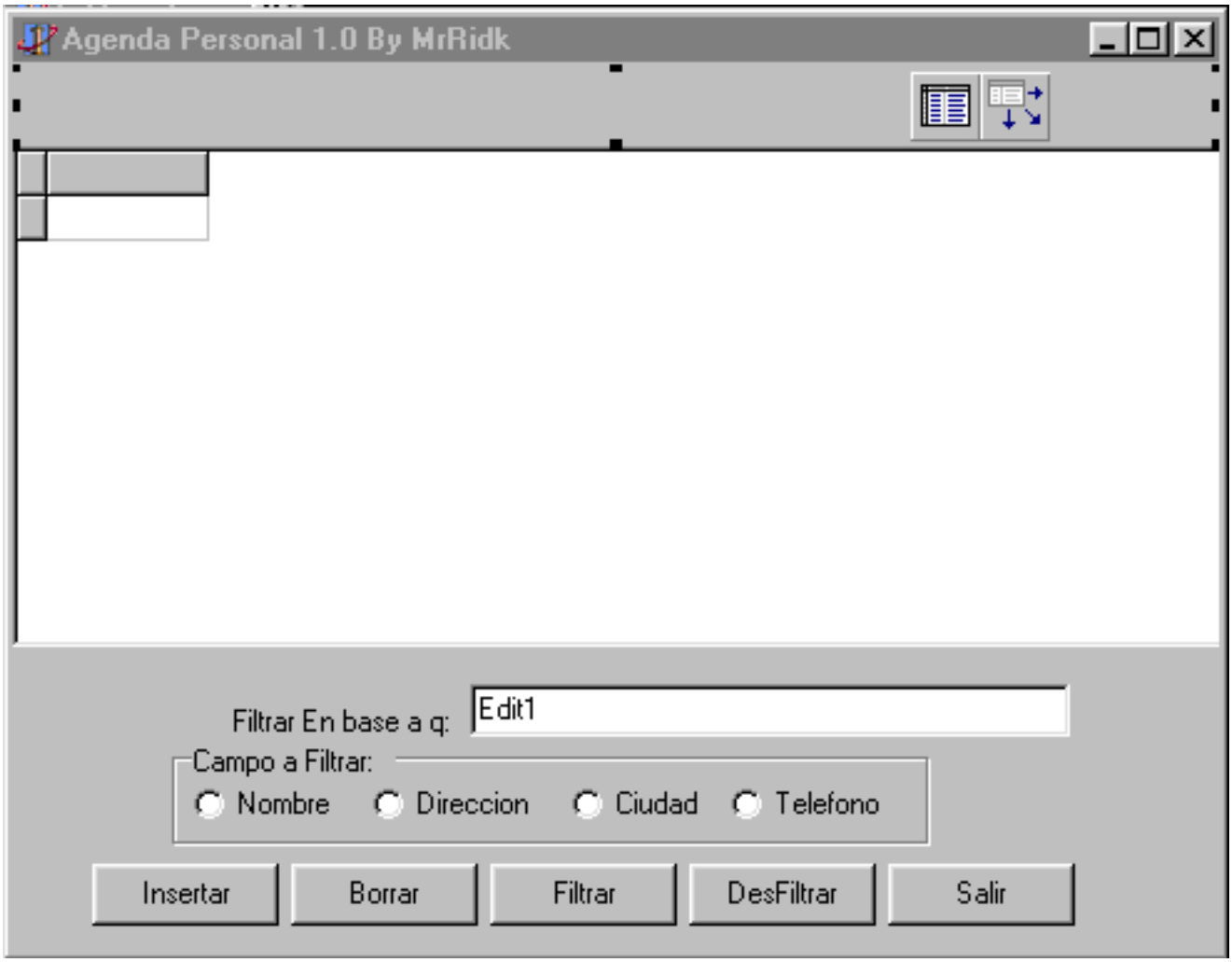
Bien ya tenemos nuestra tabla filtrada en base a lo que queríamos, pero ¿Como vuelvo a mostrar todos los campos?, pues muy sencillo, si para filtrar ponías la propiedad 'Filtered' a 'True' para quitar el filtro la deberás poner a 'False' (Table1.Filtered := False);Otra opción interesante que nos brindan las bases de datos es la posibilidad de buscar un determinado registro, con la opción de 'FindKey', que busca en un campo un registro y situa el foco allí donde este el valor.Aunque para poder usarla deberás tener definido el Index por el cual ha de buscar, así por ejemplo si queremos buscar por el campo 'Nombre' y su index es: 'ByNombre' pondremos esto:

```
Table1.IndexName := 'ByNombre'; // Situamos el index 'ByNombre' como predefinido

Table1.Findkey (['MrRidk']); // Buscamos MrRidk en el campo Nombre

end; // Acabamos.
```

