TIPOS DE DATOS

A toda variable que se use en un programa, se le debe asociar (generalmente al principio del programa) un tipo de dato específico.
Un tipo de dato define todo el posible rango de valores que una variable puede tomar al momento de ejecución del programa y a lo largo de toda la vida útil del propio programa.

Los tipos de datos más comunes en C++Builder son:

Enteros

short int	16 bits	-32,768 a 32,767
unsigned int	32 bits	0 a 4,294,967,295
int	32 bits	-2,147,483,648 a 2,147,483,647
long	32 bits	-2,147,483,648 a 2,147,483,647

Reales

float	32 bits	3.4 x 10-38 a 3.4 x 10+38
double	64 bits	1.7 x 10-308 a 1.7*10+308

Caracter

char	8 bits	0 a 255

AnsiString (String): Cadena de caracteres
AnsiString no es propiamente un tipo de dato, sino una clase que se especializa en El almacenamiento y manipulación de datos de tipo String, es decir, como clase ya contiene toda una serie de métodos (procedimientos y funciones) que pueden usar directamente las variables(objetos) que se declaren de tipo AnsiString,

Lógico (bool)

Tiene 2 valores trae y false.

IDENTIFICADORES

Es un nombre que se le da a algo en C++ (Constantes, variables, tipos de datos, funciones)

Reglas de un Identificador

Un identificador es una secuencia de letras, dígitos y carácter de subrayado. Las reglas a seguir por un identificador son:

1) el primer carácter de un nombre debe comenzar con una letra o un carácter de subrayado (_)
2) Los caracteres siguientes pueden ser letras, dígitos o carácter de subrayado.
3) Los identificadores en C++ son sensibles a las mayúsculas. Ejemplo : Los nombres area, Area, AREA son 3 identificadores distintos
4) No se pueden utilizar palabras reservadas del lenguaje como identificadores

COMENTARIOS EN UN PROGRAMA

Un comentario es ignorado por el compilador, esto no se convierte en instrucciones ejecutables.

C++ soporta dos tipos de comentarios. El primero utiliza las parejas de caracteres /* y */ para definir el rango del comentario. Este rango se puede limitar a la misma linea o extenderse a varias lineas.
El segundo tipo de comentario de C++ utiliza la pareja de caracteres que // que marca el comienzo de un comentario que se extiende estrictamente hasta el final de la linea.
Ejemplos:

```
/* El autor de Este Resumen
   nació en la Ciudad de Trujillo, estudio en la UNT */

int x; // Se declara x como una variable entera.
```

VARIABLES
Una variable es una posición de memoria donde se guarda un valor el cual puede ir cambiando durante la ejecución del programa:

Declaración de Variables.

Tipo_de_dato identificador;

Ejemplo :

```
int a,b,c; // Se declaran 3 variables enteras
float area,lado1; // se declaran 2 variables reales
char letra; // se declara 1 variable de carácter
String nombre; // se declara una cadena de caracteres
```

En C++ al declarar una variable se la puede inicializar Por ejemplo

```
int c=3; // Se declara la variable c y se inicializa con el valor 3
float r=9.81; // Se declara una variable real con valor inicial 9.81
String apellido=”Gonzales”;
```

CONSTANTES
Una constante es una posición de memoria donde se guarda un valor el cual nunca cambia durante la ejecución del programa

Declaración de constantes
Hay dos maneras de declarar constantes

a) Usando la directiva #define al comienzo de un programa.(Usado en C)
#define identificador valor

Ejemplo:

```
#define G 9.81
#define carácter ‘$’
#define mensaje “HOLA MUNDO”
```

b) Usando la sentencia const (Usado en C++)

const Tipo_de_dato identificador;

Ejemplos

```
const float G=9.81;
const char caracter='$';
const char *mensaje = "HOLA MUNDO";
```

OPERADOR DE ASIGNACIÓN

La asignación directa de valores a una variable se realiza con el signo = de la sgte manera:

Nombre_de_variable = valor que se le asigna;

En donde valor que se le asigna puede ser una constante, una variable o una expresión.
Ejemplos :

```
X=12.5;
A=12*5+10;
T=T+1;
```

Tener en cuenta que la asignación es de izquierda a derecha es decir el termino de la derecha se asigna al de la izquierda y no a la inversa.

Por ejemplo:

```
A=30;
B=75;
```

Si colocamos A=B; estamos Asignando el valor de B a la variable A en este caso el valor de A se pierde y ambos valen 75.
Si colocamos B=A; estamos asignando el valor de A a la variable B en este caso el valor de B se pierde y ambos valen 30.
Por lo tanto A=B; es distinto de B=A;

Asignaciones Multiples

El valor de una expresión puede ser asignaod a varias variables de tipo simple:

x=y=z = expresión

Por ejemplo :

a= b = c = 25;

Los valores de a, b y c tienen el valor de 25

OPERADORES ARITMÉTICOS

Con los datos constantes o variables de los tipos de datos numéricos se pueden realizar las siguientes operaciones:

+	Suma
-	Resta
-	Multiplicación
*	División
%	Módulo : Residuo de la división de dos números enteros
++	Incrementa en una unidad a la variable Ejemplo : ++x; o x++;
--	Disminuye en una unidad a la variable Ejemplo : --x; o x--;

Con excepción de la operación módulo (%) que se aplica a datos enteros, todas las operaciones dan resultados

a) Del mismo tipo que los operandos si ambos son del mismo tipo
b) Del tipo de mayor rango si los operadores son de tipos distintos

Por ejemplo, si una expresión contiene operaciones con datos de tipo char, float y double, entonces el resultado o valor de la expresión será de tipo double

PRECEDENCIA DE OPERACIONES

En C++ se pueden usar paréntesis para agrupar datos y operaciones.
Al evaluarse una expresión se sigue el siguiente orden de precedencia, de mayor a menor:

1) Paréntesis
2) *, /, %
3) +, -

Las operaciones que tienen igual precedencia se ejecutan de izquierda a derecha.

Ejemplos:
9/2 +7.0/2 = 4 + 3.5 = 7.5 (real)
15/5*3 = 3*3 = 9 (entero)

Conversión Explicita

Un dato tipo simple puede ser convertido en forma explicita a otro tipo de dato de la siguiente forma:

(Tipo_de_dato) expresión;

Ejemplos :

```
Int v=9, r=5;
float  a;  a= (float) v / r;
```

en este ejemplo se convierte momentáneamente la variable v a real y se la divide entre r y el valor resulta 1.8. En el caso de que no se hiciera la conversión explicita el resultado de v/r es de 1.

Abreviaturas

```
S = S + X  es lo mismo que S += X;
S = S - X  es lo mismo que S -= X;
S = S * X  es lo mismo que S += X;
S = S / X  es lo mismo que S /= X;
S = S % X  es lo mismo que S %= X;
```

Componente Label 

Este componente se utiliza para desplegar textos o mensajes estáticos dentro de las formularios, textos tales como encabezados, solicitud al usuario del programa para que proporcione algún dato o información(edad, dame sueldo, etc.).
También es un objeto en C++Builder y por tanto tiene asociados sus propias propiedades y eventos, al mismo tiempo como se está usando dentro del objeto form1, muchas propiedades que se definan para el objeto Form1, el objeto Label1 las va a heredar.
Si bien es cierto que el objeto se llama Label, pero cuando se ponen dentro de una forma C++Builder los va numerando automáticamente, si se ponen tres Labels en Form1, ellos se llaman o simbolizan o se procesan o programan con Label1, Label2, Label3.
Es la *propiedad* Caption, la que lleva el contenido del mensaje que se quiere desplegar en la pantalla, solo click derecho a un lado de la propiedad Caption en el Inspector de Objetos, teniendo seleccionada la caja Label1 en la forma y escribir el texto indicado.

Componente Edit 

Este componente es el mas importante componente visual, su función principal es manejar , todos los procesos de entrada y salida (input/output) al programa.
Este componente permite capturar datos y también como en el caso del componente Label desplegar datos, textos, mensajes o resultados de operaciones de ser necesario, usando la *propiedad* Text del componente Edit.
En principio su valor de default es la palabra Edit1 y es en su propiedad Text donde se modifica, generalmente al principio de un programa se deja en blanco, y al ejecutarse el programa, el usuario lo llena con los datos solicitados o el programa lo llena con el resultado de las operaciones.
Cuando un usuario lo carga con un dato, recordar que el dato almacenado queda de tipo texto, no importa lo que haya escrito el usuario.
Para resolver el problema de usar datos numéricos dentro del Text de un componente Edit, en operaciones matemáticas, solo agregarle a la propiedad Text, las funciones .ToInt() o .ToDouble().

Componente Button 

Es uno de los componentes principales del formulario contiene el código principal del programa y su activación por el usuario provoca que se realicen los principales procesos del problema planteado (aquí es donde se capturan datos, se realizan operaciones, etc.).
De este componente se maneja su propiedad Caption para etiquetarlo con la palabra "OK" o "ACEPTAR" o "CALCULAR" , y su evento OnClick para activarlo, es en dicho evento donde se construye el código del programa.
Este botón también puede activar su evento OnClick, cuando el usuario presione la tecla <ENTER>, solo poner la propiedad Default en true, en este caso el botón de ordenes, se le conoce como botón de default.
Igualmente puede activar su evento OnClick cuando el usuario, presione la tecla <ESC>, solo poner la propiedad Cancel en true, a este caso se le conoce como "CANCEL BUTTON".

1) Programa para mostrar los datos de una Persona

Para hacer este formulario se debe colocar 5 etiquetas (Label) y un boton (Button) .

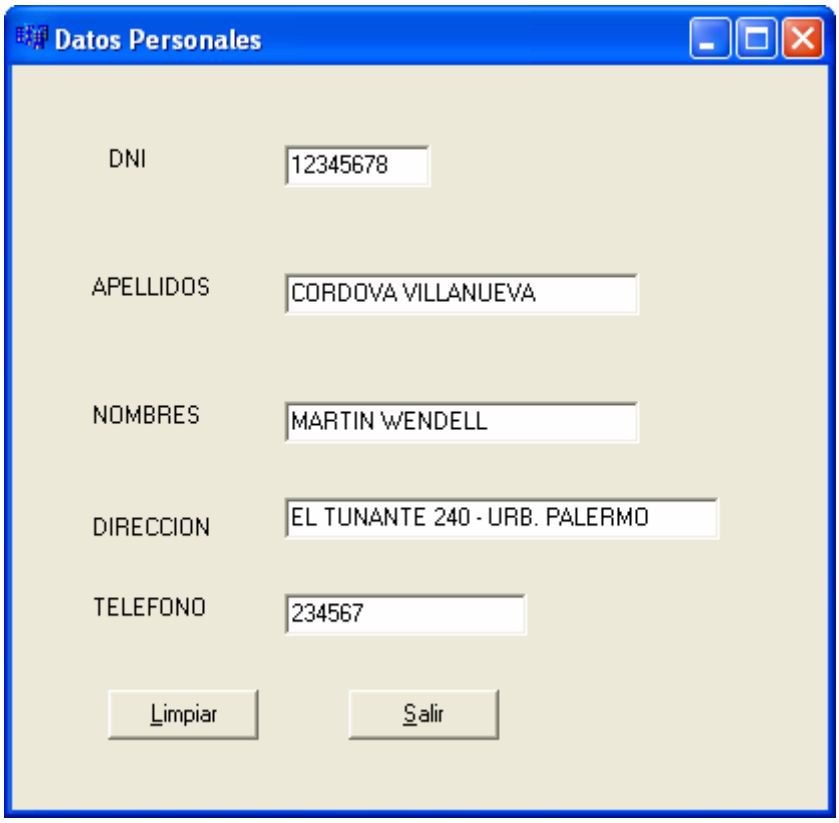
Objeto	Propiedad	Valor
Label	Name Caption	Label1 Universidad Nacional de Trujillo
Label	Name Caption	Label2 Curso : Informatica
Label	Name Caption	Label3 Alumno : Juan Perez
Label	Name Caption	Label4 Ciclo : Primero
Label	Name Caption	Label5 Trujillo – 2006
Button	Name Caption	btnCerrar &Cerrar

En cada control Label En la Propiedad Font cambiar el color de la letra a blue, el tamaño a 14 y la fuente a Comic sans MS

El codigo que se coloca en el evento Clic del Boton btnCerrar es:

```
void __fastcall TForm1::btnCerrarClick(TObject *Sender)
{
  Close(); // Para cerrar el formulario
}
```

2) Programa que permita Ingresar datos de una persona



En este formulario colocamos 5 etiquetas, 5 cajas de texto y dos botones

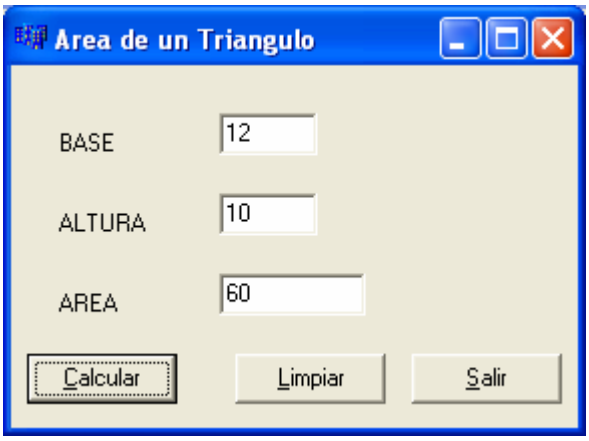
Objeto	Propiedad	Valor
Label	Name Caption	Label1 DNI
Edit	Name Text	EdDni
Label	Name Caption	Label2 APELLIDOS
Edit	Name Text	EdApellidos
Label	Name Caption	Label3 NOMBRES
Edit	Name Text	EdNombres
Label	Name Caption	Label4 DIRECCION
Edit	Name Text	EdDireccion
Label	Name Caption	Label5 TELEFONO
Edit	Name Text	EdTelefon
Button	Name Caption	BtnLimpiar &Limpiar
Button	Name Caption	BtnCerrar &Cerrar

Los códigos en los eventos clic de los botones son:

```
//---------------------------------------------------------------------------

void __fastcall TForm1::btnLimpiarClick(TObject *Sender)
{
  EdDni->Clear();
  EdApellidos->Clear();
  EdNombres->Clear();
  EdDireccion->Clear();
  EdTelefono->Clear();
  EdDni->SetFocus();
}
//---------------------------------------------------------------------------

void __fastcall TForm1::btnSalirClick(TObject *Sender)
{
  Close();
}
//---------------------------------------------------------------------------
```

En las cajas de Texto se usa el metodo Clear para limpiarlas
El metodo SetFocus de un objeto sirve para enfocar el Objeto(en otras palabrar colocar el cursor en el objeto)

