Estructuras (struct)

Una estructura (struct) es un tipo de dato definido por el usuario. Las estructuras son similares a los registros utilizados en otros lenguajes de programación.
Una estructura es un tipo de dato estructurado que consta de un conjunto de elementos que pueden ser del mismo tipo o de tipos diferentes
Los componentes de una estructura se denominan campos. Cada campo tiene un nombre llamado identificador de campo que es algún identificador elegido por el programador cuando se declara la estructura.

Definición de una estructura

La sintaxis para definir una estructura es la siguiente:

```
Struct nombreEstructura{
        TipodeDato1   campo1;
        TipoDeDato2  campo2:
        TipoDeDato3  campo3;

        TipoDeDaton campon;
};
```

Lo que se esta definiendo es un Nuevo tipo de Dato llamado nombreEstructura que tiene un conjunto de campos.

Ejemplos:
- Declarar una estructura llamada Producto con los siguientes campos : codigo, nombre, precio, stock.

```
struct Producto{
        String codigo;
        String nombre;
        float precio;
        int stock;
};
```

-Declarar la Estructura Empleado con los siguientes campos : Apellidos, nombres, sueldo.

```
struct Empleado{
        String apellidos;
        String nombres;
        float sueldo;
};
```

- Declarar la Estructura Punto con campos x e y.

```
struct Punto{
        float x;
        float y;
};
```

Variables Tipo Estructura
Al declarar la estructura se ha creado un nuevo tipo de dato, pero no se ha declarado variables de ese tipo.
Para declarar variables se puede hacer de la siguiente manera:

Producto prod1,prod2, prod3;

Se ha declarado 3 variables tipo producto.

También se puede declarar la estructura y automáticamente declarar variables.

```
struct Producto{
        String codigo;
        String nombre;
        float precio;
        int stock;
} prod1, prod2, prod3;
```

Acceso a los campos de una Estructura
El acceso a los miembros de una estructura se realiza mediante el operador punto (.)

Por ejemplo si declaramos la variable X de tipo producto

Producto X;

Para acceder a sus campos se hace de la siguiente manera

```
X.codigo = ”pr001”;
X.nombre = ”ace”;
X.precio = 1.5;
X.stock = 200;
```

Operaciones sobre Estructuras
Una operación que se puede realizar entre estructuras es la asignación (copia del contenido de una estructura en otra estructura del mismo tipo). Si A y D son variables tipo estructura del mismo tipo, la Sentencia

A=D;

Copia todo los valores asociados de la variable D a la variable A.
Ejemplo:

```
Punto  P1, P2;
P1.x=10;
P1.y=20;
```

La sentencia de asignación

P2=P1

Equivale a :

```
P2.x = P1.x;
P2.y=P1.y;
```

No se puede comparar completamente una variable tipo estructura con otra.

Por ejemplo si se tiene dos variables A y B de tipo estructura

Punto A, B;

No se puede hacer esto: if(A==B)

Si se quiere comparar se tiene que hacer campo por campo

```
If(A.x==B.x && A.y==B.y)
```

Estructuras Anidadas

Es posible declarar una estructura con campos que sean otras estructuras. Una estructura con uno o mas campos que son tipo estructura se llaman registro jerárquico o anidado.

Ejemplo:

```
struct NombreEmp{
    String apellidos;
    String nombre;
};

struct CalleNumero{
    String calle;
    Int numero;
};

Struct Empleado{
        NombreEmp nombre;
        CalleNumero dir;
};
```

Acceso a los registros Anidados

Para referenciar un campo en estructuras anidadas se debe indicar el camino a seguir en orden jerárquico desde el nombre de la estructura raíz hasta el campo específico.
Ejemplo:

Empleado E;

```
E.nombre.apellidos = “Juan”;
E.nombre.apellidos = “Peréz”;
E.dir.calle = “Zela”;
E.dir.numero= 254
```

