

Introducción a Visual C++ Builder de Embarcadero RAD Studio XE series

Introducción:

Embarcadero (*Borland*) Rad Studio C++ Builder es un IDE que permite utilizar la programación orientada a objetos utilizando como base el lenguaje C++ en modo visual. Aunque también permite hacer aplicaciones C++ de consola. Este IDE comparte las mismas funciones y librerías básicas, llamadas *VCL* que Delphi, lenguaje basado en lenguaje Pascal orientado a objetos.

- Para crear un proyecto en modo visual orientado a objetos escoger del menú: **New – Project** o **New - VCL Form Application** para emplear las librerías VCL para Windows. Las versiones XE incorporan la variante *Firemonkey* que permite compilar en multiplataforma OS – Android e incluso dispositivos móviles.
- Para un archivo Cpp en modo consola, escoger: **New – Other – Console application**. Puedes escoger el lenguaje C o C++

Conceptos básicos.

- **Proyecto**. Un proyecto es un conjunto de módulos de formulario, de códigos y de archivos que componen una aplicación. Los archivos que componen la aplicación se ven en la ventana View ▶ **Project Manager**.
- **Formulario**. Ventana donde se incluye los controles y el código asociado a dicho formulario.
- **Componentes o controles**. Los componentes son herramientas como cuadros, botones y etiquetas que se disponen en un formulario para permitir la entrada de datos o para presentar resultados
- **Objetos y Clases**: Un objeto es la instancia de un componente que pertenece a un tipo o clase determinada
- **Propiedades**. Usando la ventana de *Object inspector* (F11) se definen las propiedades de formularios y controles. Las propiedades especifican los valores iniciales de las características, tales como tamaño, nombre y posición. Y también llamados atributos del objeto.
- **Eventos**: Es quien desencadena las acciones del sistema. También llamados métodos.

Concepto de código:

Es el texto escrito en un programa que realizará una acción. Estas acciones pueden ser simples subrutinas, procedimientos o funciones según su complejidad.

- **Acciones o subrutinas**: Trozos de código como respuesta a un evento o acción. No recogen ni devuelven ningún valor.
- **Procedimientos**: Acciones que recogen o utilizan unos parámetros de entrada pero no devuelven ningún valor.
- **Funciones**: Son procedimientos que recogen valores de entrada y además devuelve un valor de salida al procedimiento desde donde se le llamó
- **Diagrama de Flujo**: Representa la esquematización gráfica de un algoritmo, el cual muestra gráficamente los pasos o procesos a seguir para estructurar bien un programa.

Reglas de nomenclatura al declarar procedimientos o variables:

- Deben comenzar por una letra. Distingue mayúsculas y minúsculas.
- No pueden contener puntos.
- No pueden superar los 255 caracteres
- Nombres de formularios no sobrepasar 40 caracteres
- No pueden tener el mismo nombre que una palabra clave
- Comentarios en el código se inician con: `//` o entre `/* */`

Eventos o acciones (sobre un componente):

Lista de eventos (Acciones sobre los controles)

Evento	Acción
Click	Al hacer clic con el mouse (o el teclado)
DragDrop	Al soltar un control arrastrado con el mouse
DragOver	Al Arrastrar un control con el mouse.
KeyDown	Presionar una tecla mientras el objeto tiene el foco.
KeyPress	Presionar una tecla y devolver su valor ASCII.
KeyUp	Liberar una tecla mientras el objeto tiene el foco.
MouseDown	Presionar el botón del mouse sobre el objeto.
MouseMove	Mover el puntero del mouse por encima del objeto
MouseUp	Liberar el botón del mouse en el objeto.

Propiedades básicas de los objetos:

Alignm. ent	- Alineación
BevellInner	- borde interior
BorderStyle	- estilo del borde
Caption	- rótulo
TabStop	- Salto tabulación
Color, Cursor	- Color, puntero
Enabled	- Activo
Font	- Tipo letra
Height	- Altura del objeto
Hint	- Etiqueta ayuda
Left	- Posición izquierda
Name	- Nombre del objeto
PopupMenu	- Menú contextual
ShowHint	- Mostrar etiquetas
Top	- Posición superior
Visible	- Visible
Width	- Ancho del objeto

Lenguaje base

Estructuras de control

- if (comparador si-entonces)
- if...else (*comparador si...sino*)
- for (*bucle con contador*)
- switch / case (*comparador múltiple*)
- while (*bucle por comparación booleana*)
- do... while (*bucle por comparación booleana*)
- break (*salida de bloque de código*)
- continue (*continuación en bloque de código*)
- return (*devuelve valor al programa principal*)

Tipos

- boolean (booleano)
- char (carácter)
- byte (entero hasta 226)
- int (entero)
- unsigned int (entero sin signo)
- long (entero 32b)
- unsigned long (entero 32b sin signo)
- float (decimal de coma flotante de 16b)
- double (decimal coma flotante de 32b)
- string (cadena de caracteres)
- array (cadena matriz)
- void (vacío)

Operadores

- = (asignación)
- + (suma)
- - (resta)
- * (multiplicación)
- / (división)
- % (resto)
- == (igual a)
- != (distinto de)
- < (menor que)
- > (mayor que)
- <= (menor o igual que)
- >= (mayor o igual que)
- && (y) || (o)
- ! (negación)
- ++ (incrementa) -- (decrementa)

Matemáticas

- min () (mínimo)
- max () (máximo)
- abs () (valor absoluto)
- sq () (eleva al cuadrado)
- sqrt () (raíz cuadrada)
- sin () (seno) cos () (coseno)
- tan () (tangente)

Conversión de tipos:

IntToStr (Integer to String) ► Cambia del tipo de número entero a cadena de texto

StrToInt (String to Integer) ► Cambia del tipo cadena de texto a número entero

Otros:

StrToFloat (texto a decimal flotante)

FloatToStr (Decimal flotante a texto)

FloatToDecimal (Decimal flotante a decimal)

Formatfloat (Decimal flotante a texto con formato)

DateTimeToStr (Fecha/hora a texto)

DateToStr (Fecha a Texto)

StrToTime (Texto a hora)

StrToDate (Texto a fecha)

Bucles:

Infinitos: **While**: Repite la ejecución infinitas veces *mientras* se cumpla la condición.

Finitos (Contadores): **For ...**, Repite la ejecución un número determinado de veces

Ventanas de mensaje. (Mostrar o pedir mensajes al usuario):

- Usando las funciones VCL internas que incorpora C++Builder:
 - ShowMessage(): Muestra un mensaje de texto en una ventana
 - MessageDlg(): Muestra una ventana con mensaje y botones de respuesta
Los botones de acción pueden ser: mbOK – mbCancel – mbYes – mbNo – mbAbort – mbRetry – mbClose
Y sus repuestas son: mrOk - mrCancel – mrYes – mrNo – mrAbort – mrRetry - mrClose
 - MessageDlgPos() ídem pero con posibilidades extras
- Usando las funciones de la API de Windows
 - Application->MessageBox(): Muestra un mensaje de texto en una ventana de Windows estándar con o sin botones de respuesta que pueden ser: MB_OK - MB_OKCANCEL- MB_ABORTRETRYIGNORE - MB_YESNOCANCEL - MB_YESNO -MB_RETRYCANCEL - MB_HELP
- Creando nuestros propios formularios/ventanas:
 - Primero: crear un nuevo formulario secundario: Flie – New – Form donde se pondrá el mensaje y los botones
 - Segundo: incluir la unidad cabecera de este formulario en la principal: #include "secundario.h"
 - Tercero: Mostrar la ventana utilizando las funciones show() o showmodal();

Pedir datos al usuario en ventanas:

- Utilizando ventanas con entradas de texto:
 - InputBox('Nombre_ventana', 'Texto_Dato', 'Respuesta_predeterminada'); → Devuelve Var String
 - InputQuery('Identificate!', 'Pon tu nombre, por favor...', Nombre); → Devuelve Var Boolean
- Añadiendo componentes de diálogo estándar al formulario desde la paleta Dialogs:
 - OpenDialog – Savedialog: Componentes para abrir y guardar archivos
 - PrintDialog – PrinterSetupDialog - PageSetupDialog: Componentes para imprimir y configurar impresión
 - FontDialog – ColorDialog - FindDialog: Cambiar la fuente, el color o buscar texto.

Cuadros de diálogo MessageBox.

Puedes hacer llamadas a cuadros de diálogo de la API de Windows usando: [Application->MessageBox](#)

Si añades la línea:

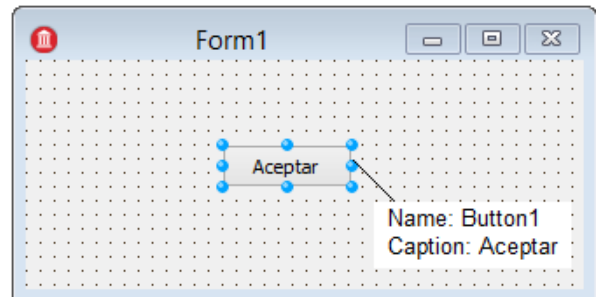
```
Application->MessageBox(L"¿Desea continuar?", L"Pregunta", MB_OKCANCEL); //con L si es Unicode
Application->MessageBox("¿Desea continuar?", "Pregunta", MB_OKCANCEL); //sin L si no es Unicode
```

Dispones de las variantes: MB_OK - MB_OKCANCEL- MB_ABORTRETRYIGNORE- MB_YESNOCANCEL- MB_YESNO- MB_RETRYCANCEL- MB_HELP

Ejercicio 1:

- Crear una nueva aplicación VCL (librería visual C para Windows) escogiendo del menú: File ► New ► VCL Form Application C++ Builder.
- En un formulario nuevo, desde la **Tool palette**, añade un componente **TButton** (botón *Button1*). Desde el panel de propiedades, cambia su propiedad **Caption** por: *Aceptar*
- Pulsa *doble clic* sobre el botón e introduce el código (en el evento **OnClick** del botón *Button1*) por:

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
    Button1->Caption="Vale";
    ShowMessage("Hola mundo");
}
```



(Importante escribir "Button" con la 1ª letra en mayúsculas, ya que C++ distingue mayúsculas de minúsculas)

- Prueba la aplicación pulsando en Run ►.
- Si funciona correctamente, guarda la aplicación: File ► Save all . El archivo de proyecto se llamará **Clase0cpp**.

Ejercicio 2:

Abre el proyecto anterior **Clase0cpp**.

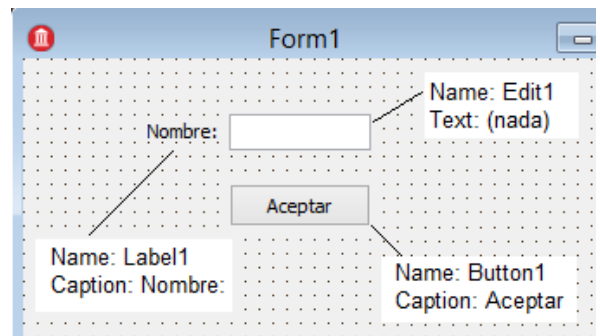
- Desde la **Tool palette**, añade al ejercicio anterior un componente **TEdit** y un componente **TLabel** con las propiedades de la imagen:

- Cambiar el evento: **OnClick** del botón *Button1* por:


```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
    ShowMessage("Hola, " + Edit1->Text);
}
```

ShowMessage: función para mostrar una ventana de mensaje

Mejoras: Utilizar una variable de texto: `String mensaje = "Hola"` y mostrar el mensaje: `ShowMessage(mensaje);`



Guarda el proyecto con el nombre: **ClaseunoCpp**

Ejercicio 2b. Pide el PIN:

- Cambia la propiedad *Label1.Caption*="Pin: "
- Cambia el Código en el evento **Button1Click**: →

Variante 3: Al pulsa Intro:

Al pulsar *Intro* en el *Edit1* comprueba si lo escrito es correcto utilizando la función **key**: →

- Cambia el Código en el evento **Edit1KeyUp**: →

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
    if (Edit1->Text=="123") ShowMessage("Bien");
    else ShowMessage("Mal");
}
```

```
void __fastcall TForm1::Edit1KeyUp(...
{ int numero; //creamos una variable tipo entero
  if (Key==13) { numero=(StrToInt(Edit1->Text));
    if (numero==555) ShowMessage("bien");
  } }
```

Ejercicio 3. Contador:

StrToInt y **InttoStr**: Son funciones que convierten un valor de numérico entero (**Integer**) a texto (**String**) y viceversa

- Recupera el ejercicio anterior.
- Cambia el texto (propiedad *caption*) del botón por: **0**
- Añade un segundo botón a la ventana formulario desde la paleta **Tool palette** – sección Standard: **Tbutton**
- Al botón 2 cambia el texto (propiedad *caption*) por **Contar**
- Pulsa doble clic encima de él, para abrir la ventana de código.
- Entre las llaves { } debes escribir el código:

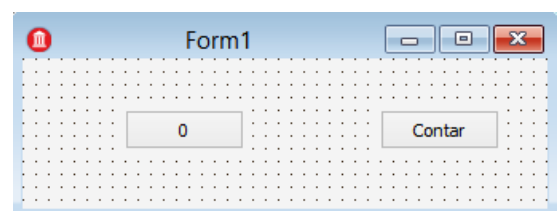

```
Button1->Caption= Button1->Caption +1;
```

Si lo ejecutas (run), verás cuenta mal porque añade la letra 1 pero no suma el número 1

Para ello hay que convertir las letras en números enteros: *string to integer* o abreviado: **StrToInt**

```
Button1->Caption = StrToInt(Button1->Caption) +1; // Si ahora lo ejecutas, verás que ya funciona.
```

- Si funciona correctamente, guarda la aplicación: File – Save Project as.. El archivo de proyecto se llamará **Contador**. El archivo unidad se llamará: **Contador1.cpp**, el proyecto **Contador.cbproj** y su archivo cabecera **ContadorCH1.h**



Ejercicio. Adivina el número.

- Crear una nueva aplicación VCL escogiendo del menú: File - New – VCL Form Application C++ Builder.
- En el formulario nuevo, desde la *Tool palette*, añade los siguientes

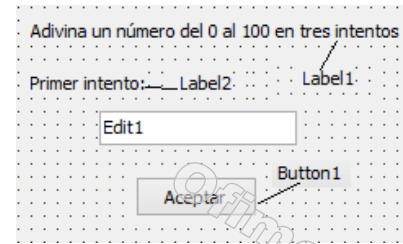
Ingredientes:

2 Labels **A** - 1 Edit **ab** - 1 Button **OK**

Preparación:

Creamos 2 variables públicas: **Bueno**: integer y **Intentos**: integer:

```
#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
int bueno, intentos; // -> declaramos dos variables numéricas enteras públicas al inicio
```



Cocción:

Al crearse el formulario se genera un número aleatorio del 1 al 100 y se guarda en la variable *bueno*:

```
void __fastcall TForm1::FormCreate(TObject *Sender)
{
    Randomize(); //función baraja. Inicia el generador de aleatorios
    bueno=random(100); //crea un num aleatorio entre 0 y 100 y lo guarda en la variable bueno
    intentos=0; //pone intentos a cero
}
```

Acciones: Al pulsar el botón, se crea una variable *numerousuario* y se evalúa si la variable es igual o mayor a la variable *bueno*:

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
    int numerousuario; //declaramos una variable numérica entera privada válida sólo para esta función
    numerousuario = StrToInt(Edit1->Text); //pasa valor de texto a numérico y lo guarda en la variable
    if (numerousuario==bueno) {
        ShowMessage("Acertaste");
    }
    else
    {
        ShowMessage("Fallaste, el número era el " + IntToStr(bueno));
    }
}
```

Mejoras:

Mejora 1ª: Consigue que el programa te informe si el número introducido es mayor o menor

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
    int numerousuario;
    numerousuario = StrToInt(Edit1->Text);
    if (numerousuario==bueno) {
        ShowMessage("Acertaste");
    }
    else
    {
        if (numerousuario>bueno) {
            ShowMessage("Fallaste, el número es más pequeño");
        }
        else
        {
            ShowMessage("Fallaste, el número es más grande");
        }
    }
}
```

Mejora 2ª:

- Añadir una barra de progreso: **progressbar1** (paleta win32) *Progressbar*: Max: 60 Position: 60
- Añadir un Temporizador: **Timer** (paleta system) Interval: 1000 (Repetirá el evento cada 1 segundo)
En el evento del timer **OnTimer** añadir el código para evaluar el tiempo tardado en acertar y *decrementar* la barra de progreso. `if (ProgressBar1->Position=0) {Showmessage('Se acabó el tiempo'); Timer1->enabled=false;}`

Mejora 3ª:

Al pulsar la tecla *Enter* en el *Edit1* (*Edit1KeyPress*), llama a la función *Button1Click*:

```
if (Key==13) Button1Click(Sender); o if (Key==13) Button1->SetFocus(); //Button1 recibe el foco
```

Guardar: Guarda la aplicación (Save all).

La unidad con el nombre: **adivinanza1.cpp** y el proyecto con el nombre: **adivinanza.cproj**

Uso del Timer y posición del objeto:

- Crea una nueva aplicación VCL (librería visual) escogiendo del menú: File - New – VCL Form Application C++ Builder.
- Añade un botón *Button1* en un formulario nuevo. Cambiar su propiedad **Caption** por: *Púlsame*.
- En el evento del botón *OnClick*:

```
void __fastcall TForm6::Button1MouseMove(TObject *Sender, TShiftState Shift, int X, int Y)
{
    Button1->Left = Button1->Left + 1;
}
```

Ejecuta y prueba (Run): Al intentar pulsar el botón, este se desplaza hacia la derecha.

- Anula esta última línea de código convirtiéndola en un comentario: añade // delante
- Añade un componente **Timer** de la paleta System.

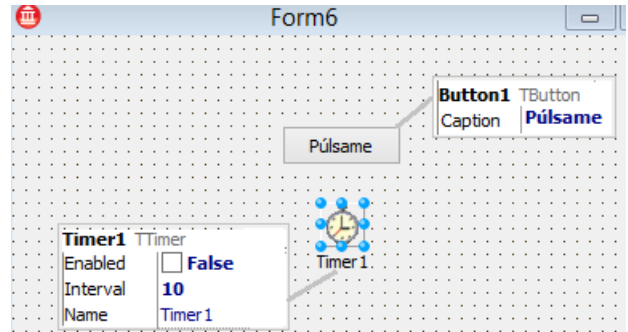
Poner su propiedad *Interval* a 10 y si propiedad *enabled* en *False*:

En el evento del botón *OnClick*:

```
void __fastcall TForm6::Button1Click(TObject *Sender)
{
    Timer1->Enabled = True;
}
```

En el evento del Timer *OnTimer*:

```
void __fastcall TForm6::Timer1Timer(TObject *Sender)
{
    Button1->Left = Button1->Left + 1;
}
```



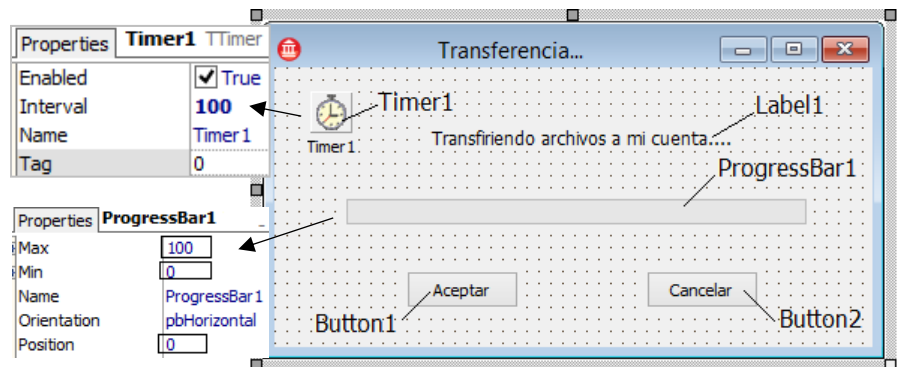
Ejecuta y prueba (Run): Al pulsar el botón, activamos el temporizador Timer que desplaza un pixel a la derecha la posición del botón a cada pulso de su intervalo, de duración 10 milisegundos.

Ejercicio ventana con barra de progreso

- Crea un nuevo proyecto: File ▶ New ▶ Application VCL (o *VCL Forms application*).

Diseño:

- Añade al formulario un objeto barra de progreso (*ProgressBar* de la paleta *win32*), dos botones (*tButtons*), una etiqueta *tLabel* (de la paleta *Standard*) y un componente *tTimer* (Paleta System)
- Distribuye y cambia las propiedades de la ventana-formulario como en la imagen.