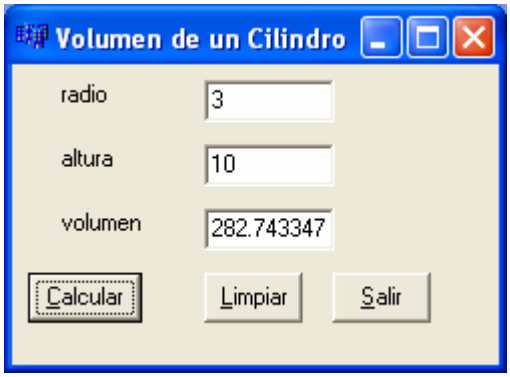
3) Programa para ingresar la base y la altura de un Triangulo y reporte su área.




```
#include <vcl.h>
#pragma hdrstop

#include "Unit1.h"
//---------------------------------------------------------------------------

#pragma package(smart_init)
#pragma resource "*.dfm"

TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
```

```
void __fastcall TForm1::btnCalcularClick(TObject *Sender)
{
      float b,h,a;
      b = EdBase->Text.ToDouble();
      h = EdAltura->Text.ToDouble();
      a = b*h/2;
      EdArea->Text=a;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::btnLimpiarClick(TObject *Sender)
{
    EdBase->Clear();
    EdAltura->Clear();
    EdArea->Clear();
    EdBase->SetFocus();
}
//---------------------------------------------------------------------------
void __fastcall TForm1::btnSalirClick(TObject *Sender)
{
   Close();
}
//---------------------------------------------------------------------------
```

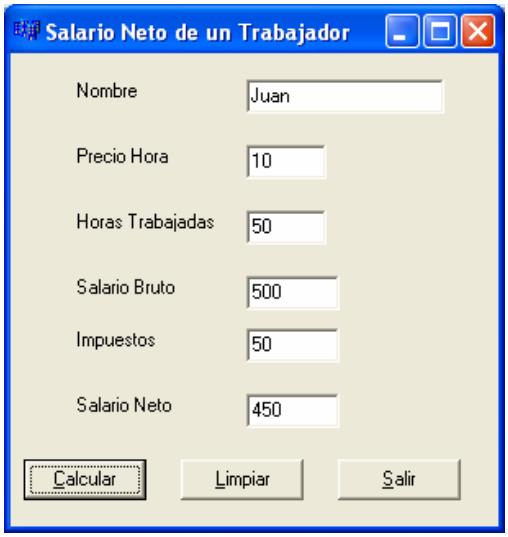
4) Programa para ingresar el radio y la altura de un cilindro y reporte su volumen.




```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"
#include<math.h>
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner): TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm1::btnCalcularClick(TObject *Sender)
{
  float v,r,h;
  r=EdRadio->Text.ToDouble();
  h=EdAltura->Text.ToDouble();
  v=M_PI*pow(r,2)*h;
  EdVolumen->Text=v;
}
```

```
//---------------------------------------------------------------------------
void __fastcall TForm1::btnLimpiarClick(TObject *Sender)
{
   EdRadio->Clear();
   EdAltura->Clear();
   EdVolumen->Clear();
   EdRadio->SetFocus();
}
//---------------------------------------------------------------------------

void __fastcall TForm1::btnSalirClick(TObject *Sender)
{
  Close();
}
//---------------------------------------------------------------------------
```

4) Programa para Ingresar el Nombre, precio Hora, y horas trabajadas de un trabajador, reportar el salario Bruto, el salario Neto y los impuestos.




```
#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner) : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm1::btnCalcularClick(TObject *Sender)
{
  float ph,ht,sb,imp,sn;
  ph=EdPrecioHora->Text.ToDouble();
  ht=EdHorasTrabajadas->Text.ToDouble();
  sb=ph*ht;
  imp=0.10*sb;
  sn=sb-imp;
  EdSalarioBruto->Text=sb;
  EdImpuestos->Text=imp;
  EdSalarioNeto->Text=sn;
}
```

```
//---------------------------------------------------------------------------
void __fastcall TForm1::btnLimpiarClick(TObject *Sender)
{
  EdNombre->Clear();
  EdPrecioHora->Clear();
  EdHorasTrabajadas->Clear();
  EdSalarioBruto->Clear();
  EdImpuestos->Clear();
  EdSalarioNeto->Clear();
  EdNombre->SetFocus();
}
//---------------------------------------------------------------------------
void __fastcall TForm1::btnSalirClick(TObject *Sender)
{
  Close();
}
//---------------------------------------------------------------------------
```

PRACTICA Nº 1

1. Escriba un programa que lea un numero y escriba su cuadrado.

2. Determinar la hipotenusa de un triángulo rectángulo si se ingresan las longitudes de los catetos.

3. Hacer un programa para que se ingresen 2 números y reporte su suma, resta y multiplicación.

4. Hacer un programa que se ingrese una temperatura en grados centígrados (°C) y la reporte en grados Fahrenheit (°F)
   F = 9/5 C + 32

5. Hacer un programa que intercambie el valor de 2 variables numéricas.

6. Hacer un programa para hallar la ganancia de la venta de un producto. Se debe ingresar el precio de costo, precio de venta. Se debe reportar la ganancia.

7. Hacer un programa para que se ingrese una cantidad en kilos y reporte su equivalencia en libras.
   1 kilo = 2.2 libras

8. Calcular la altura que cae un objeto. Se debe ingresar el tiempo recorrido en segundos. ($H = 0.5 * g * T^2$)

9. Calcular la presión de un gas en un recipiente. Se debe ingresar la temperatura (C), el número de moles n y el volumen (lts). (P V = n R T) donde R = 0.082

10. Calcular el espacio recorrido por un móvil. Ingresar Velocidad inicial (m/seg), tiempo (seg.) y aceleración (m/seg2). ($E = Vi * T + 0.5 * A * T^2$)

11. Escriba un programa para que se ingrese una determinada cantidad T de segundos y los convierten en H horas, M minutos y S segundos.

OPERADORES DE RELACION

En lenguaje C un valor distinto de cero puede ser interpretado como verdadero, y el valor cero se considera como falso.

>	mayor que
>=	mayor o igual que
<	menor que
<=	menor o igual que
==	igual que
!=	diferente que

La comparación entre datos numéricos es una expresión que toma uno de los valores 1 o 0, según estos datos hagan verdadera o falsa la comparación respectivamente

OPERADORES LÓGICOS

&&	y
||	o
!	negación

ESTRUCTURA SELECTIVA SIMPLE

if... else

Sirve para escoger una de dos caminos en un programa

```
if(expresión) instruccion1;
else instruccion2;
```

Si la expresión toma un valor distinto de cero (verdadero) se ejecuta la instruccion1; si la expresión toma el valor cero (falso) se ejecuta la instrucción2.

También se puede utilizar

```
if(expresión)
{
     Instrucciones1;
}
else
   {
     Instrucciones2;
   }
```

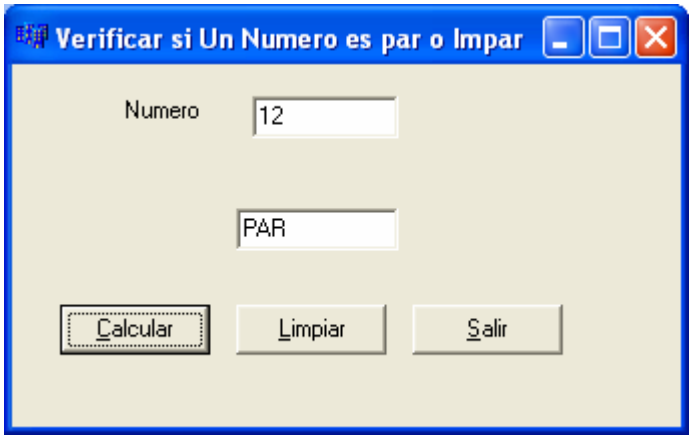
donde Instrucciones1 e Instrucciones2 , son bloques de instrucciones.

La sentencia else es opcional

```
if(expresión)
{
  Instrucciones;
}
```

En este caso se ejecuta las instrucciones si expresión toma un valor no nulo y después se continúa con las otras instrucciones del programa.

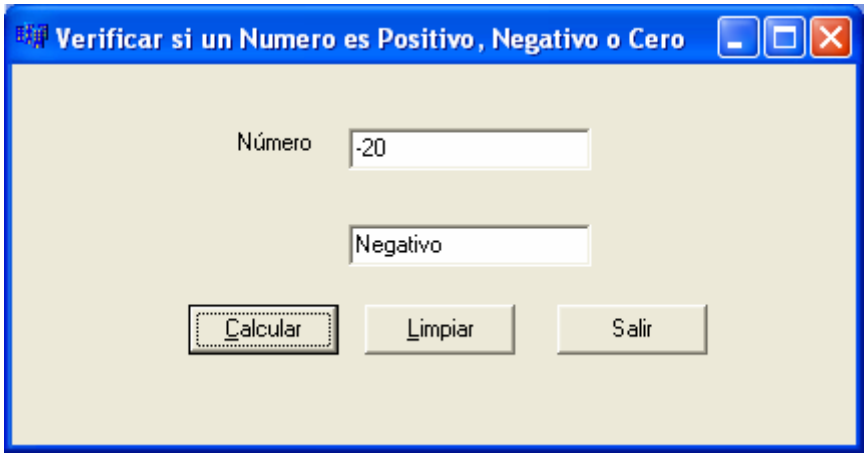
1) Ingresar un número entero y reportar si es par o impar.




```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "Unit1.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------

void __fastcall TForm1::btnSalirClick(TObject *Sender)
{
  Close();

}
//---------------------------------------------------------------------------

void __fastcall TForm1::Button1Click(TObject *Sender)
{
  int N;
  N=EdNum->Text.ToInt();
  if(N%2==0)
    EdResultado->Text="PAR";
  else
    EdResultado->Text="IMPAR";
}


//---------------------------------------------------------------------------
```

```
void __fastcall TForm1::btnLimpiarClick(TObject *Sender)
{
  EdNum->Clear();
  EdResultado->Clear();
  EdNum->SetFocus();

}
//---------------------------------------------------------------------------
```

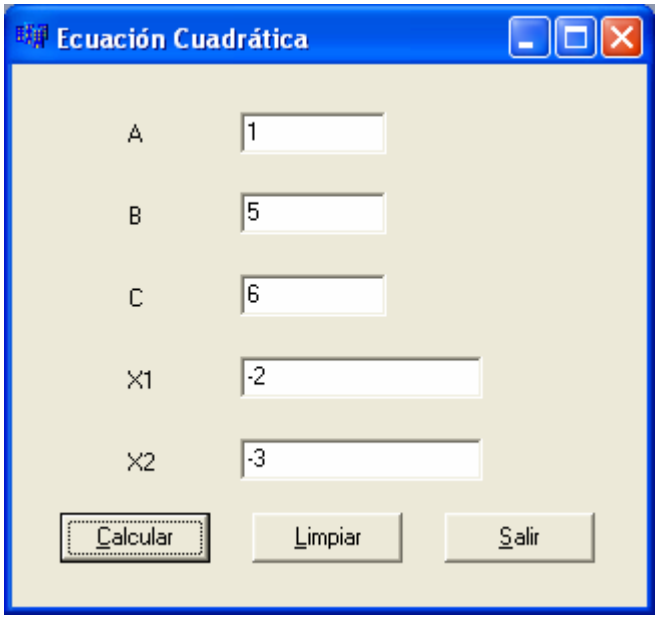
2) Hacer un programa para ingresar un número y se reporte si es positivo, negativo o cero.




```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
      : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm1::btnCalcularClick(TObject *Sender)
{
  int N;
  N=EdNum->Text.ToInt();
  if(N>0)
       EdResultado->Text="Positivo";
  else
     if(N<0)
        EdResultado->Text="Negativo";
     else
        EdResultado->Text="Cero";
}
```

```
//---------------------------------------------------------------------------

void __fastcall TForm1::btnLimpiarClick(TObject *Sender)
{
  EdNum->Clear();
  EdResultado->Clear();
  EdNum->SetFocus();
}
//---------------------------------------------------------------------------

void __fastcall TForm1::btnSalirClick(TObject *Sender)
{
  Close();
}
//---------------------------------------------------------------------------
```

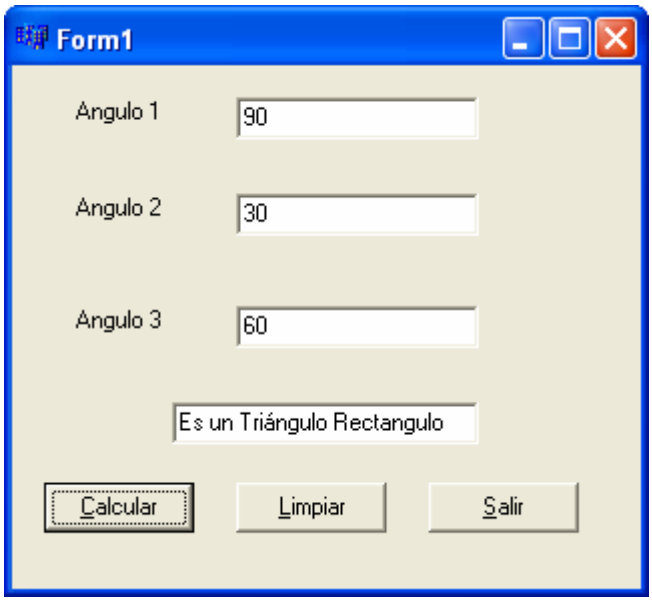
3) Ingresar los coeficientes A,B,C de la Ecuación Cuadrática y reportar las raíces reales.




```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop

#include "Unit1.h"
#include<math.h>
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
```

```
//---------------------------------------------------------------------------
void __fastcall TForm1::btnCalcularClick(TObject *Sender)
{
  float A,B,C,D,X1,X2;

  A=EdA->Text.ToDouble();
  B=EdB->Text.ToDouble();
  C=EdC->Text.ToDouble();
  if(A!=0)
  {
    D=B*B-4*A*C;
    if(D>=0)
    {
      X1=(-B+sqrt(D))/(2*A);
      X2=(-B-sqrt(D))/(2*A);
      EdX1->Text=X1;
      EdX2->Text=X2;
    }
    else
       ShowMessage("Raices Imaginarias");
  }
  else
      ShowMessage("No se puede calcular");
}
//---------------------------------------------------------------------------
void __fastcall TForm1::btnLimpiarClick(TObject *Sender)
{
   EdA->Clear();
   EdB->Clear();
   EdC->Clear();
   EdX1->Clear();
   EdX2->Clear();
   EdA->SetFocus();
}
//---------------------------------------------------------------------------
void __fastcall TForm1::btnSalirClick(TObject *Sender)
{
  Close();
}
//---------------------------------------------------------------------------
```