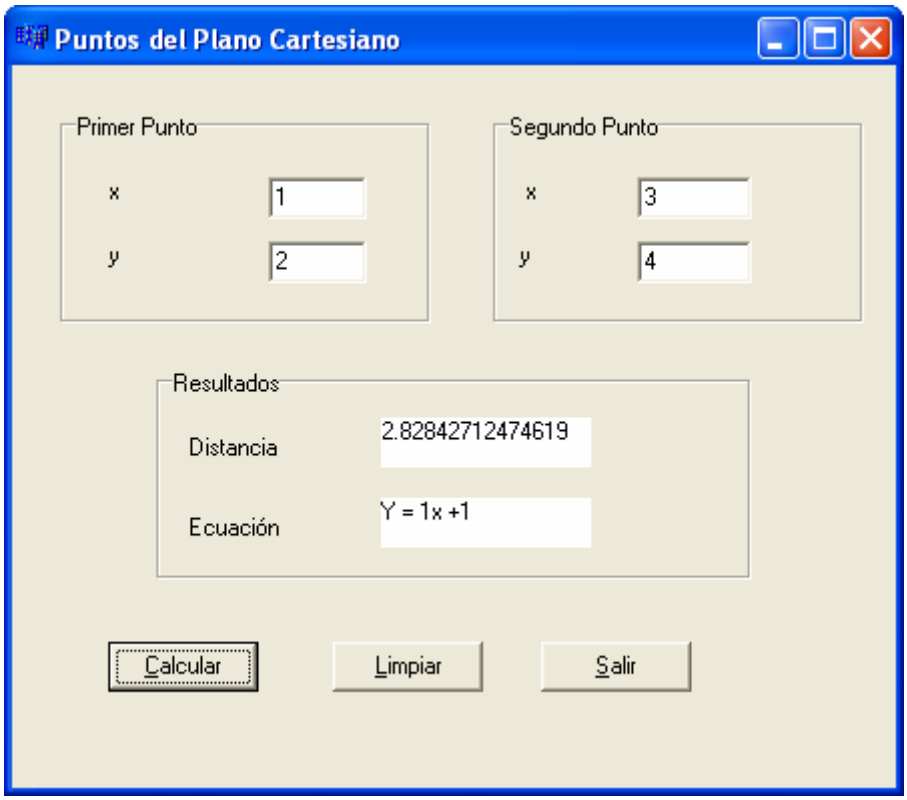
1) Programa para ingresar las coordenadas de 2 puntos del plano cartesiano. Reportar la distancia que hay entre ellos y la ecuación de la recta que pasa por ellos.

En la Unidad UnitPunto.h declaramos la estructura y las funciones para calcular la distancia y los coeficientes de la ecuación.

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

#ifndef UnitPuntoH
#define UnitPuntoH
//---------------------------------------------------------------------------
#endif

struct Punto{
  float x;
  float y;
  };

float distancia(Punto p1, Punto p2);
void calculo(Punto p1, Punto p2, float &m, float &b);
//---------------------------------------------------------------
```

En el archivo UnitPunto.cpp se implementan las funciones distancia y calculo.

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

#pragma hdrstop

#include "UnitPunto.h"
#include<math.h>

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

#pragma package(smart_init)

float distancia(Punto p1, Punto p2)
{
   return sqrt(pow(p1.x-p2.x,2)+pow(p1.y-p2.y,2));
}

void calculo(Punto p1, Punto p2, float &m, float &b)
{
 m=(p2.y-p1.y)/(p2.x-p1.x);
 b=p1.y-m*p1.x;
}
```




```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"
#include "UnitPunto.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm1::btnCalcularClick(TObject *Sender)
{
  Punto p1,p2;
  float m,b;
  p1.x=Edx1->Text.ToDouble();
  p1.y=Edy1->Text.ToDouble();
  p2.x=Edx2->Text.ToDouble();
  p2.y=Edy2->Text.ToDouble();
  lblDistancia->Caption=FloatToStr(distancia(p1,p2));
  calculo(p1,p2,m,b);
  lblEcuacion->Caption="Y = "+FloatToStr(m)+"x +"+FloatToStr(b);
}
```

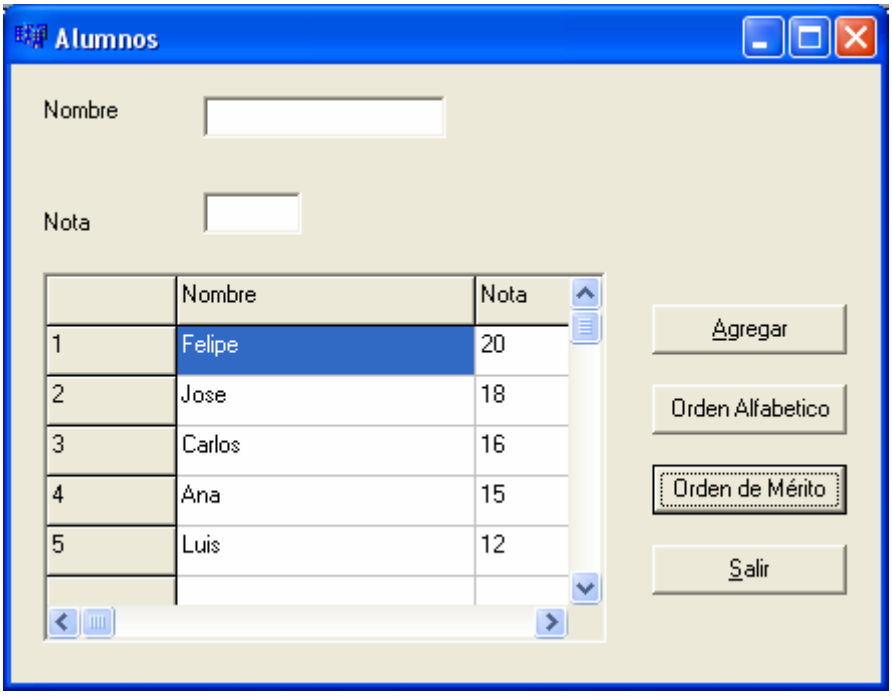
```
//---------------------------------------------------------------------------
void __fastcall TForm1::btnLimpiarClick(TObject *Sender)
{
  Edx1->Clear();
  Edy1->Clear();
  Edx2->Clear();
  Edy2->Clear();
  lblDistancia->Caption="";
  lblEcuacion->Caption="";
  Edx1->SetFocus();
}
//---------------------------------------------------------------------------
void __fastcall TForm1::btnSalirClick(TObject *Sender)
{
  Close();

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