Eventos: (Suceso que desencadena una acción)

- Pulsa *doble clic* sobre el evento **OnTimer** del **Timer1** para mostrar el código de su procedure y añade:

```
void __fastcall TForm1::Timer1Timer(TObject *Sender)
{
    ProgressBar1->Position++; //-> incrementa la barra una posición
    if (ProgressBar1->Position==100) { // -> si llega a 100...
        Timer1->Enabled=False; // -> para el temporizador
        ShowMessage("Transferencia realizada"); // -> muestra un mensaje
        Application->Terminate(); // -> termina el programa
    }
}
```

En el evento del botón *OnClick*:

```
void __fastcall TForm6::Button1Click(TObject *Sender)
{
    Timer1->Enabled = True;
}
```

Opcional: Puedes impedir cerrar la ventana si en el evento del form: *CloseQuery* pones la variable *canClose=false*;
Ejemplo: `ShowMessage("No puedes cerrar la ventana mientras se transfieren los archivos"); canClose=false;`

- **Probar la aplicación:** Pulsa Run ▶ para comprobar su funcionamiento.
- **Guardar el proyecto:** Crea una carpeta que se llame: Transferencia
 - Escoge: File ▶ Save as... Para guardar el archivo: *Transferencia1.cpp*
 - Escoge: File ▶ Save project as... Para guarda el proyecto *Transferencia.cbproj* y *Trasferencia.hpp*

Bucles:Infinito:**While:**

Repite la ejecución infinitas veces hasta que se cumpla la condición.

```
int clave=5;
while (clave != 123)
{
    ShowMessage("incorrecto, repita la clave");
}
```

Finito**For ... :**

Ejecuta una acción un número determinado de veces.

(Contador):

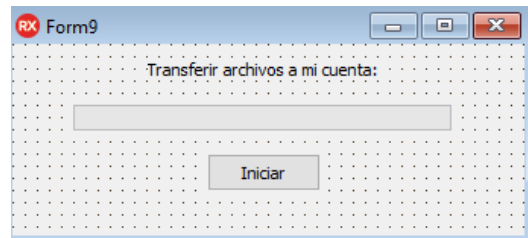
```
for (int i=0; i < 10; i++) {
    ShowMessage("Contando hasta diez: " + IntToStr(i));
}
```

Ejercicio while - for**Barra de Progreso con while (sin timer)**

- Crea un nuevo proyecto para Windows: File ► New ► Windows VCL Application (C++ Builder).

Diseño:

- Añade al formulario un objeto barra de progreso *tProgressBar*
- Un botón (*tButton*) y una etiqueta *tLabel* (de la paleta *Standard*)
- Distribuye y cambia las propiedades de la ventana-formulario como en la imagen.

**Eventos:** (Suceso que desencadena una acción)

- Pulsa *doble clic* sobre el Botón y escribe el código siguiente según la versión escogida:

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
    while (ProgressBar1->Position<100)           //mientras sea menor a 100...
    {
        ProgressBar1->Position ++;              // incrementa 1 paso
        Sleep(200);                             // delay o retardo
    }
    ShowMessage("Transferencia realizada");      // muestra un mensaje
    Application->Terminate();                   // termina el programa
}
```

Probar la aplicación: Pulsa Run ► para comprobar su funcionamiento.

Barra de Progreso con for:

Ahora añade un segundo botón con esye código y comenta la diferencia:

```
void __fastcall TForm1::Button2Click(TObject *Sender)
{
    for (int i=1 ;100;i++)                       //repite 100 veces
    {
        ProgressBar1->Position ++;              // incrementa 1 paso
        Sleep(200);                             // delay o retardo
    }
    ShowMessage("Transferencia realizada");      // -> muestra un mensaje
    Application->Terminate();                   //-> termina el programa
}
```

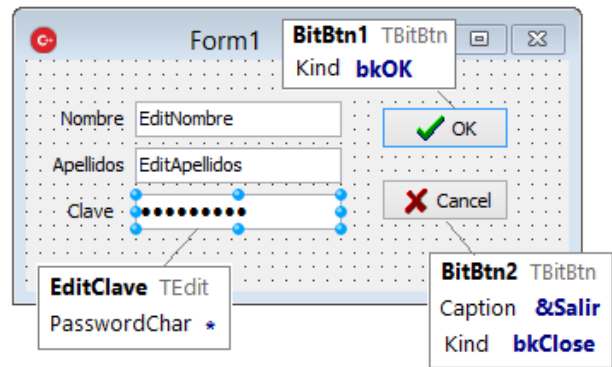
- **Guardar el proyecto:** Crea una carpeta que se llame: Transferencia2

- Escoge: *File ► Save as... Para guardar el archivo: Transferencia2.cpp*
- Escoge: *File ► Save project as... Para guarda el proyecto Transferencia2.cbproj y Trasferencia2h.hpp*

Construcción de una aplicación con dos formularios. Mensajes

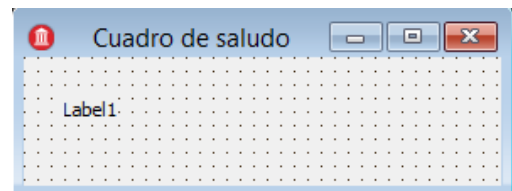
Teoría previa: **if/else** permite separar las acciones en dos casos; el caso verdadero o falso según se cumpla o no la comparación. *Sintaxis:* `if (comparación) {acción_verdadera} else {acción_falsa}`

- ▶ Crear aplicación nueva:
- ▶ (File - New – VCL Form Application C++ Builder)
- Formulario principal:** Poner los siguientes componentes:
 - 3 Label (solapa Standard), 3 Edits (solapa Standard), y 2 botones BitBtn (solapa Additional).
- ▶ Distribuir los objetos para que la ventana tenga el aspecto de la imagen.
- ▶ En una carpeta nueva llamada: Mensajes, guarda esta unidad (File ▶ Save as...) con el nombre *mensaje1.cpp*, el archivo cabecera por defecto y el proyecto con el nombre: *mensajes.cbproj*



Configurar el formulario secundario.

- ▶ Añade un nuevo formulario al proyecto: selecciona File ▶ *New Form*
- ▶ Cambia las propiedades del formulario:
 - Name = *SaludoBox* y Caption = Cuadro de saludo.
- ▶ Añade un componente Label llamado Label1. Como en la imagen.
- ▶ Guarda esta unidad (File ▶ Sav e as...) con el nombre *saludo1.cpp* y el archivo cabecera: *saludo.h*



Escribir el código.

En este caso, sólo se escribirá código para el evento *OnClick* del componente BotonOK del formulario principal. Con dicho componente seleccionado, ir a la pestaña *Events* del inspector de objetos y hacer *doble click* en el evento *OnClick*. El gestor de este evento quedará como sigue:

```
void __fastcall TForm1::BitBtn1Click(TObject *Sender)
{
    SaludoBox ->Label1->Caption = "Bienvenido, " + EditNombre->Text ; //pone texto en Label1 (*)
    SaludoBox ->Show(); //muestra la ventana en modo no modal cambiar por ShowModal()
}
```

(*) Entorno multidevice la propiedad es: Label1->Text

En el formulario principal, seleccionar **File - Use Unit** y seleccionar **Saludo1**. Observar que se añade la línea: **#include "Saludo.h"** y ejecuta el programa.

Variante 1:

Cambia el código del evento *OnClick* por el siguiente:

```
void __fastcall TForm1::BitBtn1Click(TObject *Sender)
{
    if (EditNombre->Text == "" || EditApellidos->Text == "" || EditClave->Text == "" )
    { ShowMessage ("Debe especificar todos los campos"); //mostrar mensaje
    }
    else {
        SaludoBox->Label1->Caption = "Bienvenido, " + EditNombre->Text + " " +
        EditApellidos->Text + ". Su clave es: " + EditClave->Text;
        SaludoBox->ShowModal(); //muestra ventana en modo modal
    } }
}
```

|| equivale a un O lógico, && equivale a un Y lógico

El siguiente código muestra un mensaje si la clave es incorrecta y prepara entrada para nuevo usuario:

```
void __fastcall TForm1::BitBtn1Click(TObject *Sender)
{
    if (EditNombre->Text == "" || EditApellidos->Text == "" || EditClave->Text == "" )
    { ShowMessage ("Debe especificar todos los campos"); //mostrar mensaje
    }
    else if (EditClave->Text != "ofimega") {ShowMessage("Clave incorrecta");
    }
    else {
        SaludoBox->Label1->Caption = "Bienvenido, " + EditNombre->Text + " " +
        EditApellidos->Text + ". Su clave es: " + EditClave->Text;
        SaludoBox->ShowModal(); //muestra ventana en modo modal
        //prepara un nuevo usuario:
        EditNombre->Text = "";
        EditApellidos->Text = "";
        EditClave->Text = "";
        EditNombre->SetFocus();
    }
}
```

En el siguiente código, cambia el error de clave (`ShowMessage("Clave incorrecta")`) por una ventana de diálogo: `MessageDlg("Clave incorrecta",mtError,mbAbortRetryIgnore,0);` o `MessageDialogSync` en plataforma FMX

Ejercicio 4: Conversor

Teoría previa:

- › **StrToInt** y **IntToStr**: Son funciones que convierten un valor de numérico entero (*integer*) a texto y viceversa
- › **FloatToStr** y **StrToFloat**: Convierten un valor de numérico decimal flotante (*float*) a texto y viceversa
- › **Sender**: Es un parámetro que indica el nombre del objeto que ha desencadenado el evento

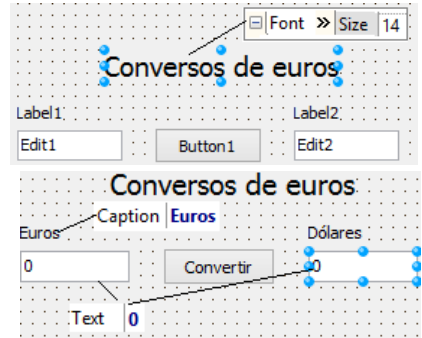
Ejercicio:

- Crear una nueva aplicación
 - **Versión VCL** (librería visual c para windows) escoge del menú: File - New – VCL Form Application C++ Builder.
 - **Versión Multidevice**: File - New - Multi-device application C++ Builder
- En el formulario nuevo, desde la Tool palette, añade: 2 edits, 1 botón y 3 labels
- Cambia la propiedad **caption** (o *Text*) de las *label* y del *botón* para que muestre el texto de la imagen derecha. →
- Pulsa doble clic encima del botón para llamar al evento (on click) `void __fastcall TForm1::Button1Click(TObject *Sender)` y escribe:


```
Edit2->Text=Edit1->Text * 1.3;
```

 Si no funciona correctamente escribe en su lugar:


```
Edit2->Text=StrToInt(Edit1->Text) * 1.3;
```
- Pulsa en Run ▶ para comprobar su funcionamiento.
 - Si funciona correctamente, guarda toda la aplicación: Fila – Save all
 - El archivo de unidad se llamará: **Conversor1.cpp**, el proyecto **Conversor.cbproj** y su archivo cabecera **ConversorCH1.h**

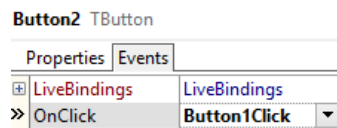
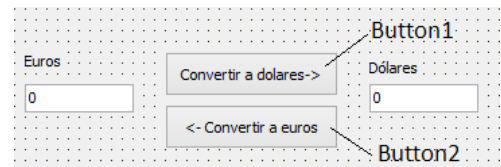


Usar dos botones con la misma función (usando el parámetro sender)

Añade un segundo botón *Button2* como en la imagen. →

En el código del botón *Button1* cambia el código por el del recuadro.

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
    if (Sender = Button2) {
        Edit2->Text=Edit1->Text*1.3;
    }
    else
    {
        Edit1->Text=Edit2->Text/1.3;
    }
}
```



- En el evento onclick del botón 2 tienes que asignar el mismo evento que el del botón 1 para que ambos botones llamen a la misma función.
- Si funciona correctamente, guarda toda la aplicación: Fila – Save all.

Versión Multidevice para android (Firemonkey. Para versiones XE3 y superiores)

- Escoje: File - New - Multi-device application C++ Builder (Versiones XE3 ~ XE4: File – New – Firemonkey)
- Diseño: *Header footer* -
- Guardar en la carpeta: *conversor*.
- Añadir 2 labels, 2 Spinbox y 2 botones.
- Cambiar las propiedades como en la figura: →
- Añadir el código a cada botón:
 - ButtonCmClick:


```
SpinBox2->Value=SpinBox1->Value*2.54;
```
 - ButtonPulgaClick:


```
SpinBox1->Value=SpinBox2->Value/2.54;
```
- Si funciona correctamente, guarda toda la aplicación: Fila – Save all.
 - Guarda la unidad **cpp**, el archivo cabecera **h** y el proyecto: **cbproj** con el nombre: **ConversorM.cbproj**



Ejercicio 4 ext: - Área del triángulo

- Añadir al ejercicio anterior los componentes de la imagen.
- Cambiar el evento: **OnClick** del botón *Button1* por:

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
    Edit3->Text=FormatFloat("###.##", Edit1->Text.ToDouble() *
        Edit2->Text.ToDouble()/ 2 );
}
```

Versión Multidevice o Firemonkey (XE):

- *Nota:* En la etiquetas Labels, cambia la propiedad **Caption** por la propiedad **Text**.

- Añadir al *Form1* un componente: **StyleBook1**

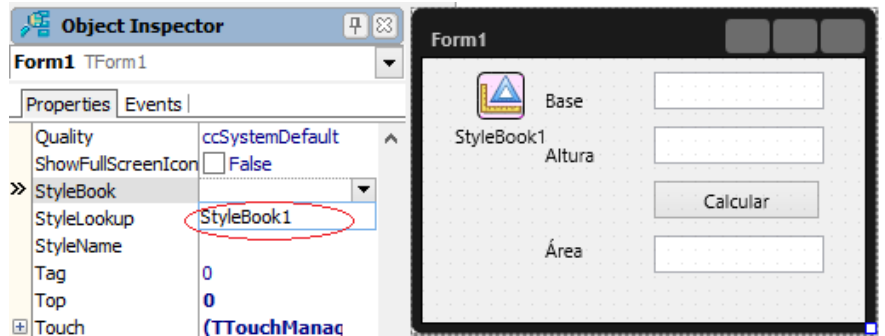
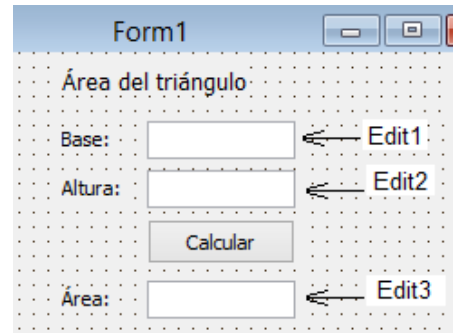
- Pulsar doble clic sobre el componente **StyleBook1** y cargar (Load...) de la carpeta:

Users\Public\Documents\RAD
Studio\12.0\Styles el estilo:

AquaGraphite.style – Apply and close

- Cambiar la propiedad del formulario **Form1. Stylebook:**

- Guardar (File - Save Project) Guarda la unidad **cpp**, el archivo cabecera **h** y el proyecto: **cbproj**



Uso de variables de fecha y decimal:

Una variable almacena un valor temporalmente para utilizarlo después. Hay que declararlas antes de usarlas.

Variables primitivas: pueden ser numéricas (*integer*), decimales (*single* o *double*), lógicas (*bool*)

Variables Complejas o de un tipo de clase: Son de fecha (*Tdatetime*), de texto(*String*) o de objetos.

- Crea una nueva aplicación VCL (librería visual) escogiendo del menú: File - New – VCL Form Applicaton.

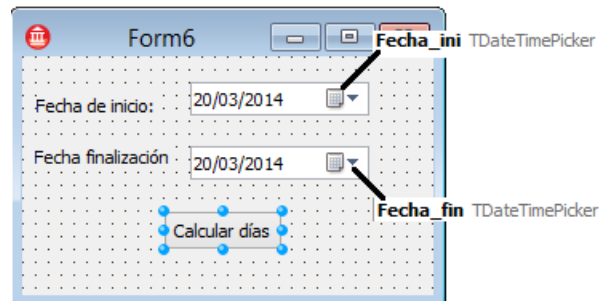
- Añade un botón *Button1*, dos etiquetas, labels y dos *DatetimePicker* de la paleta *Win32*.

- Cambia el nombre de los *DatetimePicker* por: *Fecha_ini* y *Fecha_fin*

- Pulsa doble clic sobre el botón y escribe en el evento del botón **OnClick**:

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
    double dias; //-> creo una variable decimal doble
    TDateTime hoy; //-> creo una variable del tipo fecha/hora (empieza con mayúsculas)
    dias=Fecha_fin->Date-Fecha_ini->Date; // -> resto las dos fechas
    hoy=Date(); //Date() recoge la fecha del sistema
    ShowMessage("Hoy es "+ DateToStr(hoy)+ " y han transcurrido "+FloatToStr(dias)+" dias");
}
```

- Pulsa Run > para comprobar su funcionamiento.

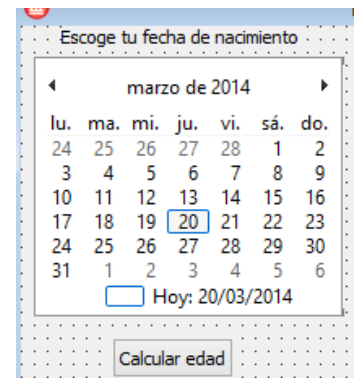


Variante del ejercicio. Calcula tu edad:

- Cambia el formulario como en la imagen →

- Escribe en el evento del botón **OnClick**:

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
    double dias,anyos; //-> declaro dos variables decimales
    TDateTime hoy = Date(); //-> declaro y asigno un valor a la vez
    dias=hoy-MonthCalendar1->Date; //-> calculo los días transcurridos
    anyos=dias/365; //-> calculo los años transcurridos
    ShowMessage("Tienes "+FormatFloat("##0.##",dias)+" días o "+FormatFloat("##0.##",anyos)+ " años");
}
```



- **Ejecuta y guarda:** Pulsa Run > para comprobar su funcionamiento.

Para guardar primero la unidad y luego el proyecto, escoge del menú: File ▶ Save project as...

Uso del switch – case – Sender. Ejercicio direcciones

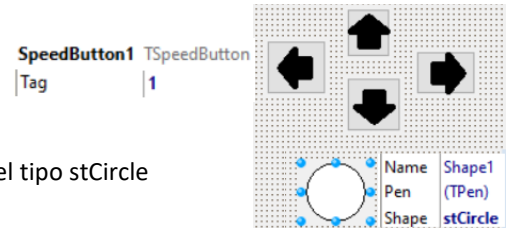
Preparación:

- ▶ Crea una Carpeta nueva llamada *Direcciones* donde vamos a guardar nuestro proyecto.
- ▶ Utiliza una aplicación como Paint para Guardar en la carpeta Direcciones las cuatro imágenes siguientes en formato bmp de 16 bits.



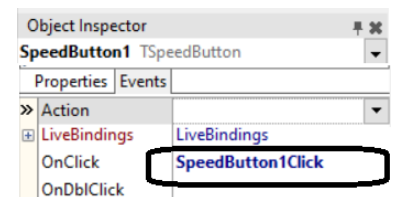
Flecha_ab.bmp Flecha_ar.bmp Flecha_der.bmp Flecha_iz.bmp

- ▶ En Visual C++ Builder, crea un nuevo proyecto (menú: File - New – VCL Form Application C++ Builder)
- ▶ Añade al formulario 4 **Speedbuttons** asigna a cada uno, en su propiedad *Glyph* las cuatro imágenes anteriores asigna a cada una en la propiedad *Tag* los números del 1 a 4
Con la propiedad *Tag* podemos marcar un objeto con un valor



- ▶ Añade al formulario un componente **Shape** y asigna su propiedad Shape: el tipo stCircle

En el primer SpeedButton, pulsa doble clic sobre el evento: OnClick →



- Escribe en el evento el código:

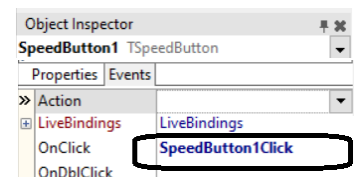
```
void __fastcall TForm1::SpeedButton1Click(TObject *Sender)
{
    TSpeedButton *Boton=(TSpeedButton*) Sender; //variable objeto boton recogido del Sender
    switch (Boton ->Tag) //evaluamos el Tag del boton
    {
        case 1: Shape1->Top++; //si el Tag es 1 desplaza arriba
        break;
        case 2: Shape1->Left++; //si el Tag es 2 desplaza a la derecha
        break;
        case 3: Shape1->Top--; //si el Tag es 3 desplaza la izquierda
        break;
        case 4: Shape1->Left--; //si el Tag es 4 desplaza a abajo
    }
}
```

Esta función distinguirá desde qué botón ha sido llamada evaluando la propiedad numérica Tag.

Para evaluar la propiedad Tag del objeto recibido por el Sender se lo asignamos a una variable del tipo objeto TSpeedbuton que la llamaremos Boton.

Otra opción para evaluar una propiedad del Sender es: **dynamic_cast** < TSpeedButton *(Sender)->Tag

- Asigna a los otros botones el mismo evento del primer botón:
Esto hará que todos los demás botones desencadenen a la misma función que el primer botón.



Prueba esta otra alternativa en el código con la opción if – else if, que puede parecer más corta pero tenemos que saber el nombre del botón...