Bueno pues con esto doy por terminado el capitulo dedicado a la teoria de Base de datos, en el siguiente capitulo veremos un caso practico de utilización de base de datos, se tratara de una agenda Personal.

BASES DE DATOS (2ª PARTE - AGENDA)

En este capitulo os voy a mostrar una forma sencilla de generar una agenda personal con bases de datos, antes que nada genera una interface que conste de: "Table" "DataSource" "DBGrid" "5 Botones" "4 RadioButton" "1Edit". Como la siguiente



Y empecemos a variar las propiedades del datasource, de la table y del DBGrid, para que se consiga lo que deseamos

Empecemos con la Table:

DatabaseName	Hay deberás poner la Path de la tabla sin el nombre de la tabla
TableName	Hay deberás poner le Nombre de la tabla sin la Path

Ahora vayamos con el DataSource

DataSet	Hay deberás poner el nombre del componente Table

Y para acabar con el DBGrid:

DataSource	El nombre del componente DataSource

Y ya esta con eso hemos acabado, por ahora...

Ahora vamos a empezar con los primeros códigos, primero diseña otro formulario con 4 Edit y 4 Label y 2 Botones, algo así como este:

¿Qué para que? Pues bien este formulario se encargara de insertar de una manera fiable los datos a la tabla, para mostrarlo en el botón de 'Insertar' del formulario principal debes de poner el código correspondiente, ¿Qué no sabes cuál es? Pues deberías, así que mira unas lecciones mas para atrás.

Y en el botón de Insertar del Formulario de Insertar Registro pones este código:

```
Form1.Table1.Insert; // Ponemos la tabla en modo insercion
Form1.Table1.FieldbyName('Nombre').AsString := Edit1.text; //
Insertamos el nuevo nombre
Form1.Table1.FieldbyName('Direccion').AsString := Edit2.text; //
Insertamos su direccion
Form1.Table1.FieldbyName('Ciudad').AsString := Edit3.text; // Insertamos
su ciudad
Form1.Table1.FieldbyName('Telefono').AsString := Edit4.text; //
Insertamos su telefono
Form1.Table1.Post; // Guardamos los cambios.
```

Y en el boton salir el comando 'Close' para cerrar ese formulario y volver al principal.

Bien ya tenemos la opción de insertar un nuevo contacto en nuestra agenda terminada, ahora falta lo más fácil. Empezaremos con el boton borrar registro. En él nada mas deberás poner este código:

```
Table1.Delete; // Borramos el
Registro activo
```

y ahora vamos con el más complejo de todos, la opción de filtrar, pero no te preocupes que te voy a explicar paso a paso que hago para conseguir que esta función funcione:

```
If Radiobutton1.Checked = true then begin // Si hemos seleccionado el
boton de filtrar por el nombre...
Table1.Filter := 'Nombre = '" + Edit1.text + ""; // Filtramos con lo que
ponga en el edit por el campo del nombre
end else // Sino
if Radiobutton2.checked = true then begin // Si hemos seleccionado el
boton de filtrar por la direccion...
Table1.Filter := 'Direccion = '" + Edit1.text + ""; // Filtramos con lo que
ponga en el edit por el campo de la direccion.
end else // sino
If Radiobutton3.checked = true then begin // Si hemos seleccionado el
boton de filtrar por la ciudad...
Table1.Filter := 'Ciudad = '" + Edit1.text + ""; // Filtramos con lo que ponga
en el edit por el campo de la ciudad
end else // sino
If Radiobutton4.checked = true then begin // SI hemos seleccionado el
boton de filtrar por el telefono...
Table1.Filter := 'Telefono = '" + Edit1.text + ""; // Filtramos con lo que
ponga en el edit por el campo de la ciudad
end; // Acabamos con las opciones
Table1.Filtered := True; // Y filtramos la tabla
end; // Acabamos con el boton.
```