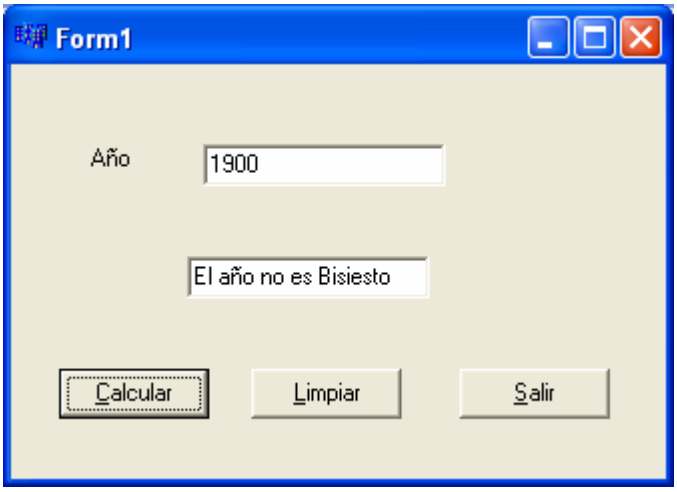
4) Hacer un programa para ingresar el valor de 3 ángulos en grados sexagesimales, y reportar si son los ángulos de un triangulo, además decir si es rectángulo, obtusángulo o acutángulo.




```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------

void __fastcall TForm1::btnCalcularClick(TObject *Sender)
{
  float A, B, C;
  A= EdA->Text.ToDouble();
  B= EdB->Text.ToDouble();
  C= EdC->Text.ToDouble();
  if(A+B+C==180)
  {
    if(A==90 || B==90 || C==90)
      EdMensaje->Text="Es un Triángulo Rectangulo";
    else
       if(A>90 || B>90 || C>90)
          EdMensaje->Text="Es un Triángulo Obtusangulo";
       else
          EdMensaje->Text="Es un Triángulo Acutangulo";
  }
  else
      EdMensaje->Text="No pertenecen a un Triángulo";
}
```

```
//---------------------------------------------------------------------------
void __fastcall TForm1::btnLimpiarClick(TObject *Sender)
{
  EdA->Clear();
  EdB->Clear();
  EdC->Clear();
  EdMensaje->Clear();
  EdA->SetFocus();

}
//---------------------------------------------------------------------------
void __fastcall TForm1::btnSalirClick(TObject *Sender)
{
  Close();
}
//---------------------------------------------------------------------------
```

5) Ingresar el valor de un año y reportar si es Bisiesto. Un año es Bisiesto si es múltiplo de 4 pero no de 100 o es múltiplo de 400.




```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop

#include "Unit1.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
```

```
void __fastcall TForm1::btnCalcularClick(TObject *Sender)
{
  int A;
  A=EdA->Text.ToInt();
  if((A%4==0 && A%100!=0) || A %400==0)
     EdMensaje->Text="El año es Bisiesto";
  else
     EdMensaje->Text="El año no es Bisiesto";
}
//---------------------------------------------------------------------------

void __fastcall TForm1::btnLimpiarClick(TObject *Sender)
{
  EdA->Clear();
  EdMensaje->Clear();
  EdA->SetFocus();

}
//---------------------------------------------------------------------------
void __fastcall TForm1::btnSalirClick(TObject *Sender)
{
  Close();
}
//---------------------------------------------------------------------------
```

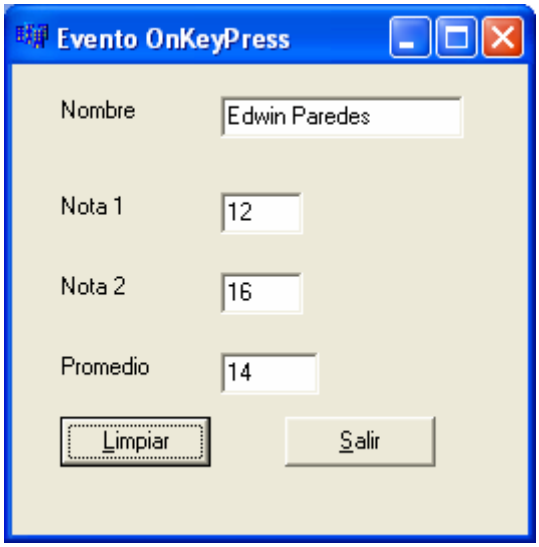
El Evento OnKeyPress

Este evento solo devuelve el caracter que el usuario ha presionado. El carácter debe caer dentro del rango de los caracteres ASCII.
OnKeypress interpreta combinaciones de teclas solo en la medidas que ellas evaluan un carácter ASCII. Asi por Ejemplo OnKeyPress reconoce el presultado de Shift+A (Una letra mayuscula) pero no evalua otras combinaciones de teclas y no reconoce teclas de funcion o teclas que no son ASCII como ctrl., Alt, Insert, Page Down etc.

El Evento OnKeyPress contiene un parámetro Key de tipo char (carácter). Por lo tanto si se desea probar que tecla se ha presionado , verificamos ese carácter.

Ejemplo

El siguiente programa ingresa el Nombre y las notas de un alumno y reporta el Promedio.
En este programa se programa el Evento OnKeyPress de las cajas de Texto para validarlas de tal manera que cuando se ingrese el Nombre solo permita ingresar letras el carácter barra espaciadora (codigo ASCII 32) y el backSpace (Codigo ASCII 8).
Cuando se ingresen las Notas permita ingresar solo Numeros y el carácter backSpace.
Ademas cuando se presiona Enter en la Caja de Texto del Nombre, el cursor se coloque en la siguiente Caja de Texto de la Nota1 y asi sucesivamente.
Como se ve no hay un Boton Calcular porque el Calculo se ha programado en el Evento OnKeypress de la caja de Texto edNota2, cuando se presiona la tecla Enter se hace el calculo y luego el cursor se va al Boton btnNuevo.




```
void __fastcall TForm1::edNombreKeyPress(TObject *Sender, char &Key)
{
  if(Key==13)
    edNota1->SetFocus();
  if( !((Key>='A' && Key<='Z') ||(Key>='a' && Key<='z') || Key==8 || Key==32))
      Key=0;
}

//---------------------------------------------------------------------------
```

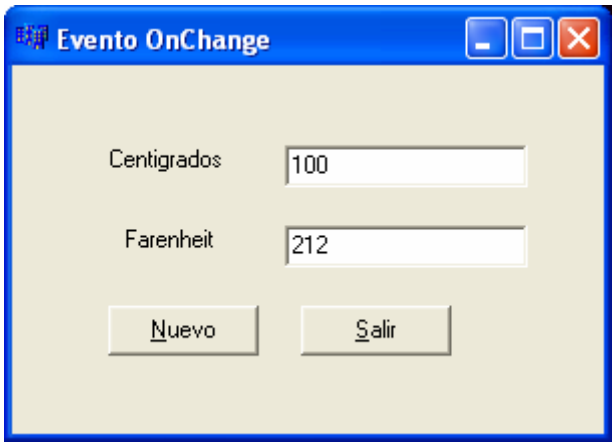
```
void __fastcall TForm1::btnSalirClick(TObject *Sender)
{

  Close();
}
//---------------------------------------------------------------------------

void __fastcall TForm1::edNota1KeyPress(TObject *Sender, char &Key)
{
 if(Key==13)
   edNota2->SetFocus();
 if( !((Key>='0' && Key<='9') || Key==8))
     Key=0;
}
//---------------------------------------------------------------------------

void __fastcall TForm1::edNota2KeyPress(TObject *Sender, char &Key)
{
 if(Key==13)
 {
   int N1,N2;
   float prom;
   N1=edNota1->Text.ToInt();
   N2=edNota2->Text.ToInt();
   prom=(N1+N2)/2.0;
   edPromedio->Text=prom;
   btnLimpiar->SetFocus();
 }
 if( !((Key>='0' && Key<='9') || Key==8))
     Key=0;
}
//---------------------------------------------------------------------------

void __fastcall TForm1::btnLimpiarClick(TObject *Sender)
{
  edNombre->Clear();
  edNota1->Clear();
  edNota2->Clear();
  edPromedio->Clear();
  edNombre->SetFocus();
}
//---------------------------------------------------------------------------
```

El Evento OnChange

Este evento se produce cuando sucede algún cambio en el Objeto que se esta trabajando. Por ejemplo cuando se esta editando un texto en una Caja de texto.

Ejemplo

En el siguiente programa cuando se va cambiando el valor de la caja de Texto donde estan los grados Centigrados automáticamente se va calculando los grados Fahrenheit. Esto se programa en el Evento OnChange de la Caja de Texto edCentigrados.




```
void __fastcall TForm1::edCentigradosChange(TObject *Sender)
{
  float c,f;
  if(edCentigrados->Text!="")
  {
     c=edCentigrados->Text.ToDouble();
     f=9*c/5+32;
     edFarenheit->Text=f;
  }
  else
     edFarenheit->Clear();

}
//---------------------------------------------------------------------------
void __fastcall TForm1::btnSalirClick(TObject *Sender)
{
  Close();
}
//---------------------------------------------------------------------------
void __fastcall TForm1::btnNuevoClick(TObject *Sender)
{
  edCentigrados->Clear();
  edFarenheit->Clear();
  edCentigrados->SetFocus();
}
//---------------------------------------------------------------------------
```

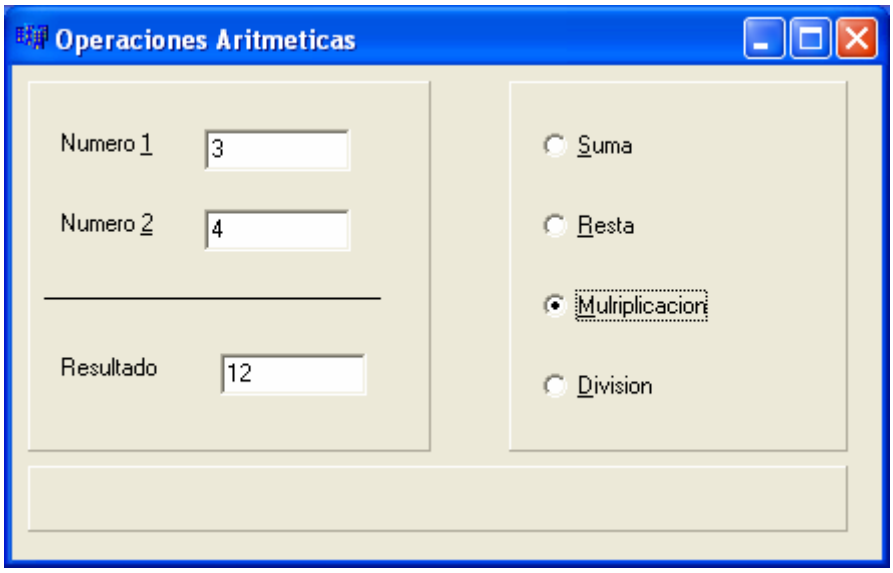
Componente Radio Button (Botones de Radio)

Se utilizan para presentar al usuario un conjunto de opciones mutuamente excluyentes entre si, es decir si el usuario selecciona un componente RadioButton todos los demás componentes RadioButton en el formulario, se desmarcan solos, o se deseleccionan solos.
Como los botones de radio se colocan en grupos, si existen grupos diferentes para poder diferenciar los grupos, se deben usar los controles Panel o groupBox para agruparlos. El procedimiento es el siguiente se coloca primero el Panel o groupBox y luego dentro de ellos se colocan los botones de Radio.

Propiedades
Name: Su Nombre por defecto es RadioButtonN. Nosotros vamos a utilizar rbNombre.
Caption: En esta propiedad Caption se coloca el texto que identifica el propósito del botón.
Checked: Establece si un RadioButton esta seleccionado. Tiene el valor **true** si el RadioButton esta seleccionado y **false** si el RadioButton no esta seleccionado

Eventos
Cuando se hace doble clic en el RadioButton se activa el evento OnClick donde se colocara el codigo, y este se ejecturá si se escoge el radioButton cuando el programa esta corriendo.

Ejercicio.
Programa para ingresar 2 numeros y luego se escoja la operación que se quiere hacer con ellos y reportar el resultado



En este programa vamos a usar tres paneles.
En el primer panel se colocan 4 Etiquetas y 3 cajas de Texto.
En el segundo Panel se colocan los Botones de Radio.
El ultimo panel esta solo, cuando se hace clic en el se cierra el Formulario.

En el evento OnClick de cada RadioButton se programa la operación respectiva, tal como se muestra el código mostrado.

Objeto	Propiedad	Valor
Panel	Name Caption	Panel1
Panel	Name Caption	Panel2
Panel	Name Caption	Panel3
Label	Name Caption	Label1 Numero &1

Edit	Name Text	EdN1
Label	Name Caption	Label2 Numero &2
Edit	Name Text	EdN2
Label	Name Caption	Label3 _____
Label	Name Caption	Label4 Resultado
Edit	Name Text	EdDireccion
Label	Name Caption	Label5 TELEFONO
Edit	Name Text	EdTelefon
RadioButton	Name Caption	rbSuma &Suma
RadioButton	Name Caption	rbResta &Resta
RadioButton	Name Caption	rbMultiplicacion &Multiplicación
RadioButton	Name Caption	rbDivision &Division

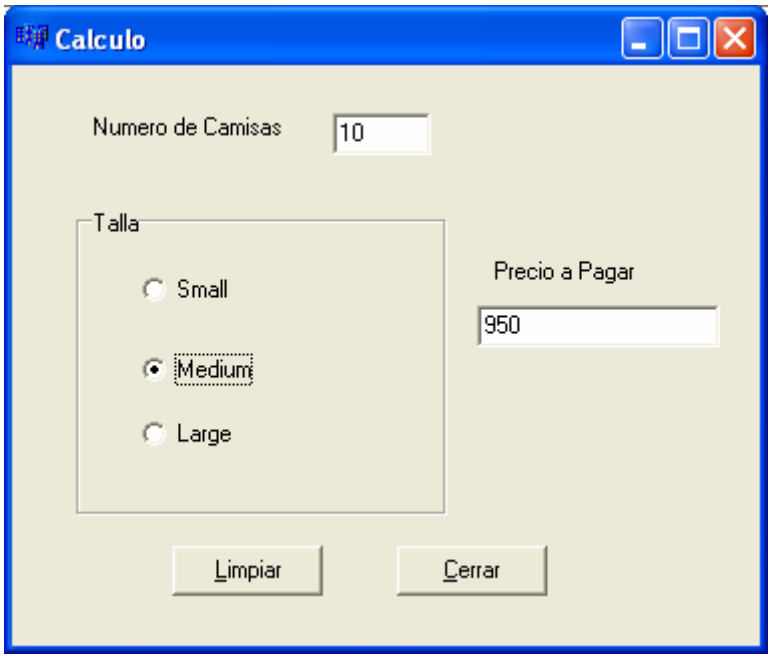
```
void __fastcall TForm1::rbSumaClick(TObject *Sender)
{
  double n1,n2,resultado;
  if(edN1->Text=="")
      n1=0;
  else
     n1=edN1->Text.ToDouble();
  if(edN2->Text=="")
      n2=0;
  else
     n2=edN2->Text.ToDouble();
  resultado=n1+n2;
  edResultado->Text=resultado;

}
/

/---------------------------------------------------------------------------

void __fastcall TForm1::rbRestaClick(TObject *Sender)
{
  double n1,n2,resultado;
  if(edN1->Text=="")
      n1=0;
  else
     n1=edN1->Text.ToDouble();
  if(edN2->Text=="")
      n2=0;
  else
     n2=edN2->Text.ToDouble();
  resultado=n1-n2;
  edResultado->Text=resultado;
}

//---------------------------------------------------------------------------
```

```
void __fastcall TForm1::rbMultiplicacionClick(TObject *Sender)
{
  double n1,n2,resultado;
  if(edN1->Text=="")
      n1=0;
  else
     n1=edN1->Text.ToDouble();
  if(edN2->Text=="")
      n2=0;
  else
     n2=edN2->Text.ToDouble();
  resultado=n1*n2;
  edResultado->Text=resultado;

}
//---------------------------------------------------------------------------
void __fastcall TForm1::rbDivisionClick(TObject *Sender)
{
  double n1,n2,resultado;
  if(edN1->Text=="")
      n1=0;
  else
     n1=edN1->Text.ToDouble();
  if(edN2->Text=="")
  {
    MessageBox(NULL,"El Numero 2 no puede estar vacio",
    "Operaciones algebraicas",MB_OK);
    edN2->SetFocus();
  }
  else
     if(edN2->Text==0)
     {
       MessageBox(NULL,"No se puede dividir por Cero",
        "Operaciones algebraicas",MB_OK);
       edN2->SetFocus();
     }
     else
     {
      n2=edN2->Text.ToDouble();
      resultado=n1/n2;
      edResultado->Text=resultado;
      }
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Panel3Click(TObject *Sender)
{
   Close();
}
```