```
void __fastcall TForm9::SpeedButton1Click(TObject *Sender)
{
    if (Sender = SpeedButton1) { Shape1->Top++;}
    else if (Sender = SpeedButton2) { Shape1->Top--;}
    else if (Sender = SpeedButton3) { Shape1->Left++;}
    else if (Sender = SpeedButton4) { Shape1->Left--;}
}
```

Ejecuta y guarda: Pulsa Run ▷ para comprobar su funcionamiento.

Para guardar el proyecto y la unidad, escoge del menú: File ▶ Save project as...

Pizarra de dibujo (VCL)

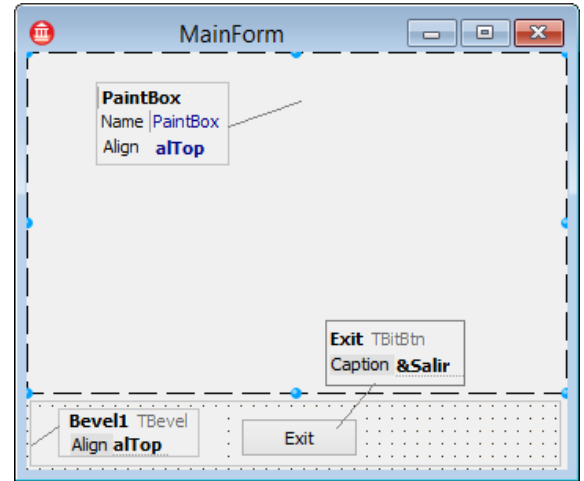
- Crear una nueva aplicación VCL (librería visual) escogiendo del menú: File - New – VCL Form Application C++ Builder.
- Cambiar el nombre y título del formulario principal: Con el inspector de objetos, cambiar la propiedad **Caption** por **PIZARRA** y la propiedad **Name** por **MainForm**.
- Guardar (File ▶ Save As) la unidad asociada al formulario (Unit1.cpp, por defecto) como **PizarraUnit.cpp**. El archivo cabecera a **Pizarrah.h** y el proyecto (File | Save Project As) como **Pizarra.bpr** o **.cbproj**
- **Poner icono:** Para asignar un icono: (Project - Options ▶ Application). Puls: **Load Icon...** - Seleccionar un icono.
- **Temas en XE:** Asignar un tema: Esoge (Project - Options - Application) - Appearance. Escoge: **Amakrits**. Asigna el tema como Default y pulsa Ok
- Comprobamos: Project – Run.

Diseño del formulario principal:

- Fijar manualmente el tamaño del Form a 300 X 400:
(Height = 300 y Width = 400) Nombre: MainForm

Añadir 3 componentes:

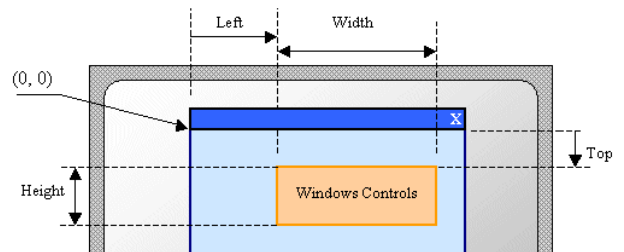
- En la parte superior, alineado a lo alto (Align: alTop), un componente **PaintBox** (solapa System). Fijar las propiedades: Name = PaintBox, Align = alTop y Height = 225.
- Debajo del PaintBox un componente **Bevel** (solapa Additional). Fijar las siguientes propiedades: Name = Linea, Align = alTop, Style = bsLowered y Height = 8.
- Debajo del Bevel, centrado horizontalmente un componente **BitBtn** (solapa Additional). Fijar las siguientes propiedades: Name = ExitBtn, Kind = bkClose y Caption = &Salir.



Escribir el código.

En primer lugar, escribiremos el código asociado al evento OnPaint del componente PaintBox. Seleccionar en el inspector de objetos dicho componente, y tras seleccionar la pestaña **Events** haremos doble click en el gestor del evento OnPaint. El gestor de este evento quedará como sigue:

```
void __fastcall TMainForm::PaintBoxPaint(TObject *Sender)
{
    // Dibujar un círculo amarillo:
    PaintBox->Canvas->Pen->Color = clBlack;
    PaintBox->Canvas->Brush->Color = clYellow;
    PaintBox->Canvas->Ellipse (100, 100, 150, 150);
    // Dibujar un cuadrado rojo:
    PaintBox->Canvas->Pen->Color = clRed;
    PaintBox->Canvas->Brush->Color = clRed;
    PaintBox->Canvas->Rectangle (200, 100, 250, 150);
    /*Si quisiera dibujar una linea en diagonal:
    PaintBox->Canvas->MoveTo(0,PaintBox->Height/2);
    PaintBox->Canvas->LineTo(PaintBox->Width,PaintBox->Height/2);*/
} //-----
```



Finalmente, escribiremos el código asociado al evento OnClick del componente BitBtn (**ExitBtn**):

```
void __fastcall TMainForm::ExitBtnClick(TObject *Sender) //-----
{
    Application->Terminate(); //-> termina la aplicación
    //o también: MainForm->Close(); //-> cierra la ventana
} //-----
```

Eventos del Mouse: Dibujar en la Pizarra:

Crearemos una variable pública **Pintar** del tipo "Flag" booleriana que nos dé permiso para pintar al presionar el botón del mouse pero no lo permita si movemos el mouse sin apretar el botón.

- Recupera el proyecto anterior (File ▶ Open): **Pizarra. .cbproj**
- Anula el código de la Función **PaintBoxPaint**. Para ello utiliza los caracteres **/* y */** para convertirlo en comentario.
- Añade la variable booleriana tras las líneas de Include de cabecera iniciada por defecto en false: **bool Pintar = false;**
- Añade las siguientes funciones para los eventos on mousedown, on mouse up y on mousemove del objeto Paintbox:

```
//-----
void __fastcall TMainForm::PaintBoxMouseDown(TObject *Sender, TMouseButton Button, TShiftState Shift, int X, int Y)
{
    Pintar = True;
    PaintBox->Canvas->MoveTo(X,Y); //los valores de X e Y son las coordenadas del mouse para empezar al pintar
}
void __fastcall TMainForm::PaintBoxMouseUp(TObject *Sender, TMouseButton Button, TShiftState Shift, int X, int Y)
{
    Pintar = False;
}
void __fastcall TMainForm::PaintBoxMouseMove(TObject *Sender, TShiftState Shift, int X, int Y)
{
    if (Pintar==True) {PaintBox->Canvas->LineTo(X,Y); //los valores de X e Y son las coordenadas del mouse
}
}
```

Ejecuta y prueba (Run): Al arrastrar el mouse sobre el objeto *Paintbox* dibujará desde la coordenada que se indica en *PaintBoxMouseDown*.

Guardar: Guarda todo (File ▶ Save All)

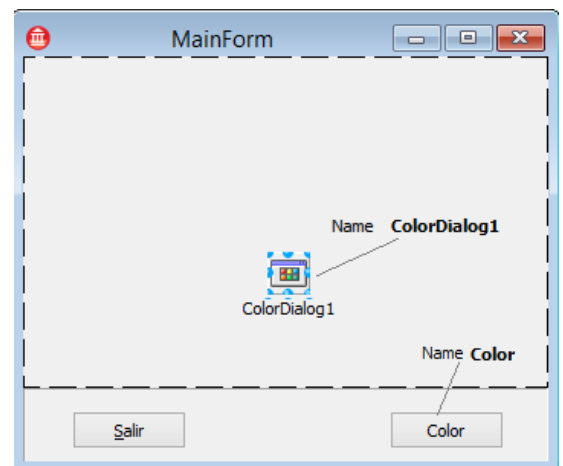
Cuadro de diálogo color.

Añade un nuevo botón **BitButton** junto al de Salir llamado: *Color* y un Componente **ColorDialog** de la paleta Dialogs.

Desencadena el evento Onclick donde añadirás el código:

```
void __fastcall TMainForm::ColorClick(TObject *Sender)
{
    if (ColorDialog1->Execute()) {
        PaintBox->Canvas->Pen->Color = ColorDialog1->Color;
    }
}
```

Comprueba su funcionamiento (Run) y Guarda todo (Save All)



Probar y guardar: Pulsar *Run* ▶ *para prueba* y Guarda todo (File ▶ Save All)

Esto guardará: *PizarraUnit.cpp* - *Pizarrah.h* - *Pizarra. .cbproj*

Archivo INI de configuración: (ampliación)

```
#include <IniFiles.hpp> //añadimos la librería para control de archivos INI
String dire;
dire=ExtractFilePath(ParamStr(0)); //path de archivos
void __fastcall TForm1::FormCreate(TObject *Sender)
{
    TIniFile *INI = new TIniFile("Config.ini"); //creamos un nuevo objeto o instancia
    Left = INI->ReadInteger("FormPos", "Left", 250);
    Top = INI->ReadInteger("FormPos", "Top", 200);
    delete INI;
}
//-----
void __fastcall TForm1::FormClose(TObject *Sender, TCloseAction &Action)
{
    TIniFile *INI = new TIniFile("dire+Config.ini"); //creamos un nuevo objeto o instancia
    INI->WriteInteger("FormPos", "Left", Left);
    INI->WriteInteger("FormPos", "Top", Top);
    delete INI;
}
```

Ejercicio - Uso de rejilla StringGrid con Firemonkey

- Crear una nueva aplicación:
 - Versiones posteriores a XE7: *File - New - Multi-device application C++ Builder*
 - Versiones XE anteriores: *File - New - Firemonkey - Mobile Application C++ Builder*
- Diseño: *Header footer* - Guardar en la carpeta Elementos
- Inserta una etiqueta *Label* y una rejilla *Grid* o mejor *StringGrid* (porque va a contener sólo texto) en el formulario.
- Con el botón derecho sobre la *Stringgrid* escoger: *Items editor* y añadir 2 columnas. Pon su propiedad *Align* al *Bottom*.
- Cambia la propiedad *Header* de la *column1* y *column2* por: elemento y nº atómico.
- Para introducir los datos en tiempo de ejecución, activar el evento *On Show* (al mostrar la ventana).

Escribe el código para el evento: *onshow FormShow*:

```
void __fastcall THeaderFooterForm::FormShow(TObject *Sender)
{
    StringGrid1->Cells[0][0]="Hidrógeno";
    StringGrid1->Cells[1][0]="1"; //columna 1 fila 0
    StringGrid1->Cells[0][1]="Helio"; //columna 0 fila 1
    StringGrid1->Cells[1][1]="2"; //columna 1 fila 1
}
```

- Al hacer clic en un elemento de la lista debe aparecer su valor en la etiqueta *Label1*

Escribe el código para el evento *OnSelectCell* de la *StringGrid1*:

```
void __fastcall THeaderFooterForm::StringGrid1SelectCell(TObject *Sender, const int ACol,
    const int ARow, bool &CanSelect)
{
    int fila; //-> creo la variable entera fila
    fila=ARow; //recojo el valor devuelto por ARow y lo pongo en mi variable fila
    Label1->Text=StringGrid1->Cells[0][fila] + " = " + StringGrid1->Cells[1][fila];
}
```

- Comprueba *runando* la aplicación [F9] y si funciona correctamente, guarda toda la aplicación: *Fila - Save all*.




Buscador de elementos	
Esoja de la lista....	
Elemento	Nº Atómico

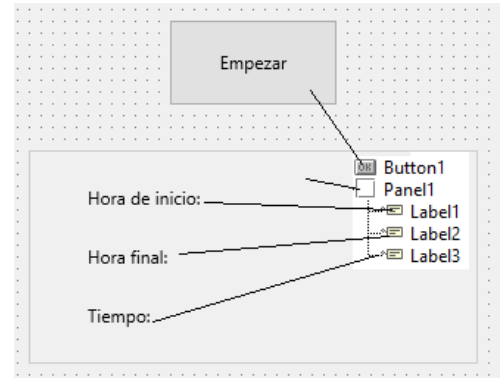
Ejercicio: Crear un contador de tiempo. Practicar con tipos de variables.

Variables públicas: Crearemos dos variables para almacenar dos datos: la hora de inicio y la hora final. Si estas variables van a ser utilizadas en todo el programa, se crea al principio, para que sean reconocidas por toda la unidad.

Ejercicio:

► Crear una nueva aplicación llamada **CRONO**. Añade al formulario (ventana) los siguientes componentes:

- 1 SpeedButton  Texto o caption: Empezar
Propiedades en VCL: Down ✓ AllowUp ✓ y GroupIndex=1
Propiedades en FMX: StaysPressed ✓ IsPressed ✓
- 1 panel  con 3 Labels  con Autosize ✓



Al inicio: double horaInicio, horaFin;

```

Versión para VCL:
void __fastcall TForm1::SpeedButton1Click( . . . )
{
double crono=0;
if (SpeedButton1->Down ==true) {
Label1->Caption = "Hora de inicio: " +
TimeToStr(Time());
SpeedButton1->Caption="Parar";
horaInicio=Time();
}
else {
Label2->Caption = "Hora final: " + TimeToStr(Time());
SpeedButton1->Caption="Empezar";
horaFin=Time();
crono=(horaFin-horaInicio)*100000;
Label3->Caption="Crono: " +
FloatToStr(RoundTo(crono, -3));
}
}

```

```

Variantes en multidevice FMX:
void __fastcall Form1::SpeedButton1Click(. . . )
double crono=0;
if (SpeedButton1->IsPressed ==true) {
Label1->Text = "Hora de inicio: " +
TimeToStr(Time());
SpeedButton1->Text="Parar";
horaInicio=Time();
}
else {
Label2->Text = "Hora final: " +
TimeToStr(Time());
SpeedButton1->Text="Empezar";
horaFin=Time();
crono=(horaFin-horaInicio)*100000;
Label3->Text="Crono: " +
FloatToStr(RoundTo(crono, -3));
}
}

```

Ejercicio. Creación de objetos en tiempo de ejecución. (Entorno firemonkey multidevice)

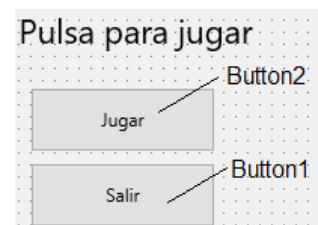
► Crear una nueva aplicación: *File - New - Multi-device application C++ Builder*
Para versiones anteriores de XE: *File - New - Firemonkey - Mobile Application C++ Builder*

- Establece la propiedad del formulario WindowState: wsMaximized (maximizado)
- Añade un Label y dos botones (TButton) como en la imagen al formulario
- En el botón2 de Jugar, Busca el Evento: On mouse enter y pulsa doble clic para tener el código: Button2MouseEnter

```

void __fastcall TForm5::Button2MouseEnter(TObject *Sender)
{
int aleatorioX=Random(Form5->Width-Button1->Width);
int aleatorioY=Random(Form5->Height-Button1->Height);
Button2->Position->X=aleatorioX;
Button2->Position->Y=aleatorioY;
}

```



► Pulsa doble clic sobre el botón Salir para escribir el código:

```

void __fastcall TForm5::Button1Click(TObject *Sender)
{
int i,j,k; //creo tres variables numéricas
TButton *miboton; //creo una variable del tipo botón en C# equivale a TButton miboton = New TButton

for (i = 0; i < 10; i++) {
miboton = new TButton(NULL); //le asigno la variable a un nuevo objeto de la clase tbotón
miboton->Parent=Form5; //le asigno un objeto padre contenedor
miboton->Width=100; //le asigno dimensiones
miboton->Height=50;
miboton->Text="Salir"; //le asigno un texto
j=Random(Form5->Width);
k=Random(Form5->Height);
miboton->Position->X=j; //le asigno a su posición las variables aleatorias
miboton->Position->Y=k;
miboton->OnClick=Button1Click; //le asigno un evento al botón
} }

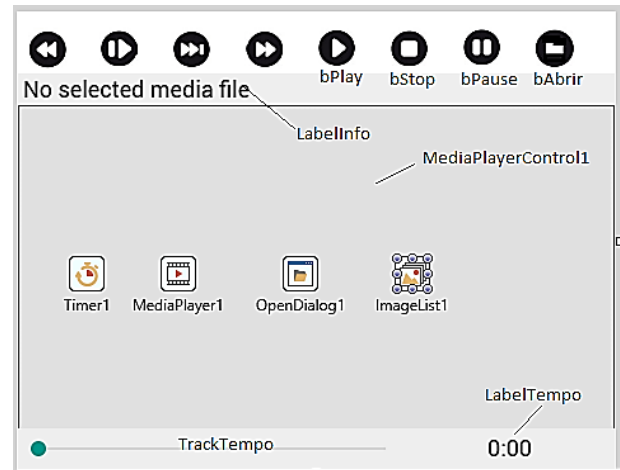
```

► Prueba la aplicación (▶ Run) Si funciona correctamente, guarda toda la aplicación: Fila - Save all.

El archivo unidad se llamará: **Juego1.cpp**, el proyecto **Juego.cbproj** y su archivo cabecera **JuegoCH1.h**

Reproductor multimedia en FMX

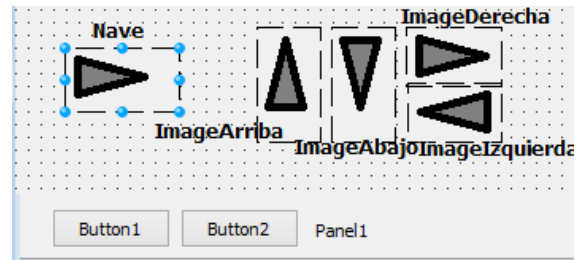
- Hacer una nueva *Multidevice application* – *C++ Builder*
 - Renombrar el formulario a *MusicForm*
 - Añadir las imágenes de los botones del reproductor en el componente *ImageList1* y asociar en los botones de la toolbar:
- ```
Images = ImageList1 - ImageIndex = 1
```
- En *MediaPlayerControl1*: *MediaPlayer* = *MediaPlayer1*
  - Nombrar los componentes como se muestran en la imagen.
  - Abajo tienes el código base.
  - Para finalizar, guarda el proyecto como *MusicPlayer.cproj*



```
void __fastcall TMusicForm::bAbrirClick(TObject *Sender)
{
 if (OpenDialog1->Execute()) {
 MediaPlayer1->FileName = OpenDialog1->FileName;
 LabelInfo->Text=OpenDialog1->FileName;
 TrackTempo->Max=MediaPlayer1->Duration;
 LabelTempo->Text="0:00/"+FloatToStr(MediaPlayer1->Duration);
 }
}
//-----
void __fastcall TMusicForm::TrackTempoTracking(TObject *Sender)
{
 MediaPlayer1->CurrentTime=TrackTempo->Value;
}
//-----
1. void __fastcall TMusicForm::bPlayClick(TObject *Sender)
{
String titulo=ExtractFileName(MediaPlayer1->FileName);
 if (titulo!="") {
 if (Sender==bPlay) {
 MediaPlayer1->Play();
 LabelInfo->Text=titulo+" - Playing";
 Timer1->Enabled=true;
 }
 else if (Sender==bStop) {
 MediaPlayer1->Stop();
 LabelInfo->Text=titulo+" - Stopping";
 }
 else if (Sender==bPause) {
 if (MediaPlayer1->State==1) //si State está en play...
 {
 MediaPlayer1->Stop();
 LabelInfo->Text=titulo+"- Pause";
 }
 else { //si State está en Stop...
 MediaPlayer1->Play();
 LabelInfo->Text=titulo+"- Playing";
 }
 }
 }
}
//-----
void __fastcall TMusicForm::Timer1Timer(TObject *Sender)
{
 double dura=MediaPlayer1->Duration;
 double tempo=MediaPlayer1->CurrentTime;
 TrackTempo->Value = MediaPlayer1->CurrentTime;
 TDateTime ttempo = tempo/1000000000000;
 TDateTime tdura = dura/1000000000000;
 LabelTempo->Text = TimeToStr(ttempo)+" / "+ TimeToStr(tdura);
}
}
```

## VCL: Eventos del teclado: dirección de nave.

- Dibujar previamente, desde Paint, 4 imágenes de una nave conteniendo las cuatro direcciones. Guardar en la carpeta **Naves**.
- Desde C++ Builder, crear una nueva aplicación escogiendo del menú: File - New – **VCL Form Application C++ Builder**.
- Cambiar la propiedad **Caption** del formulario por **Naves**, su propiedad **KeyPreview** por True y su propiedad **DoubleBuffered** por true.
- Guardar (File ▶ Save As) la unidad asociada al formulario (Unit1.cpp, por defecto) como **NaveUnit1.cpp**. El archivo cabecera a **Nave.h** y el proyecto (File | Save Project As) como **Naves.bpr** o **.cbproj** en la carpeta **Naves**.
- Añadir 5 componentes de imagen con los nombres de la figura. Cargar, desde su propiedad **Picture**, las imágenes previamente dibujadas.