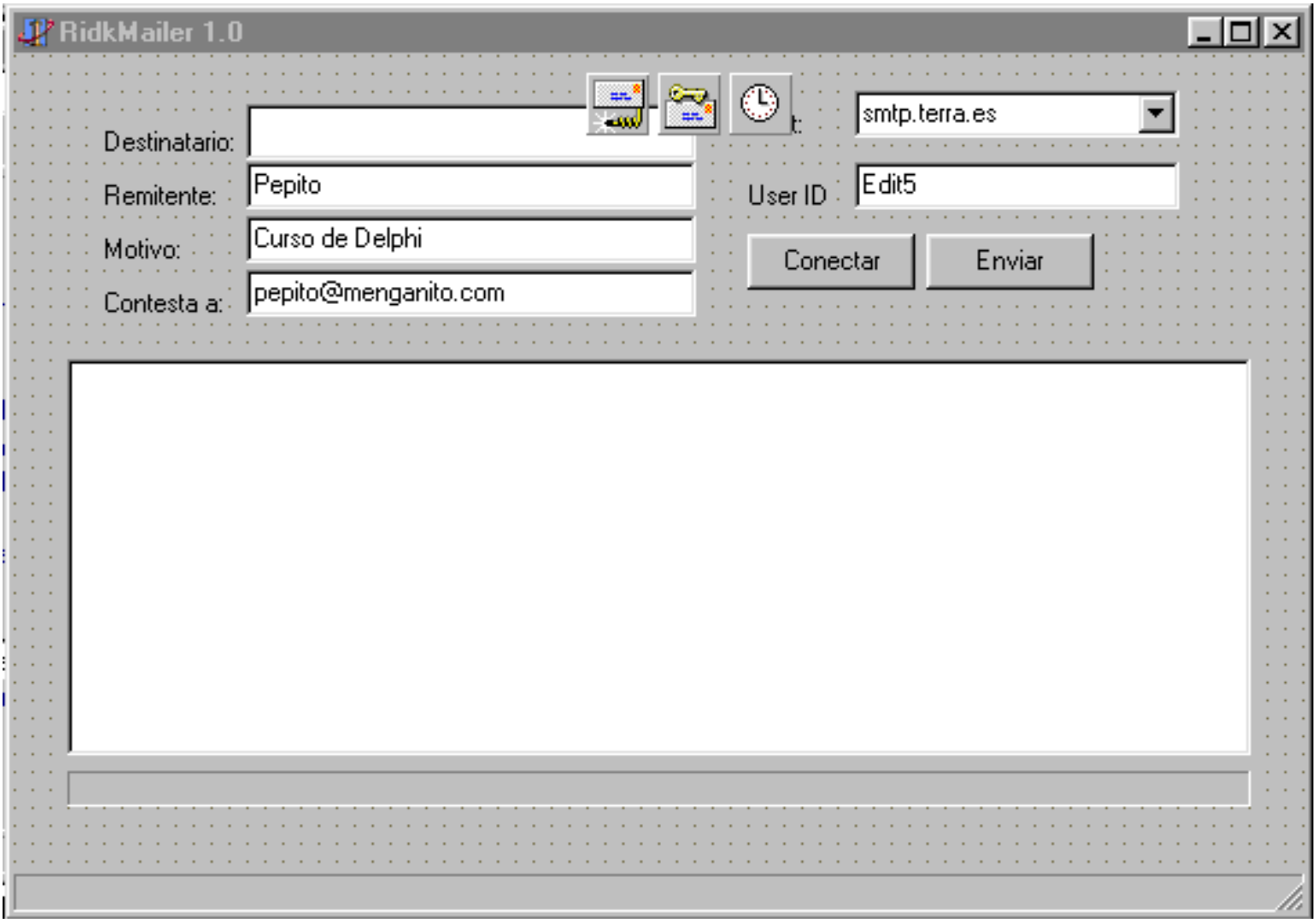
Y para desfiltrar la tabla como ya te dije en el capitulo anterior solo has de poner: "Table1.Filtered := False". Y con esto acabamos nuestra agenda personal ¿Verdad que ha sido fácil?.

FASTNET (MAILER).