2) Determinar el precio que debe pagarse por la compra de una cantidad de camisas del mismo tipo, si el precio de las camisas talla S es de 85, de talla M es de 95, y la talla L es de 100. Se debe ingresar la cantidad de camisas a comprar y la talla

Solucion

En este programa usamos un GroupBox para agrupar los botones de Radio.




```
void __fastcall TForm1::rbSClick(TObject *Sender)
{
  int nc;
  float pp;
  if(edNumCamisas->Text=="")
  {
    ShowMessage("Ingrese numero por favor");
    rbS->Checked=false;
    edNumCamisas->SetFocus();
  }
  else
  {
    nc=edNumCamisas->Text.ToInt();
    pp=nc*85;
    edPP->Text=pp;
   }
}
//---------------------------------------------------------------------------
void __fastcall TForm1::rbMClick(TObject *Sender)
{
  int nc;
  float pp;
  if(edNumCamisas->Text=="")
  {
    ShowMessage("Ingrese numero por favor");
    rbM->Checked=false;
    edNumCamisas->SetFocus();
  }
  else
  {
    nc=edNumCamisas->Text.ToInt();
    pp=nc*95;
    edPP->Text=pp;
   }}
//---------------------------------------------------------------------------
void __fastcall TForm1::rbLClick(TObject *Sender)
```

```
{
  int nc;
  float pp;
  if(edNumCamisas->Text=="")
  {
    ShowMessage("Ingrese numero por favor");
    rbL->Checked=false;
    edNumCamisas->SetFocus();
  }
  else
  {
    nc=edNumCamisas->Text.ToInt();
    pp=nc*100;
    edPP->Text=pp;
   }
}
//---------------------------------------------------------------------------

void __fastcall TForm1::btnLimpiarClick(TObject *Sender)
{
  edNumCamisas->Clear();
  edPP->Clear();
  rbS->Checked=false;
  rbM->Checked=false;
  rbL->Checked=false;
  edNumCamisas->SetFocus();
}
//---------------------------------------------------------------------------



void __fastcall TForm1::btnCerrarClick(TObject *Sender)
{
  Close();
}
//---------------------------------------------------------------------------
```

Componente RadioGroup 

Aunque es común agrupar un conjunto de RadioButton dentro de componentes Panel y GroupBox, C++Builder proporciona este componente RadioGroup que esta especializado en la agrupación de RadioButton.

Un componente u objeto RadioGroup es una caja especial que solo contiene componentes RadioButton, también cuando el usuario marca o selecciona uno de ellos, todos los demás se desmarcan o deseleccionan .

Para añadir los RadioButton a el componente RadioGroup, solo editar la propiedad Items en el Inspector de Objetos, que entonces nos muestra el minieditor de strings ya visto y practicado en el tema de ListBox, solo recordar que cada renglón en el editor corresponderá a un RadioButton dentro del RadioGroup.

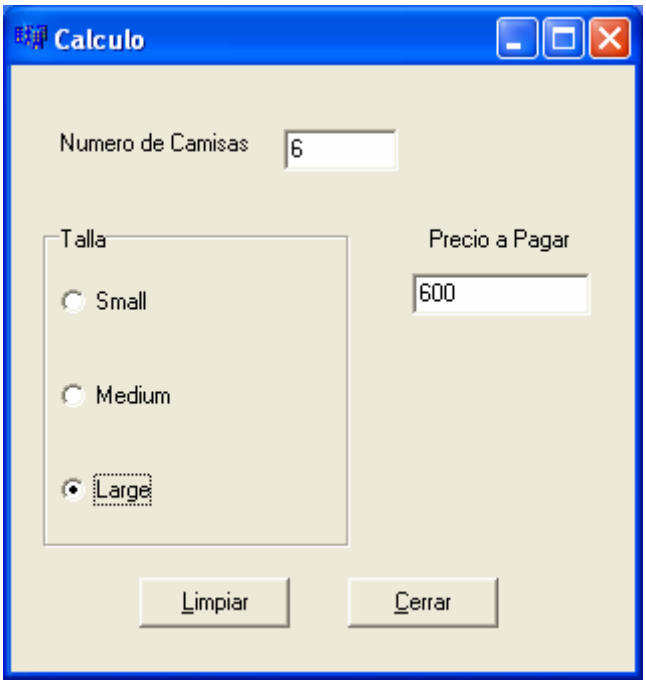
Para procesar o programar el RadioButton seleccionado por el usuario, solo usar la propiedad ItemIndex que queda cargada con el numero de RadioButton seleccionado por el usuario (Los indices comienzan en cero)

Determinar el precio que debe pagarse por la compra de una cantidad de camisas del mismo tipo, si el precio de las camisas talla S es de 85, de talla M es de 95, y la talla L es de 100. Se debe ingresar la cantidad de camisas a comprar y la talla.

Solucion

Colocaremos un RadioGroup, en el Object Inspector escogemos la propiedad Ítems y colocamos las opciones: Small, Médium, Large.

Y la programación en el evento OnClick del Radio Grau, se hace con la Propiedad ItemIndex que indica el indice del radio Seleccionado (los indices comienzan desde 0) si no se selección ningun radiobutton devuelve -1.




```
void __fastcall TForm1::rgTallaClick(TObject *Sender)
{
  int nc,i;
  float pp;
  nc=edNumCamisas->Text.ToInt();
  i= rgTalla->ItemIndex;
  if(i==0)
  {
    pp=nc*85;
    edPP->Text=pp;
  }
  else
        if(i==1)
        {
          pp=nc*95;
          edPP->Text=pp;
        }
       else
             if(i==2)
             {
               pp=nc*100;
               edPP->Text=pp;
             }

}

//---------------------------------------------------------------------------
```

```
void __fastcall TForm1::btnLimpiarClick(TObject *Sender)
{
  edNumCamisas->Clear();
  edPP->Clear();
  rgTalla->ItemIndex=-1;
  edNumCamisas->SetFocus();



}
//---------------------------------------------------------------------------
void __fastcall TForm1::btnCerrarClick(TObject *Sender)
{
  Close();
}
//---------------------------------------------------------------------------
```

Componente CheckBox 

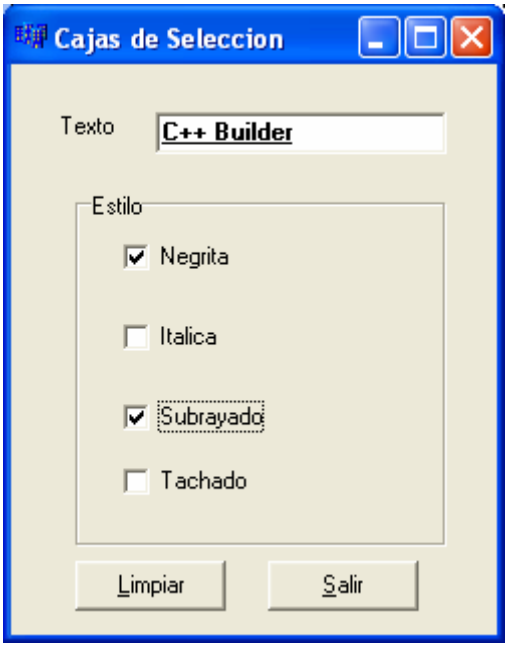
El control CheckBox, o casilla de verificación, permite elegir una opción (activada / desactivada, True/False) que el usuario puede establecer o anular haciendo click. Una X en una casilla de verificación indica que está seleccionada, activada, o con valor True. Cada casilla de verificación es independiente de las demás que puedan existir en el formulario, pudiendo tomar cada una de ellas el valor True o False, a voluntad del operador.
Aunque puede aparecer solo, un Check box a veces viene acompañado con otros checks, permitiendo al usuario seleccionar varias opciones.

Propiedades
Name: Su Nombre por defecto es CheckBoxN. Nosotros vamos a utilizar chkNombre.
Caption: En esta propiedad Caption se coloca el texto que identifica el propósito del Check
Checked: Establece si un checkBox esta seleccionado. Tiene el valor **true** si el Check esta seleccionado y **false** si el Check no esta seleccionado

Eventos
Cuando se hace doble clic en el CheckBox se activa el evento OnClick donde se colocara el codigo, y este se ejecturá si se escoge el CheckBox cuando el programa esta corriendo.

Ejemplo

Hacer un programa cuya finalidad será solicitar un Texto y mostrarlo aplicando un cierto estilo, que se podrá seleccionar mediante unas cajas de Selección.
En un texto se pueden aplicar varios estilos a la vez, por eso es apropiado usar checkbox.




```
void __fastcall TForm1::chkNegritaClick(TObject *Sender)
{
  if(chkNegrita->Checked)
    edTexto->Font->Style =edTexto->Font->Style+ TFontStyles()<<fsBold;
  else
    edTexto->Font->Style =edTexto->Font->Style+TFontStyles()>>fsBold;

}
//---------------------------------------------------------------------------
```

```
void __fastcall TForm1::chkItalicaClick(TObject *Sender)
{
  if(chkItalica->Checked)
    edTexto->Font->Style=edTexto->Font->Style+TFontStyles()<<fsItalic;
  else
    edTexto->Font->Style=edTexto->Font->Style+TFontStyles()>>fsItalic;

}
//---------------------------------------------------------------------------
void __fastcall TForm1::chkSubrayadoClick(TObject *Sender)
{
  if(chkSubrayado->Checked)
    edTexto->Font->Style=edTexto->Font->Style+TFontStyles()<<fsUnderline;
  else
    edTexto->Font->Style=edTexto->Font->Style+TFontStyles()>>fsUnderline;


}
//---------------------------------------------------------------------------
void __fastcall TForm1::chkTachadoClick(TObject *Sender)
{
  if(chkTachado->Checked)
      edTexto->Font->Style=edTexto->Font->Style+TFontStyles()<<fsStrikeOut;
  else
      edTexto->Font->Style=edTexto->Font->Style+TFontStyles()>>fsStrikeOut;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::btnLimpiarClick(TObject *Sender)
{
  edTexto->Clear();
  chkNegrita->Checked=false;
  chkSubrayado->Checked=false;
  chkItalica->Checked=false;
  chkTachado->Checked=false;
  edTexto->SetFocus();

}
//---------------------------------------------------------------------------
void __fastcall TForm1::btnSalirClick(TObject *Sender)
{
  Close();
}
//---------------------------------------------------------------------------
```

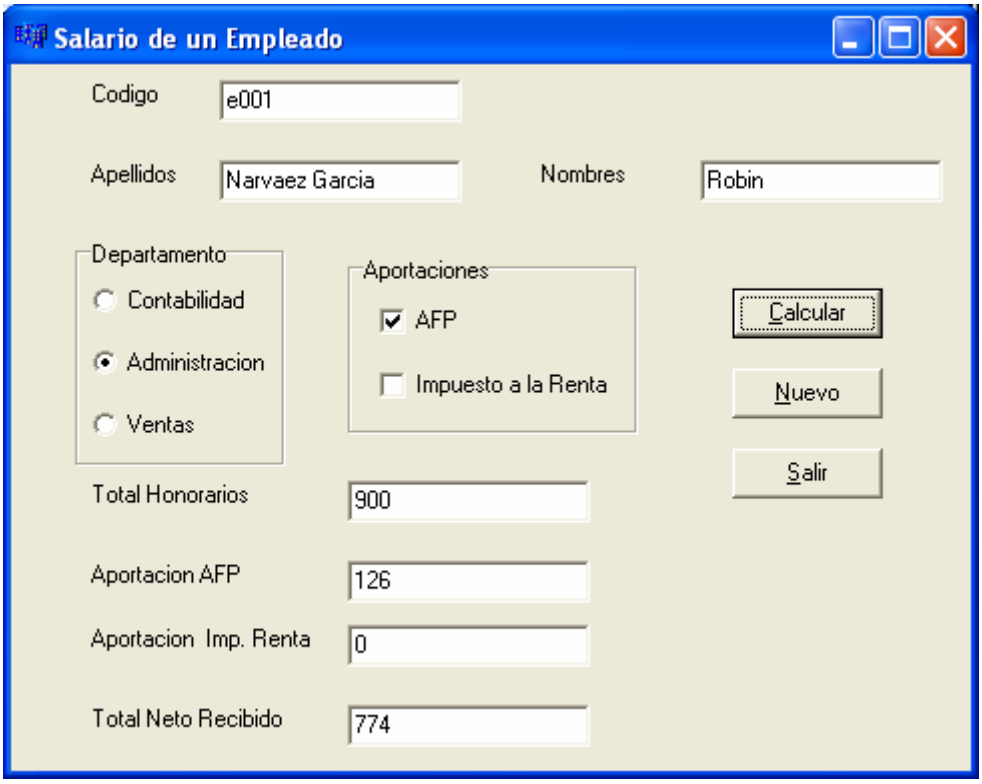
Para añadir o quitar un Estilo de Letra (Negrita, italica, subrayado, italica) al Texto de la Caja de Texto se usa:

```
edTexto->Font->Style = edTexto->Font->Style + TFontStyles()<<fsBold; para añadir Negrita

edTexto->Font->Style =edTexto->Font->Style+TFontStyles()>>fsBold; para quitar Negrita
```

Ejercicios

Hacer un programa que permita calcular el pago total a los profesionales independientes que laboran en una empresa la empresa tiene los siguientes departamentos:

Departamentos	Honorarios	
Contabilidad	s/. 1000	Retenciones :
Administración	s/. 900	AFP: 14%
Ventas	s/. 800	Imp. Renta: 10%



En este ejercicio hacemos la programación en el botón btnCalcular, allí preguntamos que valores tienen los botones del radioGroup rgDepartamento y los checkbox chkAFP y chkIR de acuerdo a eso hacemos los calculos.

```
void __fastcall TForm1::btnCalcularClick(TObject *Sender)
{
  float honorarios,afp,ir,totalNeto;


  switch(rgDepartamento->ItemIndex)
  {
    case 0 : honorarios=1000;break;
    case 1 : honorarios=900;break;
    case 2 : honorarios=800;break;
  }
  if(chkAFP->Checked)
      afp=0.14*honorarios;
  else
      afp=0;
  if(chkIR->Checked)
     ir=0.10*honorarios;
  else
     ir=0;
  totalNeto=honorarios-afp-ir;

  edHonorarios->Text=honorarios;
```

```
  edAFP->Text=afp;
  edIR->Text=ir;
  edTotalNeto->Text=totalNeto;
}

//---------------------------------------------------------------------------
void __fastcall TForm1::btnNuevoClick(TObject *Sender)
{
  edCodigo->Clear();
  edApellidos->Clear();
  edNombres->Clear();
  rgDepartamento->ItemIndex=-1;
  chkAFP->Checked=false;
  chkIR->Checked=false;
  edHonorarios->Clear();
  edAFP->Clear();
  edIR->Clear();
  edTotalNeto->Clear();
  edCodigo->SetFocus();
}
//---------------------------------------------------------------------------
void __fastcall TForm1::btnSalirClick(TObject *Sender)
{
  Close();
}
//---------------------------------------------------------------------------
```

Ejercicios Propuestos

1. Hacer un programa que permita ingresar los datos personales de los trabajadores de una constructora y permita calcular su pago semanal.
   Pago diarios según categoría :

Maestro	s/. 32.00	Bonificación:	
Operario	s/. 29.00	Estable:	20% del total de ingresos
Peón	s/. 25.00	Contratado	10% del total de ingresos.