```
void __fastcall TForm1::FormKeyDown(TObject *Sender, WORD &Key, TShiftState Shift)
{
 //ShowMessage(Key); para test de tecla
 switch (Key)
 {
 case 37: Nave->Left=Nave->Left-1; //izda
 break;
 case 38: Nave->Top=Nave->Top-1; //arriba
 break;
 case 39: Nave->Left=Nave->Left+1; //derecha
 break;
 case 40: Nave->Top=Nave->Top+1; //abajo
 }
}
```

### Variante ImageList:

- Borrar los 4 componentes de imagen y agregar un componente ImageList.
- En el componente ImageList, agregar las 4 imágenes de posición de la nave
- Cambiar el código anterior por:

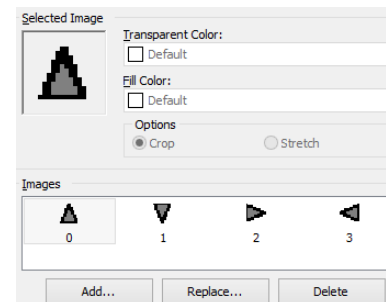
```
switch (Key)
{
 case 37: izquierda();
 break;
 case 38: arriba();
 break;
 case 39: derecha();
 break;
 case 40: abajo();
}
}
```

- Crear las funciones:


```
void __fastcall TForm1::izquierda()
{ImageList1->GetBitmap(2,Nave->Picture->Bitmap);
 Nave->Left=Nave->Left-1;}
void __fastcall TForm1::arriba()
{ImageList1->GetBitmap(0,Nave->Picture->Bitmap);
 Nave->Top=Nave->Top-1; }
void __fastcall TForm1::derecha()
{ImageList1->GetBitmap(3,Nave->Picture->Bitmap);
 Nave->Left=Nave->Left+1;}
void __fastcall TForm1::abajo()
{ImageList1->GetBitmap(1,Nave->Picture->Bitmap);
 Nave->Top=Nave->Top+1;}
```


- Declarar las funciones en el archivo cabecera h:

```
void __fastcall izquierda();
void __fastcall arriba();
void __fastcall derecha();
void __fastcall abajo();
```



**Ampliación: Uso de un procedimiento: llamar a una función con parámetros.**

Añadir  Shape de forma circle , llamarle: **bala** , visible:false

Añadir Timer  (de paleta system): Timer1 – Interval: 10

Declara la función en el archivo cabecera h: `void __fastcall mover(int direc);`

En el archivo cpp:

Añadir las variables públicas:

```
int direccion;
int velocidad;
```

Cambiar el código:

```
void __fastcall TForm1::FormCreate(TObject *Sender)
{
 ImageList1->GetBitmap(2,Nave->Picture->Bitmap);
 velocidad=2;
}
```

```
void __fastcall TForm1::FormKeyDown(TObject *Sender, WORD &Key, TShiftState Shift)
```

```
{
 switch (Key)
 {
 case 37: mover(3); //izquierda
 break;
 case 38: mover(0); //arriba
 break;
 case 39: mover(2); //derecha
 break;
 case 40: mover(1); //abajo
 break;
 case 32: dispara(); //barra espaciadora
 }
}
```

```
void __fastcall TForm1::mover(int direc)
```

```
{ImageList1->GetBitmap(direc,Nave->Picture->Bitmap);
 direccion=direc;
 switch (direccion)
 {
 case 3: Nave->Left=Nave->Left-velocidad;//izquierda();
 break;
 case 0: Nave->Top=Nave->Top-velocidad;//arriba();
 break;
 case 2: Nave->Left=Nave->Left+velocidad;//derecha();
 break;
 case 1: Nave->Top=Nave->Top+velocidad;//abajo();
 }
}
```

```
void __fastcall TForm1::dispara()
```

```
{
 bala->Left=Nave->Left;
 bala->Top=Nave->Top;
 bala->Visible=true;
 Timer1->Enabled=True;
}
```

```
void __fastcall TForm1::Timer1Timer(TObject *Sender)
```



```
{
 switch (direccion)
 {
 case 3: bala->Left=bala->Left-velocidad; //izquierda();
 break;
 case 0: bala->Top=bala->Top-velocidad; //arriba();
 break;
 case 2: bala->Left=bala->Left+velocidad; //derecha();
 break;
 case 1: bala->Top=bala->Top+velocidad; //abajo();
 }
 if ((bala->Left> Form1->Width) && (bala->Left<0) && (bala->Top<0) && (bala->Top>Form1->Height))
 {Timer1->Enabled=false;
 bala->Visible=false; }
}
```

**Trucos:**

- Para quitar el marco de la ventana: Selecciona el formulario y cambia las propiedades: `BorderStyle: bsnone` y `WindowState: wsMaximized`
- Para salir con tecla escape: Añadir el caso al procedimiento **FormKeyDown**:  
`case 27: Form1->Close(); //tecla escape = 27`

## FMX: Movimiento en pantalla (Uso del estamento CASE)

### Ingredientes:

- Shape  (de la Paleta Additional o Shapes): Name: Bola; Shape: Circle
- Timer  (de paleta system): Timer1 – Interval: 30
- Botones: 2 (Empezar y parar)
- Formulario: Si lo deseas, puedes asignar color transparente o negro (Fill: black kind: solid)



**Variables:** añadir al principio como variable pública:

```
Int mover=1;
```

### Acciones:

```
void __fastcall TForm1::BempezarClick(TObject *Sender) /* al pulsar el botón empezar
{
 mover=1;
 Timer1->Enabled=True; }
void __fastcall TForm1::BpararClick(TObject *Sender) /* al parar el botón
{
 Timer1->Enabled=False; }
```

**switch - Case:**  
Bifurca las acciones dependiendo de un parámetro del tipo ordinal

### Método utilizando las funciones swith - case

```
void __fastcall TForm1::Timer1Timer(TObject *Sender) /****** al activar el timer
{
 switch (mover)
 {
 case 1: // abajoder
 bola->Position->X++;
 bola->Position->Y++;
 if (bola->Position->X>=Form1->Width-bola->Width)
 {mover=2;};
 if (bola->Position->Y>=Form1->Height-bola->Height)
 {mover=3;};
 break;
 case 2: // abajoiz
 bola->Position->X--;
 bola->Position->Y++;
 if (bola->Position->X<=0) {mover=1;};
 if (bola->Position->Y>=Form1->Height-bola->Height)
 {mover=4;};
 break;
 case 3: // arribader
 bola->Position->X++;
 bola->Position->Y--;
 if (bola->Position->X>=Form1->Width-bola->Width)
 {mover=4;};
 if (bola->Position->Y<=0) {mover=1;};
 break;
 case 4: // arribaiz
 bola->Position->X--;
 bola->Position->Y--;
 if (bola->Position->X<=0) {mover=3;};
 if (bola->Position->Y<=0) {mover=2;};
 }
}
```

### Método utilizando if funciones creadas

```
void __fastcall TForm1::Timer1Timer(TObject *Sender)
{
 if (mover==1) {abajoder();}
 else if (mover==2) {abajoiz();}
 else if (mover==3) {arribader();}
 else if (mover==4) {arribaiz();}
}
//*****
void __fastcall TForm1::abajoder()
{
 bola->Position->X++;
 bola->Position->Y++;
 if (bola->Position->X>=Form1->Width-bola->Width)
 {mover=2;};
 if (bola->Position->Y>=Form1->Height-bola->Height)
 {mover=3;};
}
void __fastcall TForm1::abajoiz()
{
 bola->Position->X--;
 bola->Position->Y++;
 if (bola->Position->X<=0) {mover=1;};
 if (bola->Position->Y>=Form1->Height-bola->Height)
 {mover=4;};
}
void __fastcall TForm1::arribader()
{
 bola->Position->X++;
 bola->Position->Y--;
 if (bola->Position->X>=Form1->Width-bola->Width)
 {mover=1;};
 if (bola->Position->Y<=0) {mover=3;};
}
void __fastcall TForm1::arribaiz()
{
 bola->Position->X--;
 bola->Position->Y--;
 if (bola->Position->X<=0) {mover=3;};
 if (bola->Position->Y<=0) {mover=2;};
}
```

Recuerda declarar en el archivo cabecera (.h) las funciones:

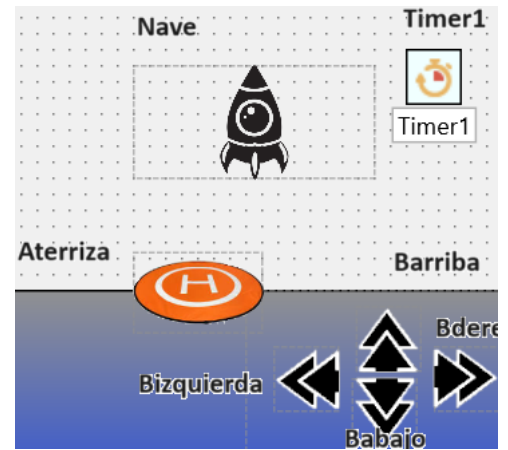
```
void __fastcall abajoder(); void __fastcall abajoiz();
void __fastcall arribaiz(); void __fastcall arribader();
```

## FMX dirección de nave Multidevice

### ▸ Código Nave para Multidevice

```
void __fastcall TFormNave::FormKeyDown
switch(Key)
{
 case 27:{
 ShowMessage("Desea salir del juego?");
 }
 break;
 case 37:{ //flecha izda
 izquierda();
 }
 break;
 case 38: { //flecha arriba
 arriba();
 }
 break;
 case 39: { //flecha derecha
 derecha();
 }
 break;
 case 40: { //flecha abajo
 abajo();
 }
 break;
}
}
```

```
void __fastcall TFormNave::izquierda()
{
 Nave->RotationAngle=Nave->RotationAngle-10;
 VelH--;
}
void __fastcall TFormNave::derecha()
{
 Nave->RotationAngle=Nave->RotationAngle+10;
 VelH++;
}
void __fastcall TFormNave::arriba()
{
 if (Nave->RotationAngle>3) {
 Nave->RotationAngle=Nave->RotationAngle-5;
 }
 else if (Nave->RotationAngle<-3){
 Nave->RotationAngle=Nave->RotationAngle+5;
 }
 VelV--;
}
void __fastcall TFormNave::abajo()
{
 if (Nave->RotationAngle>3) {
 Nave->RotationAngle=Nave->RotationAngle-5;
 }
 else if (Nave->RotationAngle<-3){
 Nave->RotationAngle=Nave->RotationAngle+5;
 }
 VelV++;
}
void __fastcall TFormNave::BarribaClick(TObject *Sender)
{
 arriba();
}
void __fastcall TFormNave::BabajoClick(TObject *Sender)
{
 abajo();
}
void __fastcall TFormNave::BderechaClick(TObject *Sender)
{
 derecha();
}
void __fastcall TFormNave::BizquierdaClick(TObject *Sender)
{
 izquierda();
}
```



```
void __fastcall TFormNave::Timer1Timer(...)
{
 Nave->Position->X=Nave->Position->X+VelH;
 Nave->Position->Y=Nave->Position->Y+VelV;
 if (Nave->Position->X>=FormNave->Width) {
 Nave->Position->X=0;
 }
 if (Nave->Position->Y>=FormNave->Height) {
 Nave->Position->Y=0;
 }
 if (Nave->Position->X<0) {
 Nave->Position->X=FormNave->Width;
 }
 if (Nave->Position->Y<0) {
 Nave->Position->Y=FormNave->Height;
 }
}
```

En la unidad Cabezera Unit1.h, añadir la declaración de las funciones

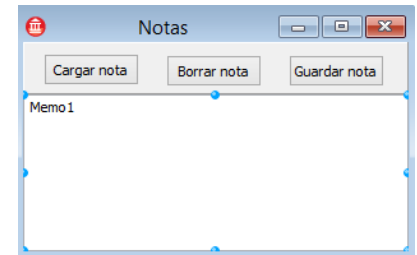
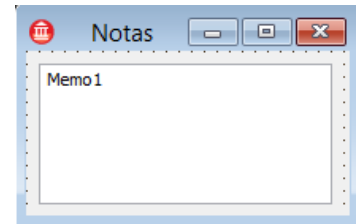
```
void __fastcall Timer1Timer(TObject *Sender);
void __fastcall izquierda();
void __fastcall derecha();
void __fastcall arriba();
void __fastcall abajo();
void __fastcall BarribaClick(TObject *Sender);
void __fastcall BabajoClick(TObject *Sender);
void __fastcall BderechaClick(TObject *Sender);
void __fastcall BizquierdaClick(TObject *Sender);
```

## USO DE COMPONENTES DE TEXTO. MEMO

Para mostrar y editar varias líneas de texto, disponemos de los componentes: Memo (paleta estándar) para texto sencillo TXT y Richedit (paleta Win32) que permite almacenar texto enriquecido (con formato).

1. Elige del menú principal: File ► New ► VCL form Application.
2. Con el formulario seleccionado, selecciona del inspector de objetos el evento **OnCreate** haciendo doble clic y escribe lo siguiente:
 

```
Memo1->Lines->LoadFromFile("Unit1.cpp");
```
3. Selecciona de la paleta Standard, el objeto Memo, y extiéndelo sobre el formulario, alineado al cliente, con el nombre: *Memo1*.
4. Guardar: En una carpeta nueva llamada *Memo*, guarda la unidad con el nombre: *Unit1.cpp* y el proyecto con el nombre: *Memo*
5. Ejecutar: Pulsa en (Run) y comprueba que al iniciar la aplicación, se muestra el texto de la unidad.



### Variación:

- Recupera el proyecto anterior (File ► Open)
- Añade tres botones dentro un panel alineado al top. (Antes deberás desajustar el Memo) Con el texto: Cargar Nota, Borrar nota y Guardar nota.
- Anula el código que añadimos en la Función **OnCreate**
- Asigna el código a los botones:

```
void __fastcall TNotas::Button1Click(TObject *Sender) → { Memo1->Lines->LoadFromFile("nota.txt"); }
void __fastcall TNotas::Button2Click(TObject *Sender) → { Memo1->Lines->Clear(); }
void __fastcall TNotas::Button3Click(TObject *Sender) → { Memo1->Lines->SaveToFile("nota.txt"); }
```

- Comprueba su funcionamiento (Run) y Guarda todo (Save All)

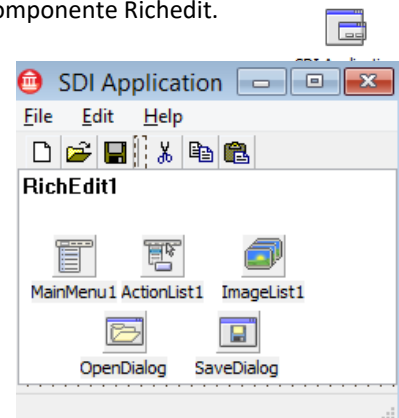
### Ejercicio propuesto **FireMonkey**: Editor de Notas para *FireMonkey Mobile*

- Repite el ejercicio en una nueva aplicación: File – New - Firemonkey – Mobile Application C++ Builder  
Diseño: Header footer

## USO DE COMPONENTES DE TEXTO. RICHEDIT y ActionList

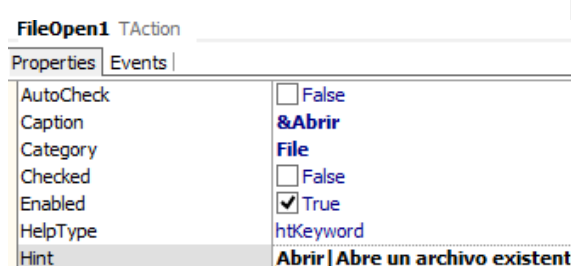
Vamos a generar un pequeño procesador de textos utilizando la plantilla SDI y el componente Richedit.

1. Elige del menú principal: File ► New ► Other...
2. Escoge de *C++ Projects: SDI Application*. Nos generará un formulario con los componentes básicos. Un menú, una barra de herramientas y una barra de estado.
3. Añade al formulario un objeto Richedit (Paleta Win32) alineado al cliente.  
Cambia su propiedad Align: alCliente y su borra el texto de la propiedad Lines  
La plantilla nos ha generado un formulario con una barra de herramientas y un menú que contienen llamadas a las mismas acciones, por ejemplo, la acción de Abrir puede ser llamada desde el botón de abrir o desde el menú: Archivo - Abrir. Cada objeto de ambas, hace referencia a una acción que está centralizada en la lista de acciones: ActionList1 y es llamado desde el evento **Action**.



Para completar el evento Abrir, pulsamos doble click sobre **Acciónlist1** y seleccionamos a acción: **FileOpen1**.

En sus propiedades: cambia sus propiedades *Caption* y *Hint* como en la figura.



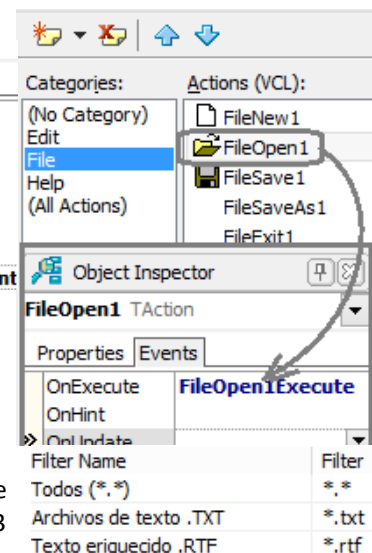
En el evento **OnExecute**, cambia el código: OpenFileDialog->Execute(); por:

```
if (OpenDialog->Execute())
{ RichEdit1->Lines->LoadFromFile(OpenDialog->FileName); }
```

Del mismo modo selecciona la Acción **FileSave1** y cambia el código por:

```
if (SaveDialog->Execute())
{ RichEdit1->Lines->SaveToFile(SaveDialog->FileName); }
```

En ambos objetos, cambiar la propiedad filter, para poder especificar e tipo de documento al abrir o guardar, como en la figura y las propiedades: Filterindex: 3 DefaultExt: \*.rtf



**Otras acciones:**

- Acciones **Cortar**, **copiar** y **pegar**: Estas acciones funcionan sin código escrito porque estás asociadas a una combinación del teclado que se designa en la propiedad: **Shortcut**: Ctrl+X, Ctrl+C y Ctrl+V.
- Acción **Nuevo**: FileNew1 - FileNew1Execute: añade el código: `RichEdit1->Lines->Clear();`
- Acción **Salir**: FileExit1 - FileExit1Execute. Comprueba el código: `Close();` ó `Application->Terminate();`
- Acción **Acrescade** (Créditos): HelpAbout1 - HelpAbout1Execute. Comprueba el código: `AboutBox->ShowModal();`

**Problemas y mejoras:****Nombre de archivo al guardar:**

Al escoger Guardar la aplicación **no** nos debería solicitar el nombre si ya se ha guardado o recuperado antes:

- Creamos una variable pública que guardará el nombre del archivo: `String nombreadchivo = "";`
- Cambiamos el código del procedimiento FileSave1Execute:
 

```
void __fastcall TSDIAppForm::FileSave1Execute(TObject *Sender)
{
 if (Sender==FileSave1) {
 if (nombreadchivo!="")
 {RichEdit1->Lines->SaveToFile(nombreadchivo);
 RichEdit1->Modified = False; }
 }
 else
 {
 if (SaveDialog->Execute()){
 RichEdit1->Lines->SaveToFile(SaveDialog->FileName);
 nombreadchivo=SaveDialog->FileName; }
 }
}

```
- Añadimos el código al Abrir: `nombreadchivo=OpenDialog->FileName;`

**Preguntar al cerrar o nuevo:**

Tanto si creamos un archivo nuevo o salimos del programa, la aplicación nos debería preguntar si deseamos conservar o guardar el documento editado. Para ello utilizamos la propiedad Modified del componente Richedit.