FastNet es una paleta de componentes que nos dan la posibilidad de acceder rapidamente a distintos servicios de Internet como son Smtp, Pop3...

En este capitulo nos aproximaremos a un componente bastante util de FastNet, es el NMSMTP que nos permite mandar e-mails a traves de servidores gratuitos (smtp.terra.es...) o a traves del nuestro propio insertando la ID

Para empezar diseña una interface con un ControlPage, y dos paginas, una para mandar y otra para recibir, algo asi:



Bien, para la pagina de mandar e-mail utilizaremos: 2 Memos(El uno para especificar a donde mandar y el otro para especificar el mensaje en si), 4 Edit (Remitente, Motivo, Direccion del remitente, User ID), 1 Combobox (Contendra los distintos host) 2 Botones (Conectar, y enviar).

Bien una vez diseñada la interface empecemos con las propiedades de los componentes, para dar los datos al Smtp, utilizaremos el PostMessage, asi por ejemplo para darle la direccion quedara algo así : "NMSmtp1.PostMessage.ToAddress.Assign(Memo2.Lines);". Bueno el codigo fuente del programa seria algo asi:

```
{Boton de conectar}
Smtp1.Host := Combobox1.Host; // le especificamos a donde conectar
                              // Decimos la Identidad del mandante
Smtp1.Connect; // Conectamos
end; // Acabamos el procedimiento


{Boton de Enviar}
Smtp1.PostMessage.ToAdress.Assign(Memo2.Lines); // Decimos a donde
enviarlo
Smtp1.PostMessage.FromName := Edit1.text; // Decimos el remitente
Smtp1.PostMessage.Subject := Edit2.text; // Decimos el motivo del
mensaje
Smtp1.PostMessage.FromAddress := Edit3.text; // Decimos la direccion de
e-mail del remitente
Smtp1.PostMessage.Body.Assign(Memo1.Lines); // Asignamos las lineas
del mensaje
Smtp1.Send; // Mandamos el mensaje
end; // Acabamos el procedimiento


{Ahora empezamos con los distintos eventos del Smtp}
{Evento OnConnect}
Statusbar1.Panels[0].Text := 'Conectado a: ' + Smtp1.Host; // En la barra
de estado decimos que ya estamos conectados


.
{Evento OnConnectFailed}
ShowMessage ('No se ha podido establecer la conexion con el servidor'); //
Si algo falla en la conexion damos el correspondiente error.


{Evento OnInvalidHost}
ShowMessage('El servidor especificado no es valido'); // Si ha especificado
un servidor invalido damos el correspondiente error.


{Evento onSuccess}
Smtp1.Disconnect; // Cuando le ha enviado satisfactoriamente
desconectamos.
```

Bien con esto hemos completado el programa para enviar e-mails, y tambien hemos acabado este capitulo dedicado a FastNet espero haberlo dejado todo bien explicado, de no ser asi ya sabeis mandadme un e-mail a: ridk@hotmail.com

LINKS

Bien con esto doy por terminado mi cursillo sobre el lenguaje de programacion delphi, en el que he intentado acercarte al mundo de la programacion en delphi, es un curso dedicado para novatos asi que si lo has visto un poco simple es por que tu ya dejaste de ser novato :-).

Para hacer programas realmente serios no tengas miedo a insertar nuevos componentes, y como decia el gran sabio Einstein "La imaginacion es mas importante que la cultura". Asi que ya sabes se innovador y conseguiras ser un gran programador.

A continuacion te pongo una serie de links relaccionados con la programacion en delphi y que espero te sirvan de algo.

LINKS

Http://welcome.to/mrridk	Mi pagina web, encontraras sources, trucos y demas...
Http://www.vclcrawler.com	Pagina con una gran recopilacion de componentes delphi
Http://www.torry.net	La gran pagina de los componentes de delphi
Http://www.clubdelphi.com	La mejor pagina para iniciarse
Http://www.q3.nu/trucomania	La gran pagina dedicada a los trucos, encontraras de todo...
Http://www.clubdelphi.com/saiyine	La pagina personal de uno de los que me ayudaron en mis comienzos
http://www.geocities.com/abdaleon	La pagina de otro de los que me ayudaron.
Http://www.delphi-novatos.es.fm	La pagina del canal #Delphi_Novatos del cual soy miembro.