2. Hacer un programa que permita calcular el pago de un trabajador de acuerdo a lo siguiente :
   Bonificación : respecto al sueldo
   Descuento : respecto al sueldo + bonificación

	Masculino	Femenino
Soltero	8%	10%
Casado	10%	12%

	Masculino	Femenino
Soltero	6%	5%
Casado	4%	3%

3. Hacer un programa que permita calcular el pago de un trabajador de acuerdo a lo siguiente :
   Bonificación : respecto al sueldo

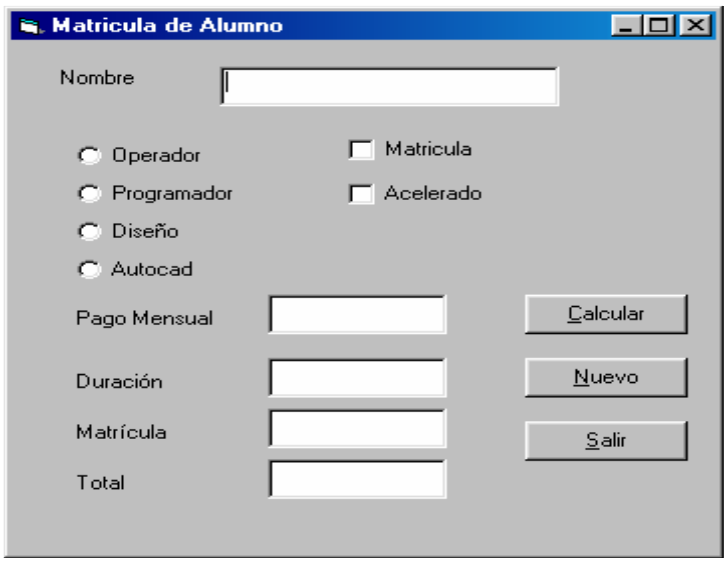
	Estable	No Estable
A	20%	17%
B	18%	15%
C	15%	14%
D	12%	10%

Descuento : respecto al sueldo + bonificación
Estable : 6% No estable : 4%

4. Hacer un programa que permita calcular el pago mensual, la duración, la matricula y el pago total de un Alumno de acuerdo a lo siguiente :

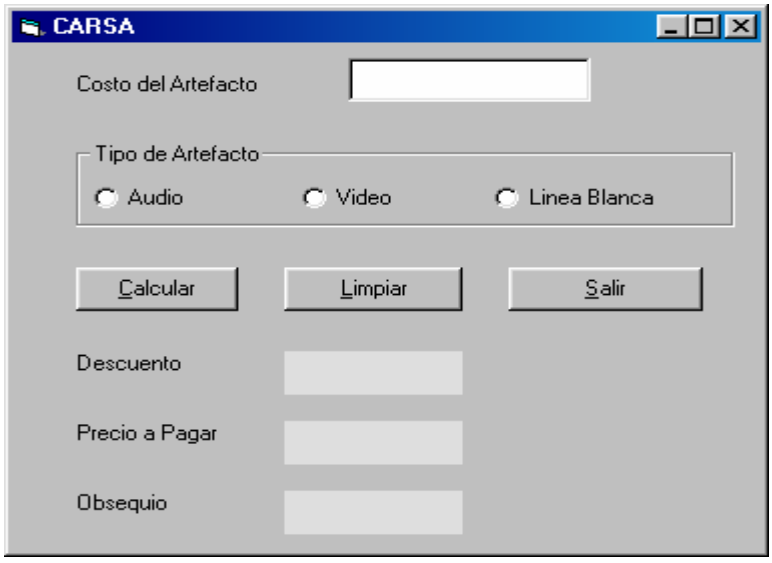
	Costo Acelerado (3 meses)	Normal (6 meses)	Matricula
Operador	230	100	20
Programador	280	130	25
Diseño	320	150	30
Autocad	350	160	40

Total = pago mensual * nro de meses + matricula



5. De acuerdo al tipo de artefacto se aplicara un descuento y se dará un obsequio tal como se señala en la tabla.

Tipo	**Tasa de Decuento** (Sobre el costo)	**Obsequio**
Audio	8%	Licuadora
Vídeo	9%	Batidora
Linea Blanca	0%	Plancha



PRACTICA Nº 2

1. Hacer un programa para que se ingrese 2 números y se reporte el mayor de ellos.

2. Escriba un programa para determinar sí un número entero A es divisible por otro B.

3. Calcule el interés mensual generado por un capital. La tasa de interés mensual depende del capital que fue depositado. Si el capital es mayor que cero pero menor de 500, la tasa de interés será del 2% mensual. Si el capital es al menos 500 pero menor o igual a 1500 entonces la tasa de interés es de 4.5%. Si el capital es mayor que 1500 la tasa de interés es del 9%. Se debe ingresar el capital y reportar el interés

4. Hacer un programa de tal manera que se ingrese las 2 evaluaciones de un alumno y reporte APROBADO si el promedio es mayor o igual a 10.5 y DESAPROBADO en caso contrario.

5. La comisión de las ventas totales es como sigue:

a) Si VENTAS < S/.80, entonces no hay comisión.
b) Si S/.80 <= VENTAS <= S/.600 entonces la comisión es igual al 12% de las ventas.
c) Si VENTAS > 600 entonces la comisión es igual al 15% de las ventas.

Hacer un programa para que se ingrese las ventas y se reporte la comisión.

6. Hacer un programa para calcular el pago semanal de un trabajador. Se debe ingresar el nombre, pago por hora y el número de horas trabajadas. Si normalmente se trabaja 40 horas a la semana y por cada hora extra trabajada se paga 1.5 veces la hora normal, reportar el nombre y el pago semanal del trabajador.

7. Se repartirá la herencia entre los hijos de un señor como sigue: Si la cantidad de hijos es menor que 4; se repartirá exactamente entre el número de hijos; si son 4 o más hijos, la mitad le tocará al hermano mayor y el resto se dividirá entre los demás hermano. Hacer un programa para que reporte cuando le corresponde a cada hijo. Se debe ingresar la herencia y el número de hijos.

8. En un triángulo se cumple lo siguiente:

s>a, s>b, s>c donde s: semiperímetro a, b, c : Lados del triángulo

Hacer un programa para que se ingresen los valores de los lados del triángulo y si estos valores cumplen las condiciones calcular el área del triángulo en caso contrario reportar 'DATOS INCORRECTOS'.

$$AREA = \sqrt{s(s-a)(s-b)(s-c)}$$

9. Calcular el valor de la función de acuerdo a lo siguiente :

$y = x^2 + 5$ Si x<=0

$y = 3x - 1$ Si 0 < x < 2

$y = x^2 - 4x + 5$ Sí x>=2

Se debe ingresar el valor de x y reportar el valor de y.

10. Los empleados de una fábrica trabajan en dos turnos: diurno y nocturno. Se desea calcular el jornal diario de acuerdo a los siguientes puntos:

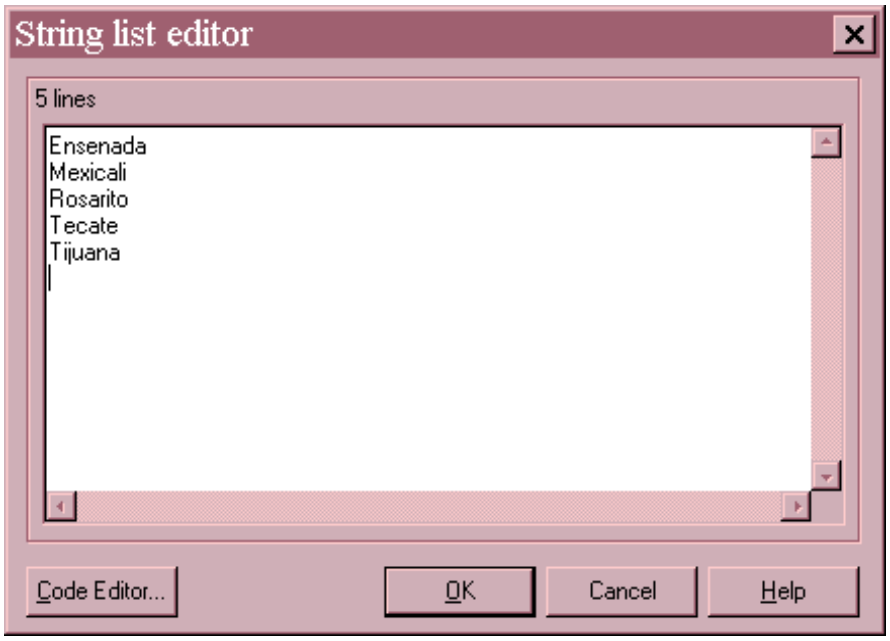
- La tarifa de las horas diurnas es de S/.1.5
- La tarifa de las horas nocturnas es de S/. 2.25
- En caso de ser domingo la tarifa aumentará en S/.1 en el turno diurno y S/. 1.25 en el turno nocturno.

Se debe leer el turno, las horas trabajadas y el día de la semana.

Ingresar el valor de 3 lados y reportar si forman un triangulo. Si estos forman un triangulo reportar si es equilátero, isósceles o escaleno. Si no forman un triangulo reportar “Datos no forman triangulo”.

Componete ComboBox 

Combina una caja de Texto con una lista. El usuario puede seleccinar un elemento de la lista o puede tipear directamente en la caja de texto.
Este componente ComboBox tiene dos partes, una parte de encabezado, para poner el nombre del grupo de respuestas, que se carga usando la propiedad Text del componente. La segunda parte es la lista de opciones o respuestas que se debe cargar al tiempo de diseño de la ventana, en el momento de poner el componente ComboBox1, solo hacer dobleclick a un lado de la propiedad Items en el Inspector de objetos y sale el siguiente editor de strings:



Al momento de ejecución del programa, toda la lista de respuestas, estarán a la vista del usuario, para que este último la seleccione.
Recordar que el usuario al momento de ejecución del programa, solo vera el encabezado, para seleccionar su respuesta deberá apretar la flechita que esta a un lado del encabezado.

El comboBox a diferencia de la lista no tiene MultiSelect

Propiedades.
Text: Establece o devuelve el Elemento seleccionado.
Ítems: Similar al ListBox.
ItemIndex: Devuelve o establece el indice del elemento seleccionado.
Sorted: ordena automáticamente los elementos alfabéticamente.

Métodos:
Clear: Borra todos los elementos del ComboBox
Eventos
OnChange: En el ComboBox se produce este Evento cuando seleccionamos un elemento de la Lista.

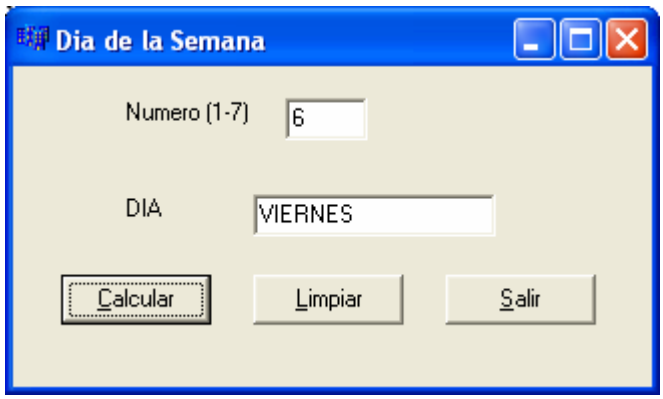
ESTRUCTURA SELECTIVA MÚLTIPLE : switch

```
switch(expresión)
{
  case  cte1 :  Instrucciones1;
                break;
  case  cte2 :  Instrucciones2:
                break;
  case  cte3 :   Instrucciones3;
                break;
   .
   .
   .
   default :   Instrucciones;
}
```

expresión: Puede ser entero o carácter no puede ser de otro tipo. Esta expresión se compara con cada uno de las constantes que se encuentran en los case, si es igual a alguna de ellas se ejecutan las expresiones correspondientes y se sale del switch. Si no es igual a ninguna de ellas se ejecutan las instrucciones que siguen a default.
La sentencia default es opcional.
En la sentecia switch solo se compara por igualdad no por otra relación.