```
void __fastcall TSDIAppForm::FileNew1Execute(TObject *Sender)
{
 if (RichEdit1->Modified==True) {
 if (MessageDlg("¿Guardar documento anterior?",
 mtInformation, TMsgDlgButtons() << mbYes << mbNo << mbCancel, 0)== mrYes)
 { FileSave1Execute(Sender);}
 }
 RichEdit1->Lines->Clear();
 RichEdit1->Modified=False;
}

```

Al cerrar el formulario ocurre antes el evento **FormCloseQuery** donde se consulta antes de cerrar la variable CanClose:

```
void __fastcall TSDIAppForm::FormCloseQuery(TObject *Sender, bool &CanClose)
{
 if (RichEdit1->Modified==True) {
 if (MessageDlg("¿Guardar documento?", mtInformation, TMsgDlgButtons() << mbYes << mbNo << mbCancel, 0)==
 mrYes) {FileSave1Execute(Sender);}
 CanClose=True;}
}

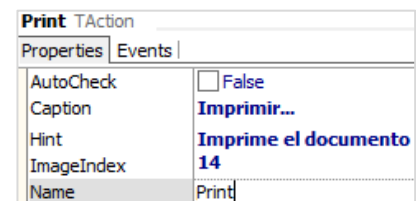
```

**Imprimir:**

- Añade un objeto **PrintDialog1** de la paleta Dialogs.
- Añade una nueva acción a la lista **ActionList1** para imprimir, con las propiedades de la imagen al que le añadiremos la siguiente acción en el evento OnExecute:
 

```
if (PrintDialog1->Execute()){ RichEdit1->Print(nombreadchivo); }

```
- Añade un elemento en la sección file del MainMenu y un nuevo botón en la ToolBar1 de modo que su acción **Action** se vincule a: Print.
- Para generar la acción Configurar Impresora, puedes crear una Standard Action del tipo: FilePrintSetup. Luego puedes añadir el elemento en el menú para abrir el cuadro de diálogo: Configurar impresora:



**Ejecuta y prueba (Run).** Si todo es correcto, Guarda todo (File ► Save All):

Unit: Editor1 - Cabezera: Editor1h - Proyecto: **Editor**

## Otras variantes de texto:

Código común habitual llamado desde el menú para uso con aplicaciones con texto memo.

En este caso se ha llamado al memo: TextMemo

### Al escoger archivo nuevo (New)

| Para Delphi                                                                                                                                                                                                                                                                                                                                                                                        | Para C++                                                                                                                                                                                                                                                                                                                                                                                                       |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre> procedure TTextEditorForm.NewExecute(Sender: TObject); var   UserResponse : Integer; begin   // Check TMemo number of lines property   if TextMemo.Lines.Count &gt; 0 then   begin     UserResponse := MessageDlg( 'Nuevo documento?',     mtInformation, mbYesNo, 0);     if UserResponse = mrYes then     begin       TextMemo.Clear;       CurrentFile := '';     end;   end; end; </pre> | <pre> void __fastcall TTextEditorForm::NewExecute(TObject *Sender) {   // Check TMemo number of lines property   if (TextMemo-&gt;Lines-&gt;Count &gt; 0)   {     int userResponse = MessageDlg(     String("Nuevo documento?", mtInformation,     TMsgDlgButtons() &lt;&lt; mbYes &lt;&lt; mbNo, 0);      if (userResponse == mrYes) {       TextMemo-&gt;Clear();       currentFile = "";     }   } } </pre> |

### Al abrir (Open)

| Para Delphi                                                                                                                                                                                                                                                                                                                                                                              | Para C++                                                                                                                                                                                                                                                                                                                                                                    |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre> procedure TTextEditorForm.FileOpen1Accept(Sender: TObject); var   FileName: String; begin   // Get file name from TFileOpen component   FileName := FileOpen1.Dialog.FileName;   if FileExists(FileName) then   begin     TextMemo.Lines.LoadFromFile(FileName);     CurrentFile := FileName;     Self.Caption := 'Text Editor - ' + ExtractFileName(FileName);   end; end; </pre> | <pre> void __fastcall TTextEditorForm::FileOpen1Accept(TObject *Sender) {   // Get file name from TFileOpen component   String fileName = FileOpen1-&gt;Dialog-&gt;FileName;   if (FileExists(fileName)) {     TextMemo-&gt;Lines-&gt;LoadFromFile(fileName);     currentFile = fileName;     this-&gt;Caption = "Text Editor - " + ExtractFileName(fileName);   } } </pre> |

### Al Guardar como

| Para Delphi                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | Para C++                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre> procedure TTextEditorForm.FileSaveAs1Accept(Sender: TObject); var   FileName: String;   UserResponse : Integer; begin   // Get file name from TFileSaveAs component   FileName := FileSaveAs1.Dialog.FileName;    if FileExists(FileName) then   begin     UserResponse := MessageDlg(     'File already exists. ' +     'Do you want to overwrite?', mtInformation,     mbYesNo, 0);     if UserResponse = mrNo then     Exit();   end;    TextMemo.Lines.SaveToFile(FileName);   CurrentFile := FileName;   Self.Caption := ExtractFileName(FileName); end; </pre> | <pre> void __fastcall TTextEditorForm::FileSaveAs1Accept(TObject *Sender) {   // Get file name from TFileSaveAs component   String fileName = FileSaveAs1-&gt;Dialog-&gt;FileName;    if (FileExists(fileName)) {     int userResponse = MessageDlg(     String( "File already exists. " ) +     "Do you want to overwrite?", mtInformation,     TMsgDlgButtons() &lt;&lt; mbYes &lt;&lt; mbNo, 0);     if (userResponse == mrNo) {       return;     }   }    TextMemo-&gt;Lines-&gt;SaveToFile(fileName);   currentFile = fileName;   this-&gt;Caption = ExtractFileName(fileName); } </pre> |

**Al Guardar (Save)**

| Para Delphi                                                                                                                                                                    | Para C++                                                                                                                                                                                             |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre>procedure TTextEditorForm.SaveExecute(Sender: TObject); begin if CurrentFile = '' then Self.FileSaveAs1.Execute() else TextMemo.Lines.SaveToFile(CurrentFile); end;</pre> | <pre>void __fastcall TTextEditorForm::SaveExecute(TObject *Sender) { if (currentFile == "") { this-&gt;FileSaveAs1-&gt;Execute(); } else { TextMemo-&gt;Lines-&gt;SaveToFile(currentFile); } }</pre> |

**Al escoger Fuente de letra**

| Para Delphi                                                                                                                                          | Para C++                                                                                                                                                       |
|------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre>procedure TTextEditorForm.FontEdit1Accept(Sender: TObject); begin // Set TMemo font property TextMemo.Font := FontEdit1.Dialog.Font; end;</pre> | <pre>void __fastcall TTextEditorForm::FontEdit1Accept(TObject *Sender) { // Set TMemo font property TextMemo-&gt;Font = FontEdit1-&gt;Dialog-&gt;Font; }</pre> |

**Al ajustar**

| Para Delphi                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           | Para C++                                                                                                                                                                                                                                                                                                                                                                                                    |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre>procedure TTextEditorForm.WordWrapExecute(Sender: TObject); begin { Toggle the word wrapping state. } TextMemo.WordWrap := not TextMemo.WordWrap; WordWrap.Checked := TextMemo.WordWrap; if TextMemo.WordWrap = True then { Only vertical scrollbars are needed when word wrapping is set. } TextMemo.ScrollBars := ssVertical else Copyright 2010 Embarcadero Technologies, Inc.Coding responses to user actions in the Code Editor (IDE Tutorial) 57 TextMemo.ScrollBars := ssBoth; end;</pre> | <pre>void __fastcall TTextEditorForm::WordWrapExecute(TObject *Sender) { // Toggle the word wrapping state. TextMemo-&gt;WordWrap = !TextMemo-&gt;WordWrap; WordWrap-&gt;Checked = TextMemo-&gt;WordWrap; if (TextMemo-&gt;WordWrap == True) { // Only vertical scrollbars are needed when word wrapping is set. TextMemo-&gt;ScrollBars = ssVertical; } else { TextMemo-&gt;ScrollBars = ssBoth; } }</pre> |

**Al bajar mouse en el texto info en la barra de estado**

| Para Delphi                                                                                                                                                                                                                                                                                                                                                                                | Para C++                                                                                                                                                                                                                                                                                                                                                                                                                               |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre>procedure TTextEditorForm.TextMemoMouseDown(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer); begin TextStatus.Panels.Items[ 0].Text := 'L: ' + IntToStr(TextMemo.CaretPos.Y + 1); TextStatus.Panels.Items[ 1].Text := 'C: ' + IntToStr(TextMemo.CaretPos.X + 1); TextStatus.Panels.Items[ 2].Text := 'Lines: ' + IntToStr(TextMemo.Lines.Count); end;</pre> | <pre>void __fastcall TTextEditorForm::TextMemoMouseDown(TObject *Sender, TMouseButton Button, TShiftState Shift, int X, int Y) { TextStatus-&gt;Panels-&gt;Items[ 0]-&gt;Text = "L: " + String (TextMemo-&gt;CaretPos.y + 1); TextStatus-&gt;Panels-&gt;Items[ 1]-&gt;Text = "C: " + String (TextMemo-&gt;CaretPos.x + 1); TextStatus-&gt;Panels-&gt;Items[ 2]-&gt;Text = "Lines: " + IntToStr (TextMemo-&gt;Lines-&gt;Count); }</pre> |

**Al bajar mouse o hacer clic en el texto info en la barra de estado**

| Para Delphi                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | Para C++                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre>procedure TTextEditorForm.TextMemoMouseDown(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer); begin TextStatus.Panels.Items[ 0].Text := 'L: ' + IntToStr(TextMemo.CaretPos.Y + 1); TextStatus.Panels.Items[ 1].Text := 'C: ' + IntToStr(TextMemo.CaretPos.X + 1); TextStatus.Panels.Items[ 2].Text := 'Lines: ' + IntToStr(TextMemo.Lines.Count); end; procedure TTextEditorForm.TextMemoKeyDown(Sender: TObject; var Key: Word; Shift: TShiftState); Copyright 2010 Embarcadero Technologies, Inc.Coding responses to user actions in the Code Editor (IDE Tutorial) 58 begin // Perform same action as for mouse down in TMemo TextMemoMouseDown(Sender, mbLeft, Shift, 0, 0); end;</pre> | <pre>void __fastcall TTextEditorForm::TextMemoMouseDown(TObject *Sender, TMouseButton Button, TShiftState Shift, int X, int Y) { TextStatus-&gt;Panels-&gt;Items[ 0]-&gt;Text = "L: " + String (TextMemo-&gt;CaretPos.y + 1); TextStatus-&gt;Panels-&gt;Items[ 1]-&gt;Text = "C: " + String (TextMemo-&gt;CaretPos.x + 1); TextStatus-&gt;Panels-&gt;Items[ 2]-&gt;Text = "Lines: " + IntToStr (TextMemo-&gt;Lines-&gt;Count); } void __fastcall TTextEditorForm::TextMemoKeyDown(TObject *Sender, WORD &amp;Key, TShiftState Shift) { // Perform same action as for mouse down in TMemo TextMemoMouseDown(Sender, mbLeft, Shift, 0, 0); }</pre> |

## Una aplicaciones de consola.

Una aplicación de consola no utiliza los objetos de la VCL, y el aspecto de una aplicación de este tipo es el generado por los compiladores "tradicionales", ejecutándose sobre una consola MS-DOS.

C++ Builder permite la creación de este tipo de aplicaciones,. Veamos cómo.

Una aplicación de consola se crea usando el asistente *Console Wizard*.

- Para versiones antiguas: Selecciona: *File | New | Console Wizard* y en el asistente, indicar que se va a crear una aplicación de consola (*Console*).
- Para versiones nuevas: *File - New – Other – Console application*

En la unidad cpp, borra el texto existente y añade el siguiente código:

```
#include <iostream.h>
#include <conio.h>

int main(int argc, char* argv[])
{
 for (int i=0; i<5; i++)
 cout << "Valor de i = " << i << endl;
 getch();
 return 0;
}
```

```
Valor de i = 0
Valor de i = 1
Valor de i = 2
Valor de i = 3
Valor de i = 4
```

Finalmente, guardar el proyecto y ejecutar el programa. Seleccionar *File | Save Project As* y guardar como *Console.bpr*. Ahora podemos construir el ejecutable y probarlo: seleccionar *Project | Build* y a continuación, *Run | Run* o pulsa en: ►

## Juego de las tres en raya. Matrices: (modo consola)

1º Recordamos el código fuente del curso anterior:

```
#include <stdio.h> //-> printf (), scanf ()
#include <stdlib.h> //-> exit ()
#define ESP ' ' //-> ESPACIO en blanco
char matriz[3][3] =
{
 ESP, ESP, ESP,
 ESP, ESP, ESP,
 ESP, ESP, ESP
};
void obtener_movimiento_de_jugador (void);
void obtener_movimiento_de_computadora (void);
void mostrar_matriz (void);
char comprobar (void); //-> función retorna char
int main (void)
{
 char hecho = ESP;
 printf ("JUEGO DE LAS TRES EN RAYA.\n\n");
 do
 {
 mostrar_matriz ();
 obtener_movimiento_de_jugador ();
 hecho = comprobar (); /* ver si gana el jugador */
 if (hecho != ESP) break;
 obtener_movimiento_de_computadora ();
 hecho = comprobar (); /* ver si gana la computadora */
 } while (hecho == ESP);

 if (hecho == 'X')
 printf ("\n *** ¡HAS GANADO! ***\n");
 else
 printf ("\n *** ¡YO GANO! ***\n");

 mostrar_matriz (); /* mostrar las posiciones finales */
 printf ("\n\nPulsa cualquier tecla para finalizar. ");
 scanf ("%d");
}

void obtener_movimiento_de_jugador (void)
{
 int x, y;
 printf ("\nIntroduzca sus coordenadas de la X (fila, columna): ");
 scanf ("%d%d", &x, &y);
 x--;
 y--;
 if (matriz[x][y] != ESP)
 {
 printf ("Movimiento inválido, prueba de nuevo.\n");
 obtener_movimiento_de_jugador ();
 }
 Else matriz[x][y] = 'X';
}

void obtener_movimiento_de_computadora (void)
```

```
{
 int encontrado = 0;
 int i, j;
 for (i = 0; ! encontrado && i < 3; i++)
 for (j = 0; ! encontrado && j < 3; j++)
 if (matriz[i][j] == ESP)
 encontrado = 1;
 if (encontrado) matriz[i-1][j-1] = 'O';
 else
 {
 printf ("Tablero completo.\n");
 exit (0);
 }
}

void mostrar_matriz (void)
{
 int i;
 printf ("\n 1 2 3");
 printf ("\n +-----+");
 for (i = 0; i < 3; i++)
 {
 printf ("\n%d | %c | %c | %c |", i+1, matriz[i][0], matriz[i][1],
 matriz[i][2]);

 if (i != 2)
 printf ("\n |---+---+---|");
 }
 printf ("\n +-----+");
}

char comprobar (void)
{
 int t;
 /* comprobar filas */
 for (t = 0; t < 3; t++)
 if (matriz[t][0] == matriz[t][1] && matriz[t][1] == matriz[t][2])
 return matriz[t][0];
 /* comprobar columnas */
 for (t = 0; t < 3; t++)
 if (matriz[0][t] == matriz[1][t] && matriz[1][t] == matriz[2][t])
 return matriz[0][t];
 /* comprobar diagonal principal */
 if (matriz[0][0] == matriz[1][1] && matriz[1][1] == matriz[2][2])
 return matriz[0][0];
 /* comprobar diagonal inversa */
 if (matriz[0][2] == matriz[1][1] && matriz[1][1] == matriz[2][0])
 return matriz[0][2];
 return ESP;
}
```

## Juego de las tres en raya. (modo Windows VCL)

### Diseño:

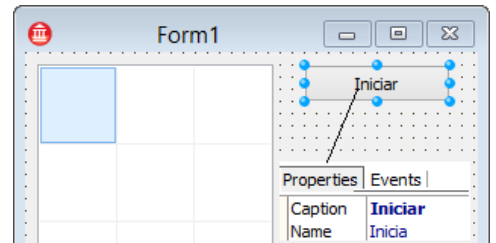
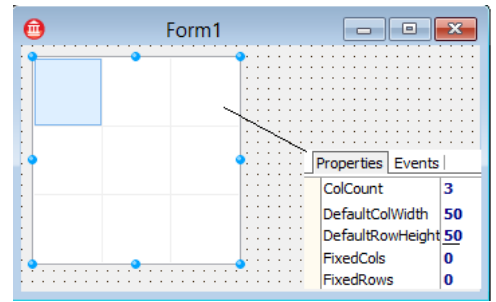
- Iniciamos nuevo proyecto: File – New – VCL Forms application C++
- De la paleta *Adicional*, añadimos una rejilla **StrinGrid**.

Cambiamos sus propiedades:

Name: Tablero  
 ColCount t RowCount: 3  
 FixedCol y FixedRow: 0  
 ScrollBars: None  
 DefaultColWith y DefaultColHeight:50

- De la Paleta *Standard*, añadimos un componente **tButton** y cambiamos sus propiedades:

Name: Inicia  
 Caption: Iniciar



### Código:

- Evaluar Movimiento del jugador: Pulsamos doble clic en el evento **OnSelectCell** del tablero para activar el procedure:  
 TableroSelectCell(TObject \*Sender, int ACol, int ARow, bool &CanSelect)

- El tablero es una rejilla (grid) que empieza a contar desde cero primero la columna y luego la fila. Así que intercambiamos nuestras variables f y c de fila y columna por las Arow y Acol, que nos devuelve el procedure, por simplicidad.
- El usuario hace clic sobre una celda de la *grid* que comparamos con ESP para ver si está vacía. En caso contrario, informamos que reinte.

```
{
int f,c;

f=ACol; //-> número de columna recogido por parámetro del procedure
c=ARow; //-> número de fila -> se intercambian por columnas

if (Tablero->Cells[f][c] == ESP)
{
Tablero->Cells[f][c]='X';
}
else
{
ShowMessage(String("No vale, intenta de nuevo"));
}
}
```

- Iniciar el Tablero: En el Form1 pulsa doble clic sobre el evento del Button **Inicia** para desencadenar la función: **IniciaClick**  
 Aquí rellenamos el tablero de espacios en blanco para luego poder comparar con algo. Para ello utilizamos dos contadores para incrementar fila y columna en la rejilla (grid) del tablero.

- Añade, al principio del código, la directiva:  
 #define ESP ' ' //-> ESPACIO en blanco

```
void __fastcall TForm1::IniciaClick(TObject *Sender)
{
int i,j;
for (i = 0; i < 3; i++) {
for (j = 0; j < 3; j++) {
Tablero->Cells[i][j]=ESP;
}
}
}
```

- Sobre el *Form1* activa (doble clic) el evento Onshow:  
 Este evento se desencadena al mostrarse el formulario, por lo tanto, llamamos aquí también a la función **IniciaClick** para que ya se nos prepare el tablero al iniciarse la aplicación: →

```
void __fastcall TForm1::FormShow(TObject
*Sender)
{
IniciaClick(Sender);
}
```

### Primera prueba de ejecución:

Guarda la unidad y el proyecto con los nombres: *tresenraya1.cpp* y *tresenraya.cbproj*  
 Para evitar cuelgues del programa, ejecuta primero con debug, pulsando en Run o F9.

### Programa completo:

En el archivo de cabecera *tresenraya.h* se han declarado las funciones automáticas, pero deberemos declarar nosotros algunas manualmente:

```
void __fastcall TableroSelectCell(TObject *Sender, int ACol, int ARow, bool &CanSelect);
void __fastcall IniciaClick(TObject *Sender);
void __fastcall FormShow(TObject *Sender);
char __fastcall Comprobar(char letra); -> Procedimiento/función manual
char __fastcall Obtener_movimiento_computadora(); -> Procedimiento/función manual
```

**Código completo TresenRaya Visual:** En el código **tresenraya1.cpp**, escribiremos lo siguiente:

```

#include <vc1.h>
#pragma hdrstop
#include "tresenraya1.h"
//-----
#pragma package(smart_init)
#pragma resource "*.dfm"
#define ESP ' ' //-> ESPACIO en blanco
TForm1 *Form1;
//-----
__fastcall TForm1::TForm1(TComponent* Owner)
: TForm(Owner)
{
}
//-----
void __fastcall TForm1::TableroSelectCell(TObject
*Sender, int ACol, int ARow, bool &CanSelect)
{
int f,c;
char ganar = ESP;

f=ACol; //-> número de columna recogido por parámetro
c=ARow; //-> número de fila -> se intercambian por
columnas

if (Tablero->Cells[f][c] == ESP)
{
Tablero->Cells[f][c]='X';
ganar = Comprobar('X');
if (ganar=='X') {
ShowMessage(String("Has ganado"));
}
else {
Obtener_movimiento_computadora();
ganar = Comprobar('O');
if (ganar=='O')
{
ShowMessage(String("Yo he ganado"));
}
}
}
else
{
ShowMessage(String("No vale, intenta de nuevo"));
}
}
//-----
void __fastcall TForm1::IniciaClick(TObject *Sender)
{
int i,j;
for (i = 0; i < 3; i++) {
for (j = 0; j < 3; j++) {
Tablero->Cells[i][j]=ESP;
}
}
}
//-----
void __fastcall TForm1::FormShow(TObject *Sender)
{
IniciaClick(Sender);
}
//-----
char __fastcall TForm1::Comprobar(char letra)
{
int t;
/* comprobar filas */
for (t = 0; t < 3; t++)
if (Tablero->Cells[t][0] == Tablero->Cells[t][1] &&
Tablero->Cells[t][1] == Tablero->Cells[t][2]
&& Tablero->Cells[t][2]==letra) return letra;
/* comprobar columnas */
for (t = 0; t < 3; t++)
if (Tablero->Cells[0][t] == Tablero->Cells[1][t] &&
Tablero->Cells[1][t] == Tablero->Cells[2][t]
&& Tablero->Cells[2][t]==letra) return letra;
/* comprobar diagonal principal */
if (Tablero->Cells[0][0] == Tablero->Cells[1][1] &&
Tablero->Cells[1][1] == Tablero->Cells[2][2]
&& Tablero->Cells[2][2]==letra) return letra;
/* comprobar diagonal inversa */
if (Tablero->Cells[0][2] == Tablero->Cells[1][1] &&
Tablero->Cells[1][1] == Tablero->Cells[2][0]
&& Tablero->Cells[2][0]==letra) return letra;
}
//-----
char __fastcall
TForm1::Obtener_movimiento_computadora()
{
int encontrado = 0;
int i, j;
for (i = 0; ! encontrado && i < 3; i++)
for (j = 0; ! encontrado && j < 3; j++)
if (Tablero->Cells[i][j] == ESP)
encontrado = 1;
if (encontrado) Tablero->Cells[i-1][j-1] = 'O';
else
{
ShowMessage(String("Tablero completo"));
// exit (0);
}
}
}

```

## Ejemplo Labels. Para VCL: Aleatorio y uso del Sender

Simula un juego de dados. Al pulsar sobre el botón realiza una tirada de los dados de forma aleatoria, y si se pulsa sobre cualquiera de las etiquetas se ha de generar una nueva tirada sobre la etiqueta que se haya pulsado.

Diseñar el formulario con el siguiente aspecto:

Al realizar el clic sobre el botón, los caption de las etiquetas mostrarán un numero aleatorio entre 1 y 6.

Evento para los 2 botones:

Botón1 - Jugador 1

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
 /*valores aleatorios entre 1 y 6 para las etiquetas*/
 randomize(); //genera la semilla
 Label1->Caption = (rand()%6)+1 ;
 Label2->Caption = (rand()%6)+1 ;
 Label3->Caption = (rand()%6)+1 ;
 Label4->Caption = (rand()%6)+1 ;
 Label5->Caption = (rand()%6)+1 ;
}

```

Evento para las etiquetas:

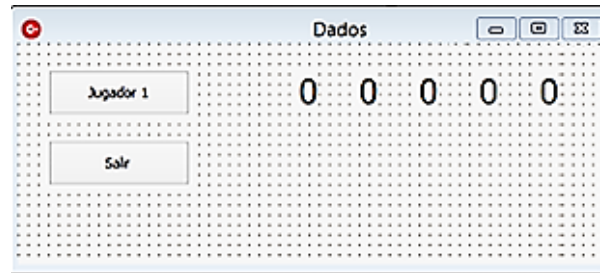
Al realizar un clic sobre una, se ha de generar una tirada aleatoria.

Para realizar este proceso programamos el evento onClick de la primera etiqueta y luego, asociaremos el evento a las otras etiquetas, utilizando el parámetro "Sender". Para hacerse pasar por una Label, ha de realizar un "casting" con la orden dynamic\_cast.

void \_\_fastcall TForm1::Label1Click(TObject \*Sender)

```
{
 randomize(); // genera un numero aleatorio para el dado seleccionado.
 dynamic_cast <TLabel *>(Sender)->Caption=(rand()%6)+1; //conversion dinamica de Sender a tipo Label
}

```



Botón 2 - Salir

```
void __fastcall TForm1::Button2Click(TObject *Sender)
{
 Form1->Close(); // finaliza la ejecución del programa
}

```



Una vez finalizada la programación de los eventos correspondientes, al ejecutar el programa se ha de visualizar un formulario similar al que se muestra a continuación.

### Apagar el sistema:

Llamar a la función SistemaShutdown() que se muestra a continuación:

```
bool TForm::SistemaShutdown()
{
 HANDLE hToken;
 TOKEN_PRIVILEGES tkp;
 if (!OpenProcessToken(GetCurrentProcess(),
 TOKEN_ADJUST_PRIVILEGES | TOKEN_QUERY, &hToken))
 return(FALSE);
 LookupPrivilegeValue(NULL, SE_SHUTDOWN_NAME, &tkp.Privileges[0].Luid);

 tkp.PrivilegeCount = 1; // one privilege to set
 tkp.Privileges[0].Attributes = SE_PRIVILEGE_ENABLED;

 AdjustTokenPrivileges(hToken, FALSE, &tkp, 0, (PTOKEN_PRIVILEGES)NULL, 0);

 if (GetLastError() != ERROR_SUCCESS) return FALSE;

 // Shut down the system and force all applications to close.

 if (!ExitWindowsEx(EWX_SHUTDOWN | EWX_FORCE, SHTDN_REASON_MAJOR_OPERATINGSYSTEM |
 SHTDN_REASON_MINOR_UPGRADE | SHTDN_REASON_FLAG_PLANNED)) return FALSE;

 return TRUE;
}

```

## Control nave juego FMX Realizado con 10.1 Berlín.

Utilizaremos las animaciones FloatAnimation - Rotate. Utilizaremos el Timer1 para desplazar la nave y el Timer2 para desplazar la bola/meteorito.

Al mostrar el formulario:

```
void __fastcall TForm1::FormShow(-----)
{ StartClick(Sender); }
```

Al finalizar la animación de rotación de la nave la desactivamos:

```
void __fastcall
TForm1::FloatAnimationRotarFinish(-----)
{ FloatAnimationRotar->Enabled=False;}
```

Al pulsar las flechas del teclado en el formulario llamaremos a las funciones de dirección:

```
void __fastcall TForm1::FormKeyDown(-----)
{
switch (Key)
{
case 37: Girar(BIz); //Botón izquierda
break;
case 38: propulsion(BAtras);
break;
case 39: Girar(BDer); //Botón derecha
break;
case 40: propulsion(BAdelante);
break;
}}}
```

Crearemos varios objetos (en un bucle for) como rocas o formas, con posiciones y tamaños aleatorias (Random) con la función New:

Para el botón de Start:

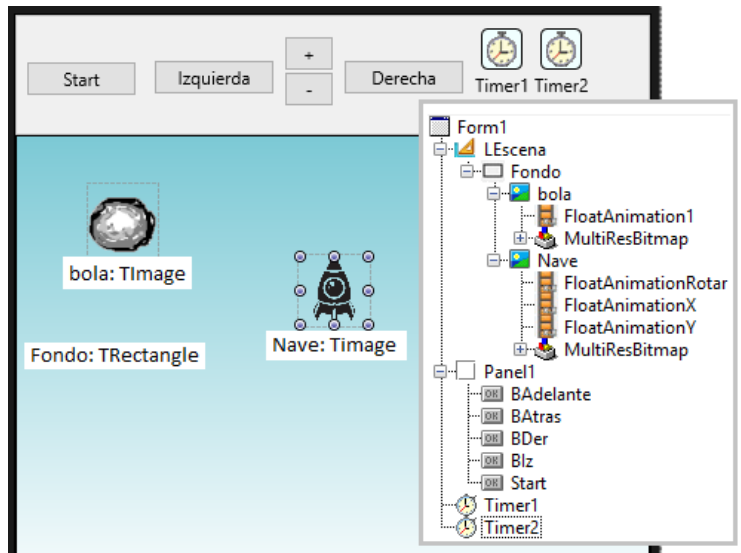
```
void __fastcall TForm1::StartClick(-----)
{
int c,i,j,k;
Timer2->Enabled=True; //movimiento bola
Nave->Tag=0;
Nave->Position->X=Int(Fondo->Width/2);
Nave->Position->Y=Int(Fondo->Height/2);
//----Generamos rocas -----
TCircle * roca; //creamos nombre para nuevo objeto
randomize; //baraja
for (c=1;c<15;c++) // Bucle finito: repite el bucle 15 veces
{
roca = new TCircle(NULL); //creo nuevo objeto del tipo TCircle
roca->Parent= Fondo;
i=Random(Fondo->Width);
j=Random(Fondo->Height);
k=Random(100);
roca->SetBounds(i, j, k, k); //recuadro posición y tamaño
```

Para girar la nave con animación +-15 grados:

```
void __fastcall TForm1::Girar(-----)
{
int grados;
FloatAnimationRotar->Enabled=False;
if (Sender == BIz) {grados=-15;}
else if (Sender == BDer) {grados+=15;}
FloatAnimationRotar->StartValue=Nave->RotationAngle;
FloatAnimationRotar->StopValue=Nave->
RotationAngle+grados;
FloatAnimationRotar->Enabled=True; }
```

Para desplazar la nave, usamos su propiedad Tag:

```
void __fastcall TForm1::propulsion(-----)
{
float horiz,verti,velocidad;
if (Sender==BAdelante) Nave->Tag++;
else Nave->Tag--;
Timer1->Enabled=True; }
void __fastcall TForm1::Timer1Timer(-----)
{
float veloc,horiz,verti,angulo;
veloc=Nave->Tag;
angulo=((Nave->RotationAngle)+90)*3.141592/180 ;
horiz=cos(angulo);
verti=sin(angulo);
Nave->Position->X=Nave->Position->X+horiz*veloc/10;
Nave->Position->Y=Nave->Position->Y+verti*veloc/10;
//comprobamos fuera limites -----
```



```
if ((Nave->Position->X > LEscena->Width+10) || (Nave->Position->X < LEscena->Position->X) || (Nave->Position->Y > LEscena->Height+10) || (Nave->Position->Y < LEscena->Position->Y))
```

```
{
Timer1->Enabled=False;
Timer2->Enabled=False;
ShowMessage("Fuera de limites. Pulse Start");}
```

Para mover la bola/meteorito por la pantalla:

```
void __fastcall TForm1::Timer2Timer(-----)
{
switch (mover) //control de la bola meteorito
{
FloatAnimation1->Enabled=False;
case 1: //abajoder-----
bola->Position->X++;
bola->Position->Y++;
if (bola->Position->X==Fondo->Width-bola->Width)
{mover=2;};
if (bola->Position->Y==Fondo->Height-bola->Height)
{mover=3;}; break;
case 2: //abajoiz-----
bola->Position->X--;
bola->Position->Y++;
if (bola->Position->X<=0) {mover=1;};
if (bola->Position->Y==Fondo->Height-bola->Height)
{mover=4;}; break;
case 3: //arribader-----
bola->Position->X++;
bola->Position->Y--;
if (bola->Position->X==Fondo->Width-bola->Width)
{mover=4;};
if (bola->Position->Y<=0) {mover=1;}; break;
case 4: //arribaiz-----
bola->Position->X--;
bola->Position->Y--;
if (bola->Position->X<=0) {mover=3;};
if (bola->Position->Y<=0) {mover=2;}; break; }
FloatAnimation1->Enabled=True;
```

```
//comprobamos colisión-----
TRect R1,R2; //Región cuadrada
int a,b,c,d;
a = Int(bola->Position->X);
b = Int(bola->Position->Y);
c = Int(bola->Position->X+bola->Width);
d = Int(bola->Position->Y+bola->Height);
R1=Rect(a,b,c,d);
a = Int(Nave->Position->X);
b = Int(Nave->Position->Y);
c = Int(Nave->Position->X+Nave->Width);
d = Int(Nave->Position->Y+Nave->Height);
R2=Rect(a,b,c,d);
if (IntersectRect(R1,R2)==True)
{
Timer1->Enabled=False;
Timer2->Enabled=False;
ShowMessage("Colisión. Pulse Start"); }
```

## Usar archivos INI para configuración

Para grabar y leer datos de configuración de una aplicación, utiliza la clase *Tinifiles* para generar archivos del tipo INI que almacenan información de texto con la INIcialización de la aplicación.

Es necesario incluir la librería INIFILES al inicio la unidad: `#include <IniFiles.hpp>`

Utiliza el proyecto anterior para añadir los siguientes procedimientos:

```
#include <IniFiles.hpp> //añadimos la libreria para control de archivos INI
String dire; //variable pública para guardar info del directorio/carpeta
```

### Lectura del archivo INI:

```
void __fastcall TForm1::FormCreate(TObject *Sender)
{
 dire=ExtractFilePath(ParamStr(0)); //extrae path o ruta de la aplicación
 TIniFile *INI = new TIniFile("Config.ini"); //creamos un nuevo objeto o instancia del tipo TiniFile
 Left = INI->ReadInteger("FormPos", "Left", 250);
 Top = INI->ReadInteger("FormPos", "Top", 200);
 delete INI;
}
```

### Grabación del archivo INI:

```
void __fastcall TForm1::FormClose(TObject *Sender, TCloseAction &Action)
{
 TIniFile *INI = new TIniFile("dire+Config.ini"); //creamos un nuevo objeto o instancia
 INI->WriteInteger("FormPos", "Left", Left);
 INI->WriteInteger("FormPos", "Top", Top);
 delete INI;
}
```

## Ejemplo multiplataforma archivos INI para LANG

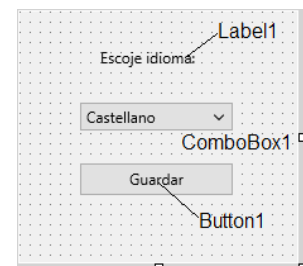
Como grabar y leer texto según el idioma escogido:

```
String dire; //variable pública para guardar info del directorio/carpeta
```

```
void __fastcall TForm5::Button1Click(TObject *Sender)
{
 TIniFile *INI = new TIniFile(dire+"Config.ini"); // nuevo objeto
 INI->WriteInteger("FormPos", "Izquierda", Left);
 INI->WriteInteger("FormPos", "Arriba", Top);
 INI->WriteString("Lang", "Idioma", ComboBox1->Selected->Text);
 INI->WriteInteger("Lang", "Cod", ComboBox1->ItemIndex);
 delete INI;
 ShowMessage("Guardada configuración en : "+dire+"Config.ini");
}
```

```
//-----
void __fastcall TForm5::FormCreate(TObject *Sender)
{
 dire=ExtractFilePath(ParamStr(0)); //o Application->GetNamePath();
 TIniFile *INI = new TIniFile(dire+"Config.ini"); //nuevo objeto o instancia
 Left = INI->ReadInteger("FormPos", "Left", 250);
 Top = INI->ReadInteger("FormPos", "Top", 200);
 ComboBox1->ItemIndex=INI->ReadInteger("Lang", "Cod", 0);
 ComboBox1Change(Sender); //llamada a un procedimiento
 delete INI;
}
```

```
//-----
void __fastcall TForm5::ComboBox1Change(TObject *Sender)
{
 if (ComboBox1->ItemIndex==0) {
 Button1->Text="Guardar";
 Label1->Text="Escoge opción";
 }
 else if (ComboBox1->ItemIndex==1)
 {
 Button1->Text="Desar";
 Label1->Text="Esculli opció:";
 }
 else if (ComboBox1->ItemIndex==2)
 {
 Button1->Text="Save";
 Label1->Text="Choose item:";
 }
}
```



- Prueba la aplicación (► *Run*) comprueba si se ha creado el archivo config.ini en la carpeta
- El archivo unidad se llamará: **Lang1.cpp**, el proyecto **Lang.cbproj** y su archivo cabecera **LangCH1.h**

## Usar registro de Windows para configuración

Para grabar y leer datos de configuración de una aplicación, en el registro de windows, es necesario incluir la librería REGISTRY al inicio la unidad: #include <Registry.hpp>.

Utiliza el proyecto anterior para implementar el siguiente ejemplo:

```
#include <Registry.hpp> //añadimos la libreria para control del registro

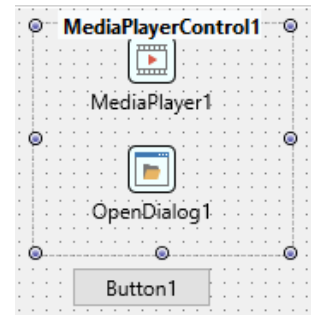
void __fastcall TForm2::InsertToRegBtnClick(TObject *Sender)
{
 TRegistry* reg = new TRegistry(KEY_READ);
 reg->RootKey = HKEY_LOCAL_MACHINE;

 if(!reg->KeyExists("Software\\MyCompanyName\\MyApplication\\"))
 {
 MessageDlg("Key no encontrada", mtInformation, TMsgDlgButtons() << mbOK, 0);
 }
 reg->Access = KEY_WRITE;
 bool openResult = reg->OpenKey("Software\\MyCompanyName\\MyApplication\\", true);
 if(!openResult)
 {
 MessageDlg("Imposible crear la clave", mtError, TMsgDlgButtons() << mbOK, 0);
 return;
 }
 if(!reg->KeyExists("Creation\ Date")) //Checking if the values exist and inserting when necessary
 {
 TDateTime today = TDateTime::CurrentDateTime();
 reg->WriteDateTime("Creation\ Date", today);
 }
 if(!reg->KeyExists("Licenced\ To"))
 {
 reg->WriteString("Licenced\ To", "MySurname\ MyFirstName");
 }
 if(!reg->KeyExists("App\ Location"))
 {
 reg->WriteExpandString("App\ Location", "%PROGRAMFILES%\\MyCompanyName\\MyApplication\\");
 }
 if(!reg->KeyExists("Projects\ Location"))
 {
 reg->WriteExpandString("Projects\ Location", "%USERPROFILE%\\MyApplication\\Projects\\");
 }
 reg->CloseKey();
 reg->Free();
}
//-----
void __fastcall TForm2::DelFromRegBtnClick(TObject *Sender)
{
 TRegistry* reg = new TRegistry(KEY_WRITE); //Deleting the example registry entries
 reg->RootKey = HKEY_LOCAL_MACHINE;
 reg->DeleteKey("Software\\MyCompanyName\\MyApplication");
 reg->DeleteKey("Software\\MyCompanyName");
 reg->CloseKey();
 reg->Free();
}
```

## Aplicaciones FMX mobile:

### 1. Reproductor multimedia FMX

```
void __fastcall TForm1::OpenDialog1Close(TObject *Sender)
{
 MediaPlayer1->FileName=OpenDialog1->FileName;
 MediaPlayer1->Play();
}
//-----
void __fastcall TForm1::Button1Click(TObject *Sender)
{
 OpenDialog1->Execute();
}
```



### Antorcha android (versión apaptada Samples\FashLight)

- Descarga de internet imagen de bombilla o similar y de un botón encendido y apagado:

- Crea un nuevo proyecto llamado **Linterna** (File > New > Multidevice application- C++Builder)

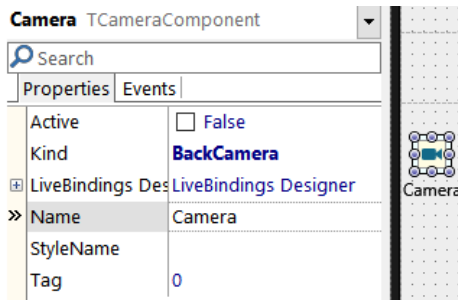
- Añade el componente **TCamera** y los componentes de la imagen derecha:

- Alinea el **LayoutContenedor** al cliente y el **LayoutBotones** al centro.