Ejercicios Resueltos

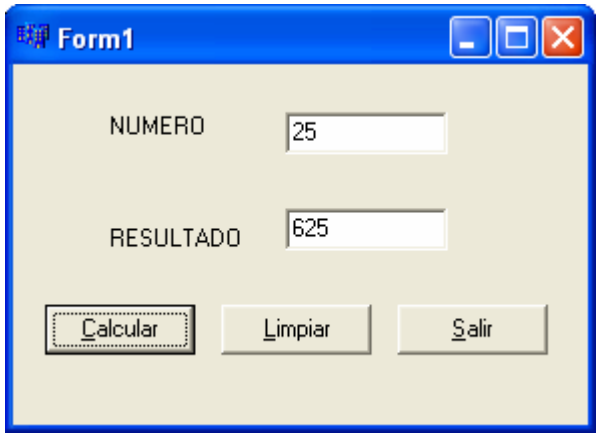
1) Hacer un programa para ingresar un número entre 1 y 7 que represente el número de día de la Semana y reporte el nombre del día que le corresponde.




```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop

#include "Unit1.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
```

```
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm1::btnCalcularClick(TObject *Sender)
{
  int N;
  N=EdNum->Text.ToInt();
  switch(N)
  {
    case 1 : EdDia->Text="DOMINGO";break;
    case 2 : EdDia->Text="LUNES";break;
    case 3 : EdDia->Text="MARTES";break;
    case 4 : EdDia->Text="MIERCOLES";break;
    case 5 : EdDia->Text="JUEVES";break;
    case 6 : EdDia->Text="VIERNES";break;
    case 7 : EdDia->Text="SABADO";break;
    default:
          ShowMessage("Numero fuera de Rango ");
  }
}
//---------------------------------------------------------------------------
void __fastcall TForm1::btnLimpiarClick(TObject *Sender)
{
  EdNum->Clear();
  EdDia->Clear();
  EdNum->SetFocus();
}
//---------------------------------------------------------------------------
void __fastcall TForm1::btnSalirClick(TObject *Sender)
{
  Close();
}
//---------------------------------------------------------------------------
```

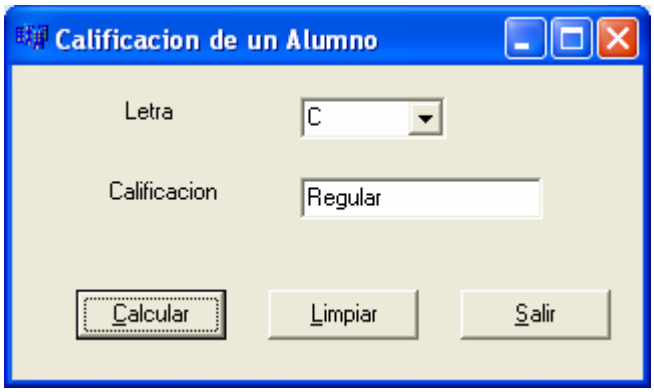
2) Ingresar un numero entero, y si este termina en 2,5 u 8 reportar el cuadrado del numero, si este termina en 4,7 o 9 reportar el numero multiplicado por 5 y reportar el mismo numero en otro caso.



```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "Unit1.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Button1Click(TObject *Sender)
{
  int N;
  N=EdNum->Text.ToInt();
  switch(N%10)
  {
    case 2: case 5: case 8:
       EdResultado->Text=N*N;
       break;
    case 4: case 7: case 9:
       EdResultado->Text=N*5;
       break;
    default:
       EdResultado->Text=N;
       break;
  }
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Button2Click(TObject *Sender)
{
   EdNum->Clear();
   EdResultado->Clear();
   EdNum->SetFocus();

}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button3Click(TObject *Sender)
{
   Close();
}
//---------------------------------------------------------------------------
```

3) Ingresar una letra entre A y E y reportar la calificación de acuerdo a lo siguiente:

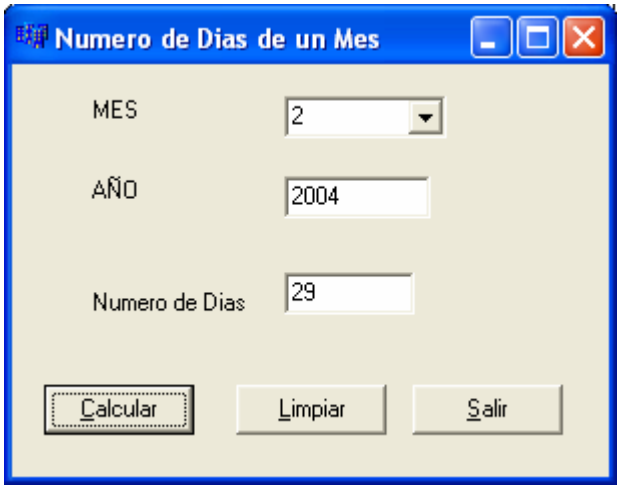
A excelente
B bueno
C regular
D malo
E pesimo



En este ejercicio utilzamos el ComboBox para mostrar las letras. Usamos la propiedad Text de este Componente que es la que devuelve el elemento Seleccionado

```
void __fastcall TForm1::btnCalcularClick(TObject *Sender)
{
  char letra;

 // El String es una cadena de caracteres cuyo indice comienza en uno
  letra=cboLetra->Text[1];
  switch(letra)
  {
    case 'A' :
      edCal->Text="Excelente";
      break;
    case 'B' :
      edCal->Text="Bueno";
      break;
    case 'C' :
      edCal->Text="Regular";
      break;
    case 'D' :
      edCal->Text="Malo";
      break;
    case 'E' :
      edCal->Text="Pesimo";
      break;
  }
}
//---------------------------------------------------------------------------
void __fastcall TForm1::btnLimpiarClick(TObject *Sender)
{
  cboLetra->Text="";
  edCal->Clear();
}
//---------------------------------------------------------------------------
void __fastcall TForm1::btnSalirClick(TObject *Sender)
{
 Close();
}
```

5) Programa para ingresar un Numero de Mes y el valor de un año y reporte cuantos días tiene ese mes.




```
void __fastcall TForm1::btnCalcularClick(TObject *Sender)
{
   int m,a;
   m=cboMes->Text.ToInt();
   a=edA->Text.ToInt();
   switch(m)
   {
     case 1: case 3: case 5: case 7:
     case 8: case 10: case 12:
        edNumDias->Text=31;
        break;
     case 4: case 6: case 9: case 11:
        edNumDias->Text=30;
        break;
     case 2:
        if((a%4==0 && a%100!=0) || a%400==0)
          edNumDias->Text=29;
        else
          edNumDias->Text=28;
        break;
   }

}
//---------------------------------------------------------------------------

void __fastcall TForm1::btnLimpiarClick(TObject *Sender)
{
  cboMes->Text="";
  edA->Clear();
  edNumDias->Clear();
  cboMes->SetFocus();

}
//---------------------------------------------------------------------------

void __fastcall TForm1::btnSalirClick(TObject *Sender)
{
 Close();
}
//---------------------------------------------------------------------------
```

Ejercicios Propuestos

1) Una empresa comercial desea hacer un programa para calcular el precio neto de un artículo de acuerdo a lo siguiente:

a) Si la venta es al contado se le da el 40% de descuento.
b) Si la venta es a plazos y:
   T=6 se recarga el 20 %
   T=12 se recarga el 40%
   T=18 se recarga el 60%

Se debe ingresar el precio del artículo, el codigo de venta (c) contado, (p) plazos y si la venta es a plazos se debe ingresar el tiempo de pago.

Componente ListBox: 

Es un componente que permite visualizar una lista de elementos. Además se puede seleccionar uno o varios de ellos.
La manera de colocar elementos en una lista se puede hacer de dos maneras:

1) Mediante codigo usando el metodo Add de la propiedad Ítems
2) En Tiempo de diseño, del Object Inspector escoger la propiedad Ítems y nos muestra una Lista donde podemos colocar los elementos.

Propiedades:

Items: contiene los elementos del ListBox; es del tipo TStrings (lista de strings).
Esta clase, a su vez, contiene métodos que permiten manipular elementos como agregar (Add), insertar (insert), eliminar (delete), numero de elementos de la lista (Count)
Los índices de los elementos de una lista comienzan en 0.
Ejemplo:
Si la lista se llama LstNombres

```
LstNombres->Ítems->Add("Luis");  Agrega el nombre luis a la Lista LstNombres
LstNombres->Ítems->delete(3);        elimina el elemento cuyo indice es 3.
LstNombres->Ítems->Count           ; devuelve el numero elementos de la lista.
LstNombres->Ítems->Strings[i]     devuelve el elemento de indice i.
```

ItemIndex: Devuelve o estable el índice del Elemento seleccionado. Si el elemento no esta seleccionado devuelve -1.
Columns: especifica el número de columnas.
MultiSelect: Si se le coloca en true permite seleccionar varios elementos al mismo tiempo. Por defecto esta en false.
SelCount: Indica el numero de elementos seleccionados cuando la Selección Multiple esta permitida.
Selected: Indica si un elemento particular esta seleccionado.
Por ejemplo si se desea sumar todos los elementos seleccionados de una lista

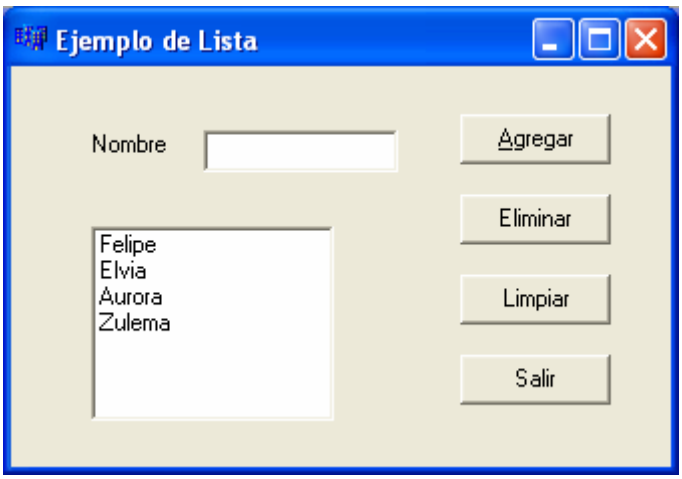
```
int s=0;
 for(int i=0;i<ListBox1->Ítems->Count;i++)
{
    if(ListBox1->Selected[i])
        s=s+ListBox1->Items->Strings[i].toInt();
 }
```

Sorted: ordena automáticamente los elementos alfabéticamente.
Métodos:
Clear: Borra todos los elementos del ListBox.

Eventos

OnClick : Este evento ocurre cuando el usuario selecciona un elemento.
Cuando escogemos un elemento y queremos efectuar algo colocamos codigo en el evento OnClick de la lista.

Ejercicio.
El siguiente programa para manipula una lista de nombres, permite agregar un nombre a una lista, y además eliminar un nombre si este esta seleccinado




```
void __fastcall TForm1::btnAgregarClick(TObject *Sender)
{
 if (edNombre->Text!="")
 {
   lstNombres->Items->Add(edNombre->Text);
   edNombre->Clear();
   edNombre->SetFocus();
 }
 else
     ShowMessage("No se ha ingresado nombre");
}
//---------------------------------------------------------------------------

void __fastcall TForm1::btnEliminarClick(TObject *Sender)
{
 int i;
 i=lstNombres->ItemIndex;
 if(i!=-1)
   lstNombres->Items->Delete(i);
 else
     ShowMessage("No se ha seleccionado elemento");

}
//---------------------------------------------------------------------------
void __fastcall TForm1::btnLimpiarClick(TObject *Sender)
{
  edNombre->Clear();
  lstNombres->Clear();
  edNombre->SetFocus();
}
//---------------------------------------------------------------------------

void __fastcall TForm1::btnSalirClick(TObject *Sender)
{
 Close();
}
//---------------------------------------------------------------------------
```

PROCESOS REPETITIVOS

1) while (mientras)

```
while(condición)
{
    Instrucciones;
}
```

En este proceso se verifica la *condición*, si esta es verdadera se ejecutan *Instrucciones* y automáticamente se vuelve de nuevo a verificar la condición. Este proceso se ejecuta hasta que la condición llegue a ser falsa.
A este proceso se le conoce como de Entrada controlada, pues primero se verifica la condición y luego se ejecutan las instrucciones

2) do... while (Hacer ... mientras)

```
do{

    Instrucciones;

}while(condición);
```

En este proceso primero se realizan las *instrucciones* y luego se verifica la *condición*, si esta es verdadera, se realizan de nuevo las *Instrucciones*. Este proceso se ejecuta hasta que la *condición* llegue a ser falsa.
A este proceso se le conoce como de Salida controlada pues la condición se encuentra al final.

3) **for**

```
for(exp1; exp2; exp3)
{
  Instrucciones;
}
```

Donde :

exp1 : Instrucciones de Inicialización. Se ejecuta 1sola vez y se va a exp2.
exp2 : Condición que controla el proceso Repetitivo. Si esta expresión es verdadera se ejecutan las Instrucciones y se va a exp3. Si esta expresión es falsa el proceso termina.
exp3 : Instrucciones de incremento de variables. Se ejecutan y luego se va a exp2.

Las sentencias break y continue

Los sentencias break y continue alteran el flujo de control.

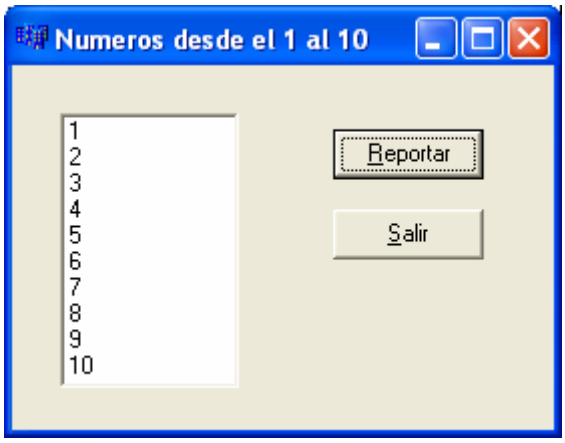
Sentencia break

La sentencia break, cuando se ejecuta en una estructura while, for, do..while o switch, causa la salida inmediata de la estructura. La ejecución continua con el siguiente instrucción después de la estructura. Un uso común de la sentencia break es terminar antes de tiempo de un ciclo (for, while, do..while) o saltarse el resto de una estructura switch

Setencia continue

La sentencia continue, cuando se ejecuta en una estructura while, for o do...while, se salta el resto de las instrucciones del cuerpo de la estructura y continua con la siguiente iteración del ciclo.
En las estructuras while, do...while, la prueba para continuar el ciclo se evalúa inmediatamente después de ejecutarse la sentencia continue. En la estructura for, se ejecuta la expresión de incremento y luego se ejecuta la prueba para ejecutar el ciclo.