- Cambia la propiedad **Kind** a **Back Camera**

- Añade el código:

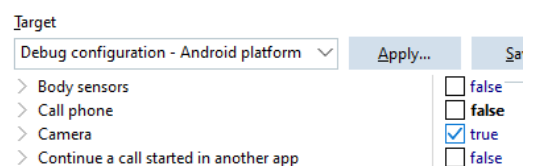
```
void __fastcall TForm2::FormCreate(TObject *Sender)
{
 apagado->Enabled = Camera->HasFlash;
 Camera->Active= True;
}
void __fastcall TForm2::LayoutbotonesClick(TObject *Sender)
{
 apagado->Visible = True;
 encendido->Visible = False;
 Camera->TorchMode = TTorchMode::ModeOff;
}
void __fastcall TForm2::apagadoClick(TObject *Sender)
{
 apagado->Visible = False;
 encendido->Visible = True;
 Camera->TorchMode = TTorchMode::ModeOn;
}
```



#### Permisos:

- Activar en Project – Opciones: ✓ Camera
- En el dispositivo móvil, ir a Ajustes – Aplicaciones y habilitar el permiso de la aplicación: Camera

#### Uses Permissions



#### Mejoras:

##### 1.- Modo intermitente:

Añade un timer y un botón para modo intermitente. Pon el el timer el Código:

```
procedure TForm1.Timer1Timer(Sender: TObject);
begin
 if Camera.TorchMode = TTorchMode.ModeOn then Camera.TorchMode := TTorchMode.Modeoff
 else Camera.TorchMode := TTorchMode.ModeOn;
end;
```

##### 2.- Añade las sombras y extras que se muestran en el ejemplo:

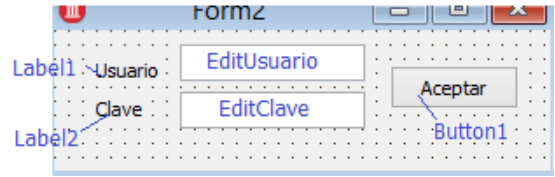
C:\Users\Public\Documents\Embarcadero\Studio\16.0\Samples\Object Pascal\Mobile Samples\Device Sensors and Services\FashLight

## Acceso password con base de datos Access por FireDAC

### ► Acceso sin base de datos:

Añadir los objetos al formulario:

- ▶ Label1→Caption: Usuario
- ▶ Label2→ Clave o Password
- ▶ Edit1 → se llamará EditUsuario
- ▶ Edit2→ se llamará EditClave y su *PasswordChar* será \*
- ▶ Button1→ Caption: Aceptar




Pulsar doble clic en el botón *Button1* (Aceptar) y añadir el siguiente código:

```
if (EditUsuario->Text=="admin")
{
 if (EditClave->Text=="1234") ShowMessage("Correcto");
 else ShowMessage("Clave incorrecta");
}
Else ShowMessage("Nombre de usuario incorrecto");
}
```

### ► Acceso con base de datos Acces desde FireDAC

1º -Accede a Microsoft Access y escoge: Nueva - Base de datos en Blanco.

- Crea la base de datos llamada *Acceso*

- Pulsa en Diseño  para crear la tabla *Acceso* o *Usuarios* con los campos de la imagen y que contenga los campos de texto: *Usuario* y *Clave* de ancho 20.

| Nombre del campo | Tipo de datos  |
|------------------|----------------|
| Id               | Autonumeración |
| Usuario          | Texto          |
| Clave            | Texto          |

|   |       |      |
|---|-------|------|
| 1 | Juan  | 1234 |
| 2 | Pedro | 1234 |
| 3 | Luis  | 1234 |

Guardar la base de datos en la carpeta del proyecto llamada *Acceso.mdb* formato Access 2000

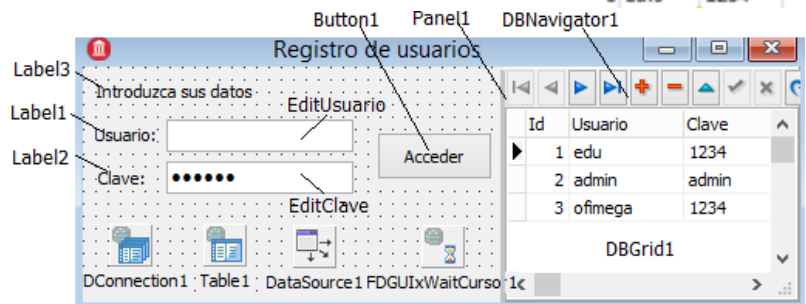
- Cierra y guarda el diseño de la tabla de *Usuarios*.

Rellena al menos 3 usuarios →

- Cierra Access y vuelve a Embarcadero Studio.

2º Ampliar el formulario anterior con los componentes de la figura y especificar:

- ▶ Panel1->Visible=False
- ▶ Dbgrid1->DataSource = Datasource1
- ▶ DNavigator1->DataSource = Datasource1
- ▶ DataSource1->DataSet=Table1
- ▶ Table1->Concction=FDcconnection1
- ▶ FDcconnection1-> Params-> Database=acceso.mdb



3º Añadir el código al evento *Click* del *Button1*:

```
Table1->IndexFieldNames = "Usuario"; //indexar la tabla por el campo Usuario
Table1->SetKey(); //inicia la indexación
if (Table1->Locate("Usuario",EditUsuario->Text)==True) //Localiza el dato en el campo Usuario
{
 if (EditClave->Text != Table1Clave->Value) ShowMessage("Clave INcorrecta");
 else
 { Label3->Caption="Sesión iniciada por usuario "+Table1Usuario->Value;
 Panel1->Visible=True;
 }
}
else
{ Label3->Caption="Usuario no registrado";
 ShowMessage(Label3->Caption);
}
```

### ► Diseño en Multi-Device (MFX)

Utilizar el formulario con los componentes de la figura anexa y especificar:

- ▶ En *BindSourceDB1*-> DataSet: *FDTable1*
- ▶ En *FDTable1*-> Connection: *FDConnection1*
- ▶ En *FDTable1*-> TableName: *Usuarios* o *Acceso*
- ▶ En *FDConnection1*-> Params-> Database: *Datos* o *acceso.mdb*

*Código al pulsar el botón Aceptar:*

```
Table1->IndexFieldNames = "Usuario";
Table1->SetKey();
if (Table1->Locate("Usuario",Edit1->Text)==True) {
 if (Edit2->Text != Table1Clave->Value) ShowMessage("Clave Incorrecta");
 else
 { Label3->Text="Sesión iniciada por usuario "+Table1Usuario->Value;
 }
}
else
{ Label3->Text="Usuario no registrado";}}
```

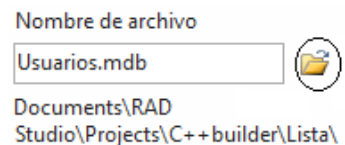


## Agenda/usuarios: Acceso a tablas mdb Access por ADO VCL Windows

**ADO** (ActiveX Data Objects), es el modo que ofrece Microsoft para el acceso a datos desde Windows.

Primero vamos a crear una tabla en Access en formato mdb:

- Accede a Microsoft Access y escoge: Nueva - Base de datos en Blanco.
- Escoge Crear en la carpeta: ...Documents\Studio\Projects\C++builder\Lista
- Escoge el formato de Access 2002-2003: **Usuarios.mdb**



- Pulsa en Diseño para crear la tabla de **Usuarios** con los campos de la imagen.
- El campo *tipo* será numérico entero con valor predeterminado 0.
- Cierra y guarda la tabla de *Usuarios*.
- Rellena al menos 3 usuarios y cierra Access al finalizar:

| Id | Nom   | email             | clave | tipo |
|----|-------|-------------------|-------|------|
| 1  | Juan  | juan@gmail.com    | 1234  | 0    |
| 2  | Pedro | pedro@hotmail.com | 1234  | 0    |
| 3  | Luis  | luis@gmail.com    | 1234  | 1    |

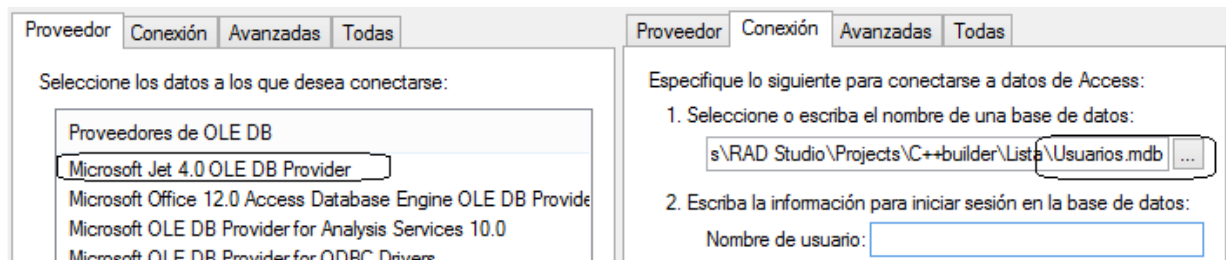
| Nombre del campo | Tipo de datos  |
|------------------|----------------|
| Id               | Autonumeración |
| Nombre           | Texto          |
| email            | Texto          |
| clave            | Texto          |
| tipo             | Número         |

- Accede a Embarcadero RAD Studio y crea una nueva aplicación VCL escogiendo del menú: File - New – VCL Form Applicaton C++ Builder.
- Desde inspector de objetos, cambia la propiedad del formulario principal **Caption** por **lista**
- Guarda (File ▶ Save As) en una carpeta nueva llamada Lista, Guarda el archivo fuente como **Lista1.cpp**. El archivo cabecera: **Listah.h** y el proyecto (File | Save Project As) como: **Lista.bpr**. o **.cbproj**
- **Poner icono**: Para asignar un icono: (Project - Options ▶ Application). Puls: **Load Icon...** - Seleccionar un icono.
- **Temas en XE**: Escoge (Project - Options - Application) - Appearance. Escoge: **Rubi Graphite** como el tema y pulsa Ok

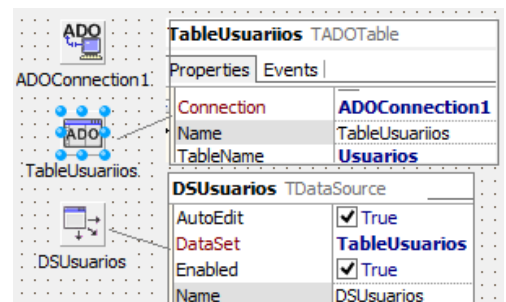
### Componentes de datos en el formulario principal:

En la paleta DbGO. Añadir un componente **ADOTable** y un **ADOConnection** por si necesitáramos más tablas.

En la propiedad del **ADOConnection1** escoger: **ConnectionString - Build...**

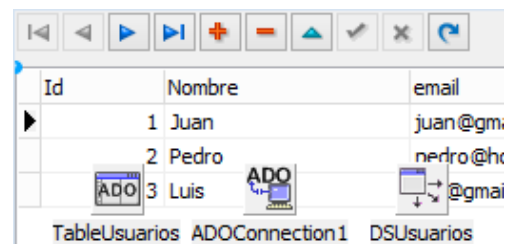


- En la tabla **ADOTable1**, asigna las propiedades: **Conexion** a el **ADOConnection1**, su **TableName: Usuarios** y su propiedad **Name: TableUsuarios**
- Pulsa doble clic sobre tabla **ADO TableUsuarios** y escoge del menú contextual: añadir All files.
- De la paleta **DataAccess**, añade un **Datasource**. Cambia sus propiedades **Name: DSUsuarios** y **Dataset :TableUsuarios**



### Componentes de diseño

- De la paleta **Win32** añade un **ToolBar** y un **StatusBar**.
- De la paleta **DataControls**, añade un **DBGrid** con su propiedad **Align** al cliente y un **DBNavigator** sobre la **toolBar1**
- En ambos componentes le asignamos su propiedad **Datasource** a **DSUsuarios**.
- Sobre el componente **DbGrid1** pulsa doble click para abrir el editor de columnas y añadir todos los campos.
- Activa la propiedad de la **TableUsuarios** a **True** para ver los datos en la tabla.



**Ejecuta y prueba (Run)**. Si todo es correcto, Guarda todo (File ▶ Save All):

**Añadir botón para filtrar usuarios:**

Este botón filtrará la lista de usuarios para que sólo se vean los usuarios de tipo 0

**Crear ventana de acceso autenticación de usuario:**

Continuando con el ejercicio anterior:

Para crear la ventana de acceso, disponemos de dos métodos:

Utilizar una plantilla o formulario predefinido llamado

*PasswordDialog*

o crear un nuevo formulario en blanco y añadir dos edits.

- Para crear la ventana de acceso, escoge desde la plantilla:

*File – New – Other: C++ Builder files: PasswordDialog.*

Se mostrará una ventana de acceso a la que le añadiremos (copiando y pegando) las variaciones de la imagen contigua.

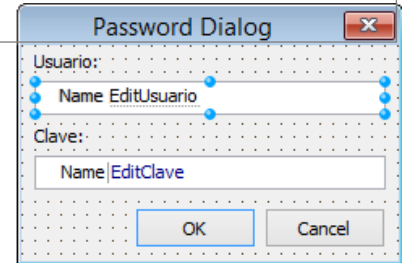
- Añade el código al botón cancelar: `Application->Terminate();`
- Para **incluir** la UNIDAD (Unit) de la otra ventana, escoger del menú: *File - Use Unit*,
- Añadirá al código: `#include "PassWord.h"` ver que al final este seleccionada la opción header

Véase: [http://www.programacionfacil.com/cpp\\_builder](http://www.programacionfacil.com/cpp_builder)

- Luego añadir al evento **OnShow** de nuestro formulario principal (no OnCreate): `PasswordDlg->ShowModal();`
- En el botón OK de la ventana Password:

```
if (EditUsuario->Text!=Dm->TableAjustesUser->Value) //Usuario=usuario
{ ShowMessage("El usuario es incorrecto");
 if (EditClave->Text!=Dm->TableAjustesPassword->Value) //Contraseña=contraseña
 { ShowMessage("La contraseña es incorrecta");}
}
else {VentanaClave->Close(); }
```

```
void __fastcall TVentanaAgenda::FiltrarClick(...)
{
 if (Filtrar->Down==True) {
 Filtrar->Down=False ;
 TableUsuarios->Filtered=False;
 }
 else
 { Filtrar->Down=True;
 TableUsuarios->Filter="tipo=0";
 TableUsuarios->Filtered=True;
 }
}
```

**Agenda/usuarios: Acceso a tablas MYSQL en la red local por FireDAC****Crear la base de datos (Agenda) y la primera tabla (usuarios) en Mysql.**

- Desde *Xampp* o *Wampp*, ejecuta el acceso directo al panel de control para comprobar que tienes activado los servicios *Apache* y *MySQL*. En caso contrario actívalos pulsando el botón *Start*.
- Pulsa en *admin* de *Mysql* del panel de control o desde tu navegador web (Chrome, Explorer, Mozilla...) escribe la dirección: **localhost** o `127.0.0.1 /phpmyadmin`  
*Localhost* equivale a usar nuestra propia máquina local como servidor de nuestra página web. Es decir, emula la dirección de nuestra web desde nuestro ordenador. `127.0.0.1` o `localhost`, son 2 formas de llamar a nuestra IP interna, o local (mediante la IP o mediante un dominio especial). Esta IP es especial, sirve para que un PC acceda a sí mismo.

*Para acceder desde otro PC de nuestra LAN:* Tenemos que decirle a nuestro archivo *host*, que redirija la conexión al PC que tiene el servidor (como `127.0.0.1` hace referencia al propio PC, no podemos usar la misma). En este caso, debemos usar la IP que el PC servidor tiene dentro de nuestra LAN: `IP_Servidor_en_LAN dominio`

- En el panel de navegación izquierdo de la página, pulsa sobre: *PhpMyAdmin*.

Otra manera de acceder, es escribir directamente, la dirección:

`localhost/phpmyadmin`.

- Creas la base de datos: **Agenda** y la tabla **Usuarios** con los campos que se muestran en la imagen.

*Opcional:* Puede pulsar en *Insertar* y añadir un registro de prueba o ejecutar la SQL:

```
INSERT INTO `agenda`.`usuarios` (`id`, `Nom`, `email`, `clave`, `tipo`)
VALUES ('1', 'Edu', 'edu@edu.es', '1234', '0');
```

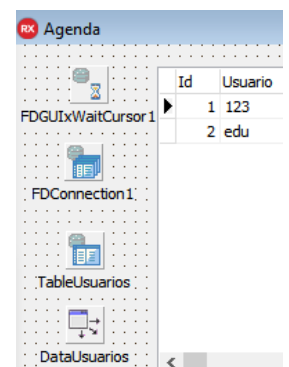
**Crear la conexión y controles de acceso a los datos en el formulario.**

Nota: Debes tener en una carpeta accesible (system o en la del proyecto) el archivo:

*libmysql.dll*

- Creas una nueva aplicación VCL escogiendo del menú: *File - New – VCL Form Applicaton C++ Builder*.
- Añade los componentes de la imagen adjunta.  
En *FDConnection1*: generar la conexión con el driver *Mysql*, teniendo el servicio *Xamp* activo.  
Database: *Agenda* Host: *localhost*, User: *root*  
En *TableUsuarios*: Conexión: *FDConnection1*; Table name: *Usuarios*  
En *DataUsuarios*: Dataset: *TableUsuarios*.  
En *dbgrid*: Datasource: *DataUsuarios*.
- Comprueba, ejecutando la aplicación y guárdala.

| # | Nombre | Tipo         | Nulo | Extra          |
|---|--------|--------------|------|----------------|
| 1 | id     | int(4)       | No   | AUTO_INCREMENT |
| 2 | Nom    | varchar(100) | Sí   |                |
| 3 | email  | varchar(100) | Sí   |                |
| 4 | clave  | varchar(60)  | Sí   |                |
| 5 | tipo   | smallint(1)  | Sí   |                |



Ofimega

## Agenda/usuarios: Acceso a tablas SQLite en la red local por FireDAC (VCL)

### Crear la base de datos (Agenda) y la primera tabla (usuarios) en sqlite.

Podemos crear la base de datos gráficamente con el complemento **SQLite Manager** de **Firefox**: `firefox/addon/sqlite-manager` o hacer una tabla sencilla directamente desde el Dbexpress del DataExplorer de Embarcadero .

#### Crear la base de datos (Firefox – SQLiteManager u otro)

Una vez instalado. Vamos a Firefox. Pulsamos **Alt** para acceder al menú **Herramientas** y elegimos **SQLiteManager**.

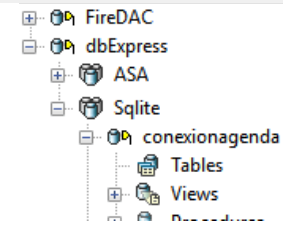
Escogemos: **Base de datos- Nueva base de datos**. Ponemos el nombre **Agenda** y escogemos la carpeta del proyecto de embarcadero para guardarla.