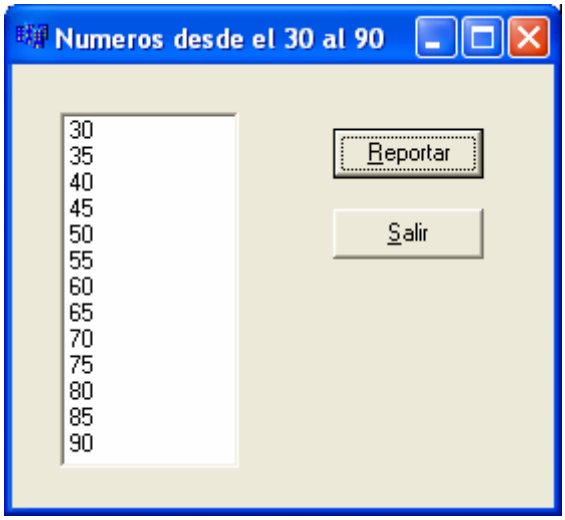
1) Programa para reportar los números desde el 1 al 10




```
void __fastcall TForm1::btnReportarClick(TObject *Sender)
{
  int i;
  lstNum->Items->Clear();
  for(i=1;i<=10;i++)
  {
    lstNum->Items->Add(i);
  }
}
//---------------------------------------------------------------------------
```

2) Reportar los numeros enteros:

30,35,40,45,50,55,.....,85,90



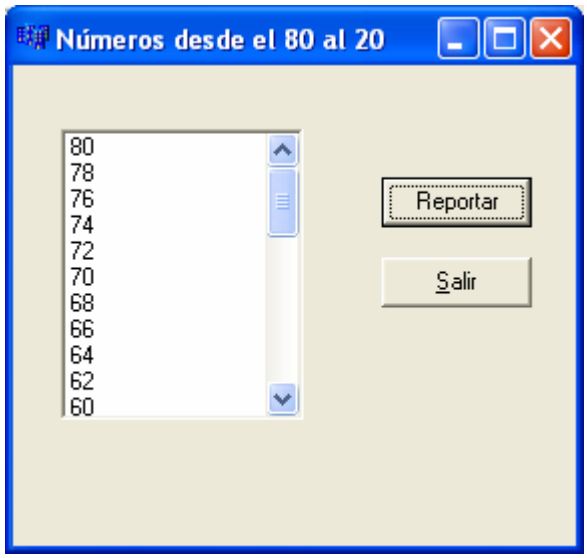
Para hacer esto solo modificamos el Codigo de btnCalcular del ejercicio anterior de la siguiente manera:

```
void __fastcall TForm1::btnReportarClick(TObject *Sender)
{
  int i;
  lstNum->Items->Clear();
  for(i=30; i<=90; i=i+5 )
  {
    lstNum->Items->Add(i);
  }
}
```

3) Reportar los numeros de la siguiente manera;
80, 78, 76, 74,.....,22, 20

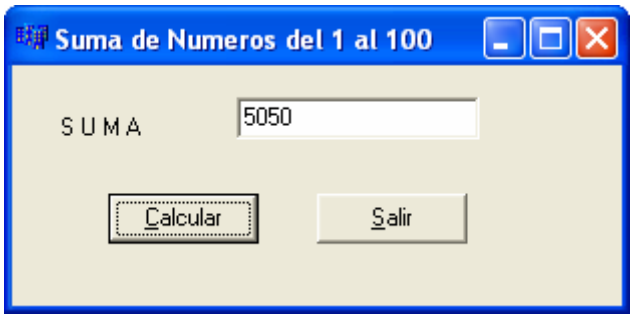
Aca los numeros van en orden descendente de dos en dos, el codigo sería:

```
void __fastcall TForm1::btnReportarClick(TObject *Sender)
{
  int i;
  lstNum->Items->Clear();
  for(i=80; i>=20; i=i-2)
  {
   lstNum->Items->Add(i);
  }

}
```



4) Calcular la siguiente suma:

S = 1+ 2 + 3 + 4 + 5 +....+ 99 + 100




```
void __fastcall TForm1::btnCalcularClick(TObject *Sender)
{
  int i,s=0;
  for(i=1;i<=100;i++)
  {
     s = s + i;
  }
  edSuma->Text = s;
}
```

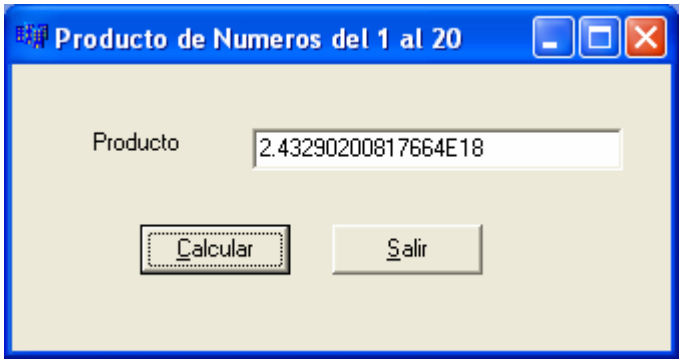
Vemos que para generar los números desde el 1 al 100 usamos la variable i como en los ejercicios anteriores, Para sumar los números utilizamos la variable S inicializada en Cero y por cada numero i generado lo sumamos de la siguiente manera: S = S + i

Acumulador: Variable que comienza con un valor inicial que generalmente es cero y se incrementa una cantidad variable.
Un acumulador tiene el siguiente formato:

S = S + Valor a Sumar;

5) Calcular el siguiente producto

P = 1 * 2 * 3 * 4 * ..... * 20




```
void __fastcall TForm1::btnCalcularClick(TObject *Sender)
{
  int i;
  double P=1;
  for(i=1;i<=20;i++)
  {
    P = P * i;
  }
  edProducto->Text=P;}
//---------------------------------------------------------------------------
```

Como en el ejemplo anterior la variable i se utiliza para generar los números desde el 1 al 100, para calcular el producto se utiliza una variable P que se inicializa en Uno y por cada numero generado lo multiplicamos de la siguiente manera: P = P * i

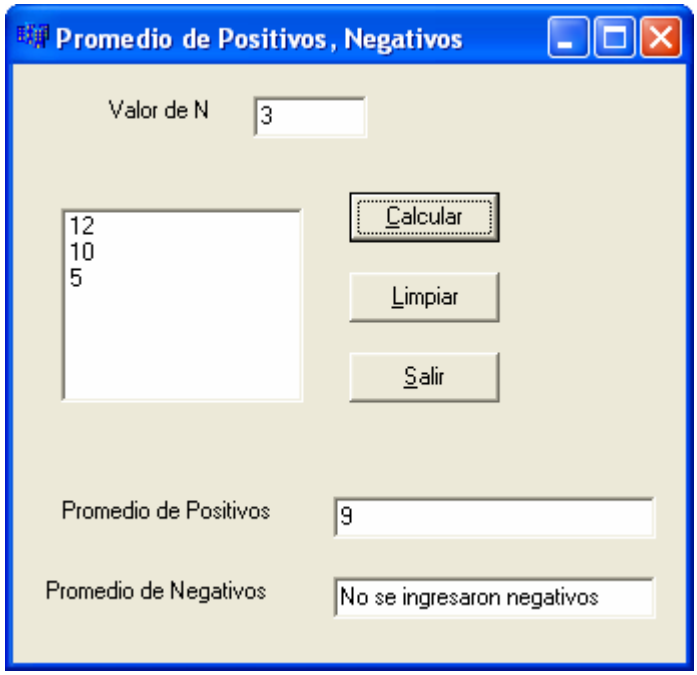
Multiplicador: Variable que comienza con un valor inicial que generalmente es 1. Tiene el siguiente formato:

P = P * Valor a Multiplicar;

6) Ingresar n números y calcular el promedio de ellos.


```
void __fastcall TForm1::btnCalcularClick(TObject *Sender)
{
  int N, i;
  float x,s = 0, p;
  N = edN->Text.ToInt();
  lstNum->Items->Clear();
  for(i=1;i<=N;i++)
  {
    x = InputBox("Ingreso","Ingrese numero : ","").ToDouble();
    lstNum->Items->Add(x);
    s = s+ x;
  }
  p = s / N;
  edPromedio->Text = p;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::btnLimpiarClick(TObject *Sender)
{
  edN->Clear();
  lstNum->Items->Clear();
  edPromedio->Clear();
  edN->SetFocus();
}

//---------------------------------------------------------------------------
void __fastcall TForm1::btnSalirClick(TObject *Sender)
{
   Close();
}
//---------------------------------------------------------------------------
```

7) Ingresar n números y calcular el promedio de los positivos y el promedio de los negativos.




```
void __fastcall TForm1::btnCalcularClick(TObject *Sender)
{
  int n,i,cp=0,cn=0;
  float x,sp=0,sn=0,pp,pn;
  n=edN->Text.ToInt();
  for(i=1;i<=n;i++)
  {
    x=InputBox("Ingreso","Ingrese Numero : ","").ToInt();
    lstNum->Items->Add(x);
    if(x>0)
    {
     sp=sp+x;
     cp=cp+1;
    }
    else
       if(x<0)
       {
         sn=sn+x;
         cn=cn+1;
       }
  }
  if(cp>0)
  {
   pp=sp/cp;
   edPP->Text=pp;
  }
  else
     edPP->Text="No se ingresaron positivos";
  if(cn>0)
  {
   pn=sn/cn;
   edPN->Text=pn;
  }
  else
    edPN->Text="No se ingresaron negativos";
```

FUNCIONES

Los Funciones en c++ permiten modularizar sus programas. Todas las variables declaradas en las definiciones de Funciones son variables locales es decir solo se conocen en la función que se definen.

Casi todos las Funciones tienen una lista de parámetros que permiten comunicar información entre funciones. Los parámetros de una función también son variables locales.

La sintaxis para declarar un función es la siguiente :

```
tipo_de_valor_devuelto nombre_de_la_función ( lista_de_parámetros)
{
  Cuerpo de la funcion
}
```

Donde :

tipo_de_valor_devuelto : Indica el tipo de valor que se devuelve a la función invocadora. Si una función no devuelve un valor, el tipo_de_valor_devuelto se declara como void.

nombre_del_función : Es cualquier identificador válido

lista_de_parámetros : Es una lista separada por comas que contiene las declaraciones de las variables que se pasarán a la función. Si una función no recibe parámetros las lista_de_parámetros se deja vacía.

El cuerpo de la función : es el conjunto de declaraciones e instrucciones que constituyen la función.

Cuando un programa encuentra una llamada a una función, se transfiere el control desde el punto de invocación a la función invocada, se ejecuta la función y el control regresa a la función invocadora.

Una función invocada puede devolver el control a la invocadora de tres maneras.

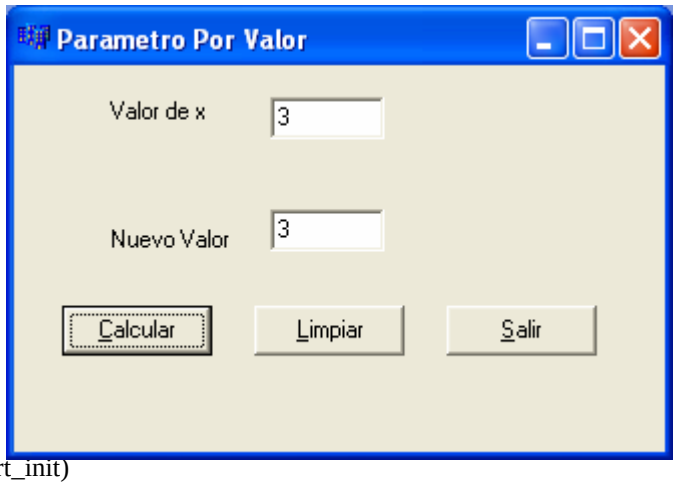
Si la función no devuelve valor, el control se devuelve al llegar a la llave derecha que termina la función o ejecutando la sentencia return;

Si la función devuelve un valor, la sentencia: return expresión

Devuelve el valor de expresión.

Tipos de Parámetros de las Funciones

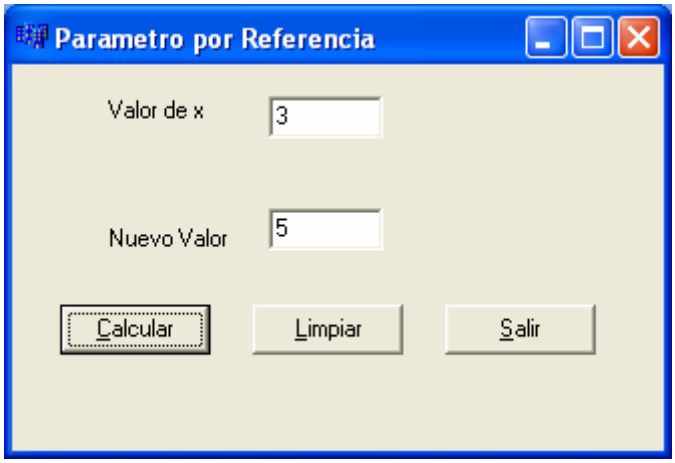
1) **Parámetros por Valor**: Son aquellos a través de los cuales se pasan valores a la función, es decir se hace una copia de la variable pasada como argumento. A estos parámetros se les conoce como parámetros de entrada.




```
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void calculo(float x)
{
  x=x+2;
}

void __fastcall TForm1::btnCalcularClick(TObject *Sender)
{
   float x;
   x=EdX->Text.ToDouble();
   calculo(x);
   EdV->Text=x;
}
//---------------------------------------------------------------------------

void __fastcall TForm1::btnLimpiarClick(TObject *Sender)
{
  EdX->Clear();
  EdV->Clear();
  EdX->SetFocus();
}

void __fastcall TForm1::btnSalirClick(TObject *Sender)
{
  Close();
}
```

La variable x del programa principal al llamar a la función calculo hace una copia de su valor al parámetro de la función, luego la que se incrementa es la variable x de la función y no la variable x del programa principal.

2) **Parámetros por referencia:** Son aquellos a través de los cuales se pasan referencias de las variables esto permite que las variables pasadas como argumento se puedan modificar.
Para declarar un parámetro por referencia se utiliza el operador referencia &.

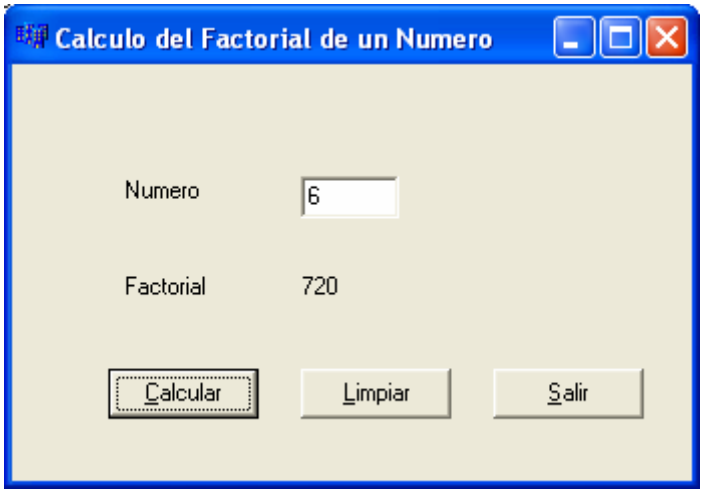



```
//---------------------------------------------------------------------------

void calculo(float &x)
{
  x=x+2;
}

void __fastcall TForm1::btnCalcularClick(TObject *Sender)
{
   float x;
   x=EdX->Text.ToDouble();
   calculo(x);
   EdV->Text=x;
}
//---------------------------------------------------------------------------



void __fastcall TForm1::btnLimpiarClick(TObject *Sender)
{
  EdX->Clear();
  EdV->Clear();
  EdX->SetFocus();
}
//---------------------------------------------------------------------------

void __fastcall TForm1::btnSalirClick(TObject *Sender)
{
  Close();
}

//---------------------------------------------------------------------------
```

La variable x del programa principal al llamar a la función calculo pasa la referencia de la variable esto hace que la variable x se pueda modificar.

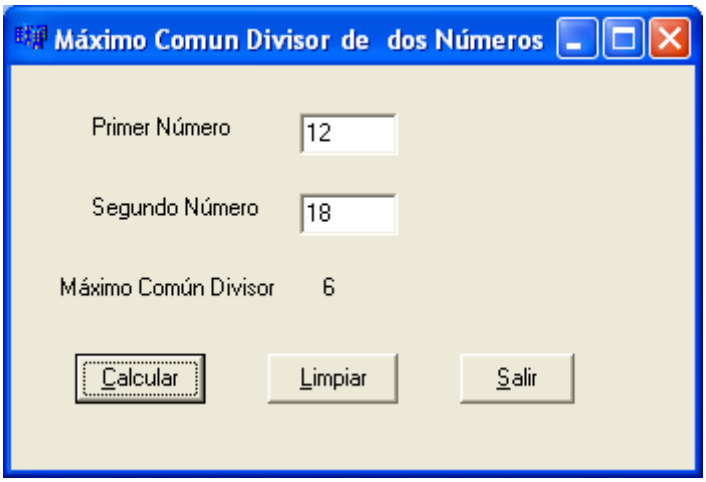
Ejercicios

1) Hacer un programa para calcular el factorial de un número entero n.



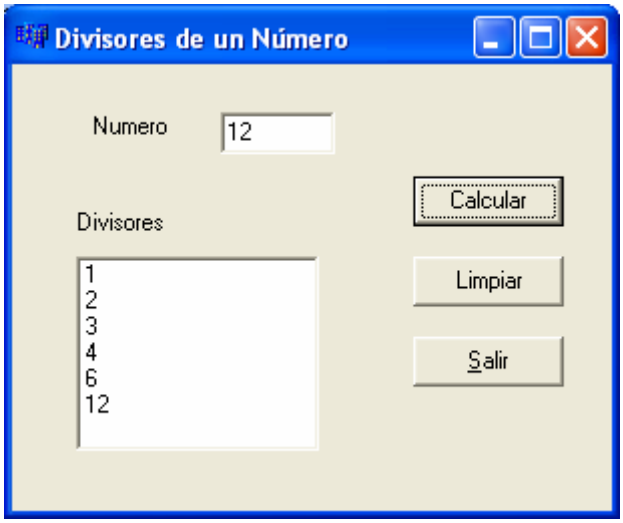

```
#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"
#pragma package(smart_init)
#pragma resource "*.dfm"

TForm1 *Form1;
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}

int  factorial(int n)
{
  int f=1,i;
  for(i=1;i<=n;i++)
    f=f*i;
  return f;
}

void __fastcall TForm1::btnCalcularClick(TObject *Sender)
{
  int n;
  n=edN->Text.ToInt();
  lblF->Caption=factorial(n);

}


void __fastcall TForm1::btnLimpiarClick(TObject *Sender)
{
   edN->Clear();
   lblF->Caption="";
   edN->SetFocus();
}

void __fastcall TForm1::btnSalirClick(TObject *Sender)
{
  Close();
}
```

2) Calcular el MCD de dos números enteros.




```
#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"

#pragma package(smart_init)
#pragma resource "*.dfm"

TForm1 *Form1;

__fastcall TForm1::TForm1(TComponent*
Owner)
        : TForm(Owner)
{
}

int mcd(int x, int y)
{
  int d=2,p=1;
  while(d<=x && d<=y)
  {
     if(x % d == 0 && y % d ==0)
       {
         p=p*d;
         x=x/d;
         y=y/d;
       }
     else
         d++;
  }
  return p;
}

void __fastcall
TForm1::btnCalcularClick(TObject *Sender)
{
   int n1,n2;
   n1=edN1->Text.ToInt();
   n2=edN2->Text.ToInt();
   lblMcd->Caption=mcd(n1,n2);

}

void __fastcall
TForm1::btnLimpiarClick(TObject *Sender)
{
   edN1->Clear();
   edN2->Clear();
   lblMcd->Caption="";
   edN1->SetFocus();
}

void __fastcall TForm1::btnSalirClick(TObject
*Sender)
{
   Close();
}
```

3) Programa para reportar todos los divisores de un número entero N



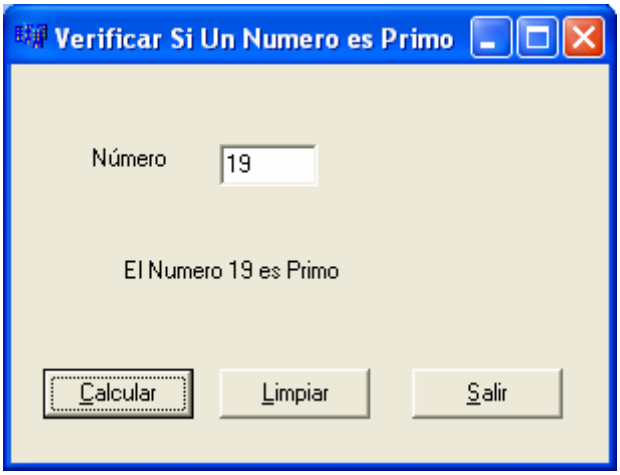

```
#include <vcl.h>
#pragma hdrstop

#include "Unit1.h"
#pragma package(smart_init)
#pragma resource "*.dfm"

TForm1 *Form1;
__fastcall TForm1::TForm1(TComponent* Owner) : TForm(Owner)
{
}

void reporteDivisores(int n,TListBox *lst)
{
  int i;
  lst->Items->Clear();
  for(i=1;i<=n;i++)
  {
    if(n % i ==0)
      lst->Items->Add(i);
  }
}

void __fastcall TForm1::btnCalcularClick(TObject *Sender)
{
   int n;
   n=edN->Text.ToInt();
   reporteDivisores(n,lstDivisores);
}

void __fastcall TForm1::btnLimpiarClick(TObject *Sender)
{
  edN->Clear();
  lstDivisores->Items->Clear();
  edN->SetFocus();
}

void __fastcall TForm1::btnSalirClick(TObject *Sender)
{
  Close();
}
```

4) Programa para reportar los factores primos de un número entero n


```
#include <vcl.h>
#pragma hdrstop

#include "Unit1.h"
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;

__fastcall TForm1::TForm1(TComponent*
Owner)
        : TForm(Owner)
{
}

void factoresPrimos(int n,TListBox *lst)
{
  int d=2;
  lst->Items->Clear();
  while(n>1)
  {
    if(n % d ==0)
      {
       lst->Items->Add(d);
       n=n/d;
      }
    else
       d++;
  }
}

void __fastcall
TForm1::btnCalcularClick(TObject *Sender)
{
 int n;
 n=edN->Text.ToInt();
 factoresPrimos(n,lstFactores);
}
```

```
void __fastcall
TForm1::btnLimpiarClick(TObject *Sender)
{
  edN->Clear();
  lstFactores->Clear();
  edN->SetFocus();
}
void __fastcall TForm1::btnSalirClick(TObject
*Sender)
{
   Close();
}
```

5) Programa para ingresar un numero y se reporte si es primo

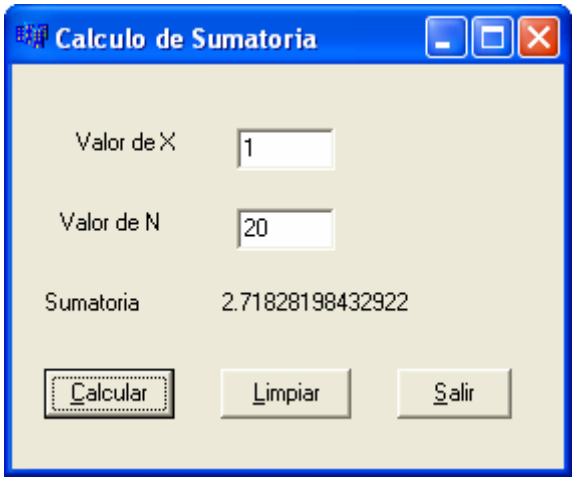



```
#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}

bool esPrimo(int n)
{
  int d=1;
  do{
    d=d+1;
  }while( n%d!=0 && d*d<=n);
  if(d*d>n) return true;
  else return false;
}

void __fastcall TForm1::btnCalcularClick(TObject *Sender)
{
  int n;
  n=edN->Text.ToInt();
  if(esPrimo(n))
    lblMensaje->Caption="El Numero "+IntToStr(n) +" es Primo";
  else
    lblMensaje->Caption="El Número "+IntToStr(n)+" No es Primo";
}

void __fastcall TForm1::btnLimpiarClick(TObject *Sender)
{
 edN->Clear();
 lblMensaje->Caption="";
 edN->SetFocus();
}

void __fastcall TForm1::btnSalirClick(TObject *Sender)
{
  Close();
}
```

6) Calcular el valor de la siguiente sumatoria:

s = 1+ x + x2/2! + x3/3! + x4/4! + ... + xn/n!

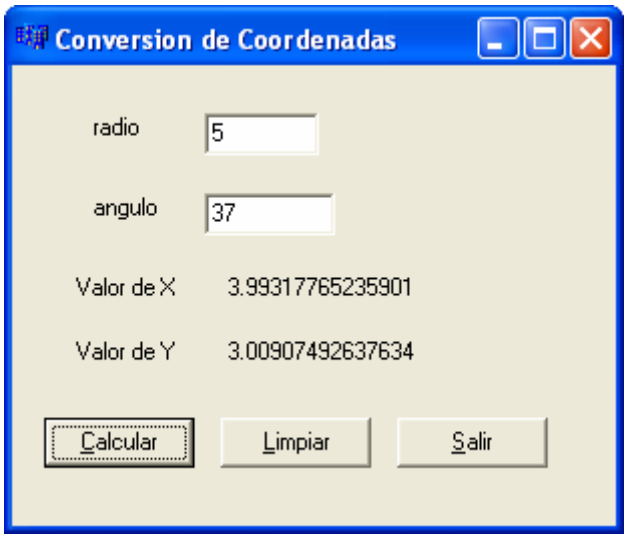
Se debe Ingresar el valor de x, y el valor de n>0.




```
#include <vcl.h>
#pragma hdrstop

#include "Unit1.h"

#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;

__fastcall TForm1::TForm1(TComponent*
Owner)
        : TForm(Owner)
{
}

float suma(float x,int n)
{
  float s=1,f=1,i,p=1;
  for(i=1;i<=n;i++)
  {
    f=f*i;
    p=p*x;
    s=s+p/f;
  }
  return s;
}

void __fastcall
TForm1::btnCalcularClick(TObject *Sender)
{
   float x;
   int n;
   x=edX->Text.ToDouble();
   n=edN->Text.ToInt();
   lblS->Caption=suma(x,n);
}
```

```
void __fastcall
TForm1::btnLimpiarClick(TObject *Sender)
{
   edX->Clear();
   edN->Clear();
   lblS->Caption="";
   edX->SetFocus();
}

void __fastcall TForm1::btnSalirClick(TObject
*Sender)
{
   Close();
}
```

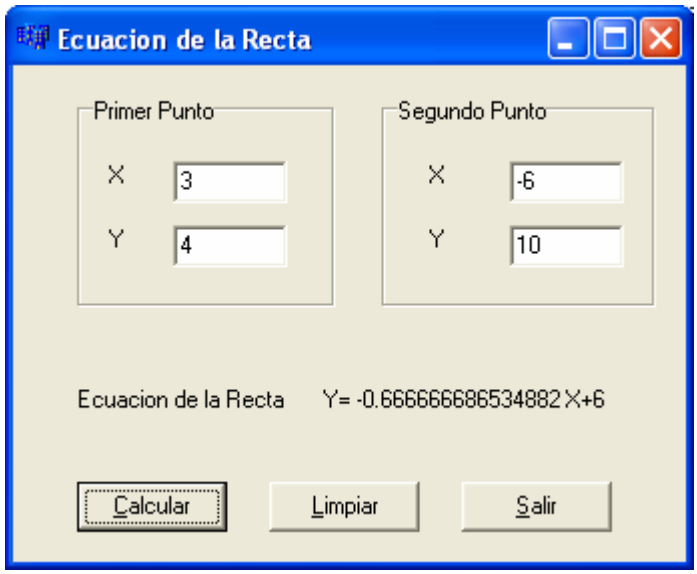
7) Programa para ingresar el valor de un Punto en coordenadas Polares y reporte su equivalente en coordenadas cartesianas.




```
#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"
#include<math.h>

#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;

__fastcall TForm1::TForm1(TComponent*
Owner)   : TForm(Owner)
{
}
void calculo(float r, float t, float &x, float &y)
{
   // convertimos el angulo de sexagesimales a
radianes
   t=t*M_PI/180;
   x=r*cos(t);
   y=r*sin(t);
}

void __fastcall
TForm1::btnCalcularClick(TObject *Sender)
{
  float r,t,x,y;
  r=edR->Text.ToDouble();
  t=edA->Text.ToDouble();
  calculo(r,t,x,y);
  lblX->Caption=x;
  lblY->Caption=y;
}

void __fastcall TForm1::btnSalirClick(TObject
*Sender)
{
   Close();
}
```

```
void __fastcall
TForm1::btnLimpiarClick(TObject *Sender)
{
  edR->Clear();
  edA->Clear();
  lblX->Caption="";
  lblY->Caption="";
  edR->SetFocus();
}
```

3) Ingrese 2 puntos del plano cartesiano y reporte la ecuacion de la recta que los contiene.



```
#include <vcl.h>
#pragma hdrstop

#include "Unit1.h"
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;

__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}

void calculo(float x1, float y1, float x2, float y2, float  &m, float &b)
{
  m=(y2-y1)/(x2-x1);
  b=y1-m*x1;
}

void __fastcall TForm1::btnCalcularClick(TObject *Sender)
{
  float x1,y1,x2,y2,m,b;

  x1=edX1->Text.ToDouble();
  y1=edY1->Text.ToDouble();
  x2=edX2->Text.ToDouble();
  y2=edY2->Text.ToDouble();
  calculo(x1,y1,x2,y2,m,b);
  lblEcuacion->Caption="Y= "+FloatToStr(m)+" X";
  if (b>0)
    lblEcuacion->Caption=lblEcuacion->Caption+"+"+FloatToStr(b);
  else
    lblEcuacion->Caption=lblEcuacion->Caption+" "+FloatToStr(b);
}

void __fastcall TForm1::btnLimpiarClick(TObject *Sender)
{
  edX1->Clear();
  edY1->Clear();
  edX2->Clear();
  edY2->Clear();
  lblEcuacion->Caption="";
  edX1->SetFocus();
}

void __fastcall TForm1::btnSalirClick(TObject *Sender)
{
  Close();
}
```