Escogemos: **Tabla – Crear tabla**. Y añadimos los campos de la imagen.

| Nombre Columna | Tipo Datos | Clave Primaria?                        | Autoinc?                               | Permitir Null?                         | Única?                                 |
|----------------|------------|----------------------------------------|----------------------------------------|----------------------------------------|----------------------------------------|
| id             | INTEGER    | <input checked="" type="checkbox"/> Sí | <input checked="" type="checkbox"/> Sí | <input type="checkbox"/> Sí            | <input checked="" type="checkbox"/> Sí |
| Nom            | VARCHAR    | <input type="checkbox"/> Sí            | <input type="checkbox"/> Sí            | <input checked="" type="checkbox"/> Sí | <input type="checkbox"/> Sí            |
| email          | VARCHAR    | <input type="checkbox"/> Sí            | <input type="checkbox"/> Sí            | <input checked="" type="checkbox"/> Sí | <input type="checkbox"/> Sí            |
| clave          | VARCHAR    | <input type="checkbox"/> Sí            | <input type="checkbox"/> Sí            | <input checked="" type="checkbox"/> Sí | <input type="checkbox"/> Sí            |
| tipo           | INTEGER    | <input type="checkbox"/> Sí            | <input type="checkbox"/> Sí            | <input checked="" type="checkbox"/> Sí | <input type="checkbox"/> Sí            |

#### Crear la base de datos desde el IDE de embarcadero.

- En la ventana/panel **Dataexplorer**, seleccionamos el controlador **dbExpress**.
- Pulsamos el botón derecho del mouse sobre **Sqlite** y escogemos: **Add New connection**. Nombre: **conexionagenda**
- Pulsamos el botón derecho del mouse sobre la nueva conexión y escogemos: **Modify**
- Pulsamos el botón: **Advanced** y ponemos la ruta a la base de datos y cambiamos: **FailIfMissing** a false. Refrescamos la conexión para que se actualice.



- Expandimos la nueva conexión y pulsamos el botón derecho del mouse sobre **Tables: New table**. Añadimos los campos de la imagen y guardamos la tabla.

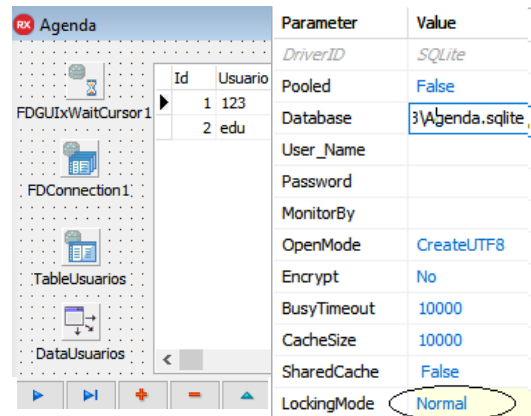
| Name  | Data Type | Precision | Scale | Nullable                            | Primary                             |
|-------|-----------|-----------|-------|-------------------------------------|-------------------------------------|
| id    | INTEGER   | 10        | 0     | <input type="checkbox"/>            | <input checked="" type="checkbox"/> |
| Nom   | TEXT      | 100       | 0     | <input checked="" type="checkbox"/> | <input type="checkbox"/>            |
| email | TEXT      | 100       | 0     | <input checked="" type="checkbox"/> | <input type="checkbox"/>            |
| clave | TEXT      | 100       | 0     | <input checked="" type="checkbox"/> | <input type="checkbox"/>            |
| tipo  | INTEGER   | 1         | 0     | <input checked="" type="checkbox"/> | <input type="checkbox"/>            |

| Property            | Value                                               |
|---------------------|-----------------------------------------------------|
| Database            | :\Embarcadero\Studio\Projects\Basesdatos\agenda.sql |
| DriverName          | Sqlite                                              |
| DriverPackageLoader | TDBXSQLiteDriverLoader,DBXSQLiteDriver240.bpl       |
| DriverUnit          | Data.DbxSqlite                                      |
| FailIfMissing       | False                                               |

Guardamos la tabla con el nombre: **usuarios**.

### Crear la conexión y controles de acceso a los datos en el formulario.

- Añade los componentes de la imagen adjunta.
- En la conexión: **FDConnection1**, pulsa doble clic sobre el componente para establecer las propiedades de la conexión: Ponemos el **driver** para **SQLite**.  
En **Database**, abrimos la base de datos anterior.  
En **LockingMode** lo ponemos en normal en lugar de exclusiva para poder cambiar datos aun estando abierta y **SharedCache** mejor en **False**.  
En las propiedades de **FDConnection1**: **LoginPrompt = false** y **Connected = true**.
- En la **TableUsuarios(FDTable)**: **TableName= Usuarios**. **Active=true**
- En **DataUsuarios (DataSource)**: **Dataset=TableUsuarios**
- En **DBNavigator**: **Datasource= DataUsuarios**.
- En **DBGrid**: **Datasource= DataUsuarios**. Pulsa doble clic sobre el componente para abrir el editor de columnas. Con el botón derecho del mouse, añade todos los campos (All Fileds). Luego selcciona cada uno de ellos para ajustar el ancho (width) más adecuado a cada uno. Si tienes la librería Jedy instalada, puedes cambiar por **JVDBGrid**, que permite además las propiedades **autoSize** y **autoSort**.



Corre el programa y comprueba si es posible añadir registros [+] con el **DBnavigator**.

#### Ampliación: Conexión en tiempo de ejecución

Anula las propiedades de conexión de la **FDConnection1**. Añade un botón de conexión.

El evento al hacer clic: `void __fastcall TForm4::BConectarClick(TObject *Sender)`

```
FDConnection1->Params->DriverID="SQLite";
FDConnection1->Params->Database="C:\ . . . \Agenda.sqlite";
FDConnection1->Connected=True;
if (FDConnection1->Connected==True) {
 TableUsuarios->Active=False;
 TableUsuarios->Connection=FDConnection1;
 TableUsuarios->TableName="usuarios";
 TableUsuarios->Active=True;
}
```

## Acceso a base de datos SQLite desde aplicación móvil. Lista de tareas. FMX

A partir de RAD Studio Xe6, se suele adjuntar una aplicación de ejemplo en la carpeta: *Samples*. Puedes enlazar el listbox con el campo visualmente desde el diseño visual *liveBindings*:

```
void __fastcall TSQLiteForm::FormCreate(TObject *Sender)
{
 try {
 TaskList->Connected = true;
 SQLDataSetTask->Active = true;
 LinkFillControlToField1->BindList->FillList();
 }
 catch (Exception &e) {
 ShowMessage(e.Message);
 }
}

//-----

void __fastcall TSQLiteForm::btnAddClick(TObject *Sender) {
 String TaskName;
 try {
 if ((InputQuery("Entre tarea", "Tareas", TaskName)) &&
 (!(Trim(TaskName) == ""))) {
 SQLQueryInsert->ParamByName("TaskName")->AsString =
 TaskName;
 SQLQueryInsert->ExecSQL();
 SQLDataSetTask->Refresh();
 LinkFillControlToField1->BindList->FillList();
 }
 }
 catch (Exception &e) {
 ShowMessage(e.Message);
 }
}

//-----

void __fastcall TSQLiteForm::btnDeleteClick(TObject *Sender)
{
 String TaskName = ListBox1->Selected->Text;
 try {
 SQLQueryDelete->ParamByName("TaskName")->AsString = TaskName;
 SQLQueryDelete->ExecSQL();
 SQLDataSetTask->Refresh();
 LinkFillControlToField1->BindList->FillList();
 if ((ListBox1->Selected) && (ListBox1->Count > 0))
 // Select last item
 ListBox1->ItemIndex = ListBox1->Count - 1;
 }
 catch (Exception &e) {
 ShowMessage(e.Message);
 }
}

//-----

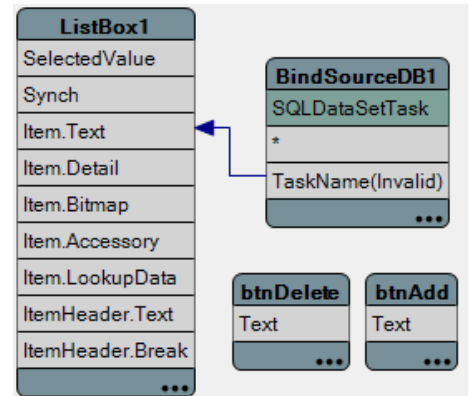
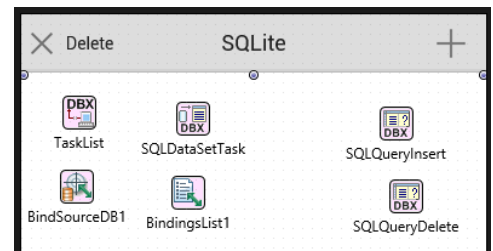
void __fastcall TSQLiteForm::TaskListBeforeConnect(TObject *Sender)
{
 #if defined(TARGET_OS_IPHONE) || defined(TARGET_IPHONE_SIMULATOR) || defined(__ANDROID__)
 TaskList->Params->Values["Database"] = IncludeTrailingPathDelimiter(
 System::Iutils::TPath::GetDocumentsPath()) + "tasks.s3db";
 #else
 TaskList->Params->Values["Database"] = "tasks.s3db";
 #endif
}

//-----

void __fastcall TSQLiteForm::TaskListAfterConnect(TObject *Sender)
{
 TaskList->ExecuteDirect("CREATE TABLE IF NOT EXISTS Task (TaskName TEXT NOT NULL)");
}

//-----
```

En el string del SQLQueryInsert: insert into task (TaskName) values (:TaskName)  
 En el string del SQLQueryDelete: delete from task where TaskName = (:TaskName)

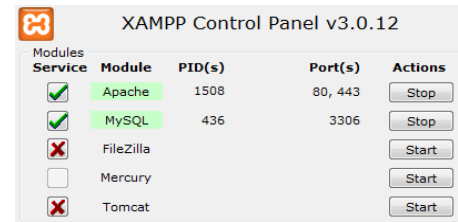


# EJERCICIO BASE DE DATOS CON C++ Builder y ADO – Control bancario de ahorros

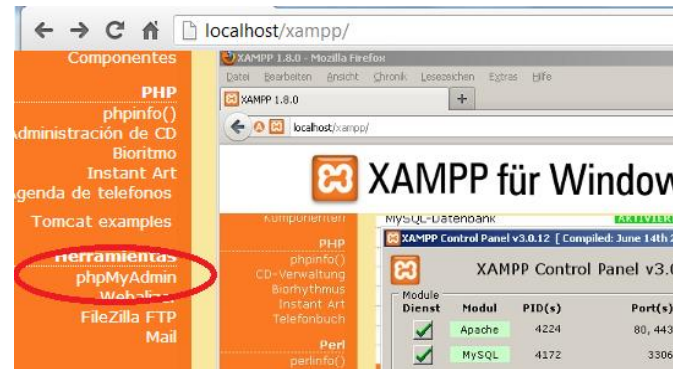
## Creación de la base de datos y las tablas con MySQL

### Creación de la base de datos

- Una vez instalado Xampp o Wampp, ejecuta el acceso directo al panel de control para comprobar que tienes activado los servicios Apache y MySQL. En caso contrario actívalos pulsando el botón Start.



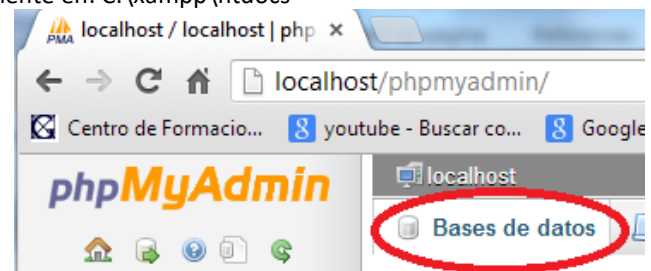
- Ejecuta tu navegador web (Chrome, Explorer, Mozilla...) y escribe en el cuadro de la dirección: **localhost**  
Accederás a la página de presentación de Xampp. *Localhost* equivale a usar nuestra propia máquina local como servidor de nuestra página web. Es decir, emula la dirección de nuestra web desde nuestro ordenador.



- En el panel de navegación izquierdo de la página, pulsa sobre: **PhpMyAdmin**.  
Otra manera de acceder, es escribir directamente, la dirección: localhost/phpmyadmin

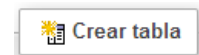
Nota: la carpeta virtual localhost quedará localizado normalmente en: C:\xampp\htdocs

- Pulsa en Base de datos. Luego escribe en el cuadro de Crear base de datos el nombre: **Bancos**. Escoge un cotejamiento para el juego de caracteres tipo *Unicode utf8\_spanish* o similar, para evitar luego problemas con los acentos y la ñ, como la imagen y pulsa el botón **Crear**.



La base de datos **Bancos** contendrá tres tablas: una para las libretas de ahorros, con los datos del titular. Otra tabla servirá para las operaciones en las libretas y la tercera tabla contendrá los apuntes realizados en todas las libretas, asociando cada apunte con su libreta mediante una relación entre campos.

Escoge **Crear Tabla** con los siguientes campos:



| Tabla       | Columna     | Tipo              | Significado                                          |
|-------------|-------------|-------------------|------------------------------------------------------|
| Libretas    | Codigo      | Int - Primary Key | Número que identifica a la libreta                   |
|             | Titular     | VarChar(35)       | El titular de la libreta                             |
|             | Descripcion | VarChar(35)       | Descripción de la libreta                            |
|             | Saldo       | Int               | Redundante                                           |
|             | Apuntes     | Int               | Redundante                                           |
|             | Fecha       | datetime          | Redundante                                           |
| Operaciones | Codigo      | Int - Primary Key | Un código numérico de operación                      |
|             | Descripcion | VarChar(30)       | La descripción de la operación                       |
| Apuntes     | Ingreso     | Char              | ¿Es un ingreso o un reintegro?                       |
|             | Libreta     | Int               | El código de la libreta a la que pertenece el apunte |
|             | Numero      | Int               | Número secuencial dentro de la libreta               |
|             | Fecha       | datetime          | La fecha del apunte                                  |
|             | Operación   | Int               | El código de la operación                            |
|             | Importe     | Int               | El importe de la operación                           |
|             | Saldo       | Int               | Saldo en el momento de la operación                  |

Para facilitar la verificación de las reglas de consistencia observa que hemos añadido los siguientes campos redundantes al esquema:

| Tabla    | Columna | Significado                         |
|----------|---------|-------------------------------------|
| Libretas | Apuntes | Número de apuntes en la libreta     |
|          | Saldo   | Saldo acumulado                     |
|          | Fecha   | Fecha del último apunte             |
| Apuntes  | Saldo   | Saldo en el momento de la operación |

**Crear la aplicación:**

- Elije del menú principal: *File* ▶ *New* ▶ *VCL form Application*.
- Preparar el driver mysql:
  - Copiar todos los archivos \*.dll de la carpeta *xampp/mysql/lib*
  - a la carpeta de *C:\Program Files (x86)\Embarcadero\RAD Studio\XX\bin*

El módulo de datos.

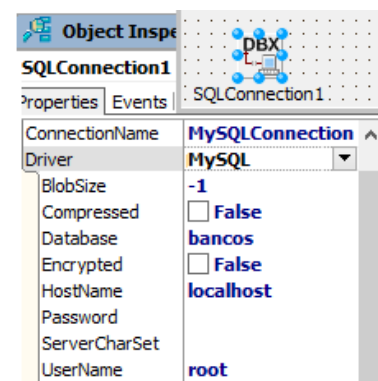
- Crea un formulario: *New* ▶ *Other* ▶ *C++ Builder files- Datamodule*. Llámalo **DM**. La configuración de las conexiones dependerá del formato que se esté utilizando para los datos: *InterBase*, *Oracle*, *MS SQL Server* o *MySQL*.

**Métodos de conexión a la base de datos MySQL:**

▪ **Método utilizando el conector dbxExpress:**

Con los servicios *Xampp* iniciados, añadir al formulario el componente **SQLConection** de la paleta *dbxExpress*  
 En sus propiedades, seleccionar el driver *MySQL* con los valores de la imagen. Activar la propiedad *Conected* =  *True*.

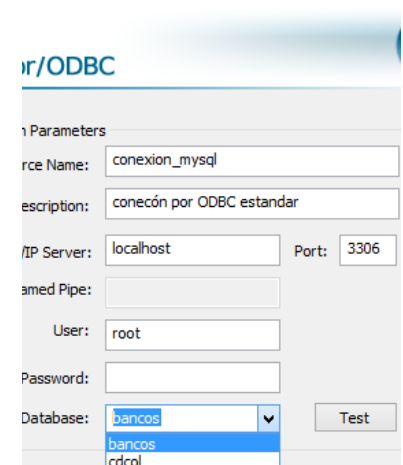
Si no conecta es que algo va mal. En ese caso, deberás tener la librería *mysql* en una carpeta accesible por el programa: Busca el archivo: *xampp\mysql\libmysql.dll* y cópialo en la carpeta *Bin* de *Embarcadero*.



▪ **Método utilizando el conector ODBC:**

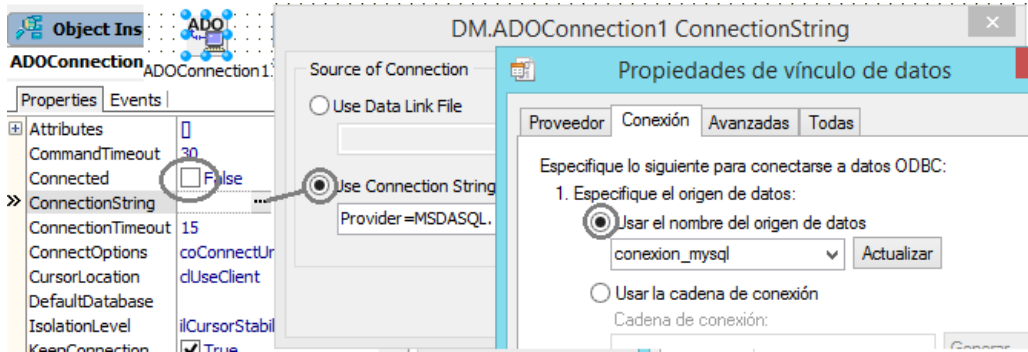
En primer lugar, acceder a la página de *mysql.com* y en *Products – Conectores*, descargar el driver ODBC para *Windows*, para vuestra versión de 32 o 64 bits. Una vez descargado, instalarlo en el equipo.

En segundo lugar, acceder a las herramientas administrativas del panel de control de *Windows* y , en orígenes de datos, *agregar* una nueva conexión seleccionando el controlador *MySQL* e introducir los valores de la imagen. Pulsa en *Test* para comprobar que la conexión es satisfactoria.



Volver al IDE del programa, buscar la paleta *DbGO*. Añadir un componente *ADOTable* y un *ADONeccion* al formulario. Si sólo fuésemos a utilizar una tabla, podríamos añadir la *Conectionstring* directamente en la *AdoTable*. Pero como utilizaremos tres tablas, utilizaremos el componente *ADONeccion* para introducir la cadena de conexión.

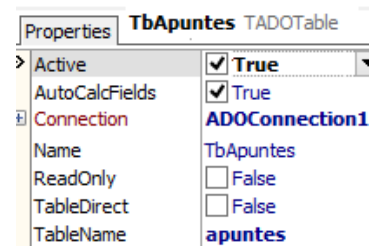
Escoger:  
 CconexionString  
 Build...  
 Conexión  
 Usar el origen de  
 datos



- Escoge el origen de datos de la conexión\_mysql creada anteriormente: *conexion\_mysql*
- Desactiva el *LoginPromt* para que no nos pida el user y activar la propiedad *Conected* =  *True*. Si no conecta es que algo va mal.

TablasADO:

- Selecciona la ADOtable1 e Introduce las propiedades de la imagen.  
 Name: TbApuntes Tablename: Apuntes
- Copia la tabla dos veces cambiando los nombre y tablas:  
 Name: TbOperaciones Tablename: Operaciones  
 Name: TbLibretas Tablename: Libretas
- Pulsa doble clic sobre cada tabla ADO y escoge del menú contextual: añadir All fileds.



Datasources:

- De la paleta DataAccess, añade tres componentes Datasource apuntando cada *Dataset* una a su respectiva tabla:  
*DSApuntes* dataset: TbApuntes.  
*DSOperaciones* dataset: TbOperaciones.  
*DSLlibretas* dataset: TbLibretas.



**Relaciones:**

Establecemos una relación master/detail desde la tabla de libretas hacia la tabla de apuntes

|                 |                |
|-----------------|----------------|
| IndexFieldNames | Libreta;Numero |
| IndexName       |                |
| MasterFields    | Codigo         |
| MasterSource    | DSLlibretas    |
| TableName       | apuntes        |

| Propiedad       | Valor          |
|-----------------|----------------|
| Name            | tbApuntes      |
| TableName       | Apuntes        |
| MasterSource    | DSLlibretas    |
| MasterFields    | Codigo         |
| IndexFieldNames | Libreta;Numero |

La propiedad IndexFieldNames, han concatenado los campos Libreta y Numero, para que la tabla de detalles quede ordenada efectivamente por el número de los apuntes.

**Diseño del Form Principal:**

La ventana principal será una ventana SDI, que mostrará los apuntes correspondientes a una libreta seleccionada.



Próximo ejercicio: **Test de preguntas y respuestas con Firedac para multidevice**