```
2) Hacer un programa para ingresar los nombres y las notas de los
alumnos de Lenguaje de Programación y se reporte:
a) Una Lista en orden Alfabético
b) Una Lista en orden de Merito.

En la Unidad VectorAlumnos.h declaramos la estructura y las funciones
para ordenAlfabetico y ordenMerito

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

#ifndef UnitVectorAlumnosH
#define UnitVectorAlumnosH
//---------------------------------------------------------------------
-------
#endif
#include<vcl.h>

#define MAX 100

struct Alumno{
     String nombre;
     float nota;
};


void ordenAlfabetico(Alumno A[], int n);
void ordenMerito(Alumno A[], int n);
```

En el Archivo UnitVectorAlumnos.cpp se implementan las funciones declaradas en el archivo UnitVectorAlumnos.h

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

#pragma hdrstop

#include "UnitVectorAlumnos.h"

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

#pragma package(smart_init)

void ordenAlfabetico(Alumno A[], int n)
{
  int i,j;
  Alumno temp;
  for(i=0;i<n-1;i++)
    for(j=i+1;j<n;j++)
      if(A[i].nombre>A[j].nombre)
       {
            temp=A[i];
            A[i]=A[j];
            A[j]=temp;
       }
}
```

```
void ordenMerito(Alumno A[], int n)
{
  int i,j;
  Alumno temp;
  for(i=0;i<n-1;i++)
    for(j=i+1;j<n;j++)
      if(A[i].nota<A[j].nota)
       {
            temp=A[i];
            A[i]=A[j];
            A[j]=temp;
       }
}
```




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

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

TForm1 *Form1;
Alumno A[MAX];
int n;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner) : TForm(Owner)
{
}
```

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

void mostrar(TStringGrid *Grd)
{
  int i;
  for(i=0;i<n;i++)
  {
    Grd->Cells[1][i+1]=A[i].nombre;
    Grd->Cells[2][i+1]=A[i].nota;
  }
}

void __fastcall TForm1::btnAgregarClick(TObject *Sender)
{
  if(EdNombre->Text!="" && EdNota->Text!="")
  {

    A[n].nombre=EdNombre->Text;
    A[n].nota=EdNota->Text.ToDouble();
    n++;
    GrdAlumnos->RowCount++;
    GrdAlumnos->Cells[0][n]=n;
    GrdAlumnos->Cells[1][n]=A[n-1].nombre;
    GrdAlumnos->Cells[2][n]=A[n-1].nota;
    EdNombre->Clear();
    EdNota->Clear();
    EdNombre->SetFocus();
  }
  else
     ShowMessage("No se ingresaron completamente los datos");

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

void __fastcall TForm1::FormCreate(TObject *Sender)
{
  GrdAlumnos->ColCount=3;
  GrdAlumnos->RowCount=2;
  GrdAlumnos->Cells[1][0]="Nombre";
  GrdAlumnos->Cells[2][0]="Nota";
  n=0;

}

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

```
void __fastcall TForm1::btnOrdenAlfabeticoClick(TObject *Sender)
{
   ordenAlfabetico(A,n);
   mostrar(GrdAlumnos);

}
//---------------------------------------------------------------------------
void __fastcall TForm1::btnOrdenDeMeritoClick(TObject *Sender)
{
   ordenMerito(A,n);
   mostrar(GrdAlumnos);
}
//---------------------------------------------------------------------------
```

3) Dada una colección de puntos (xi,yi), i=1, 2, ...,n, existe una recta L : y = Mx + B para la cual es mínima la suma de los cuadrados de las distancias de los puntos a la recta.
Así, L constituye la recta mas próxima a los puntos dados, y puede ser utilizada para estimar valores aproximados (x,y)
Escríbase un programa que permita hallar la recta de regresión asociada a una colección de datos.

Los coeficientes M y B son dados por las formulas:

$$M = \frac{N * SXY - SX * SY}{N * SXX - SX * SX}$$

$$B = \frac{SXX * SY - SXY * SX}{N * SXX - SX * SX}$$

en donde :

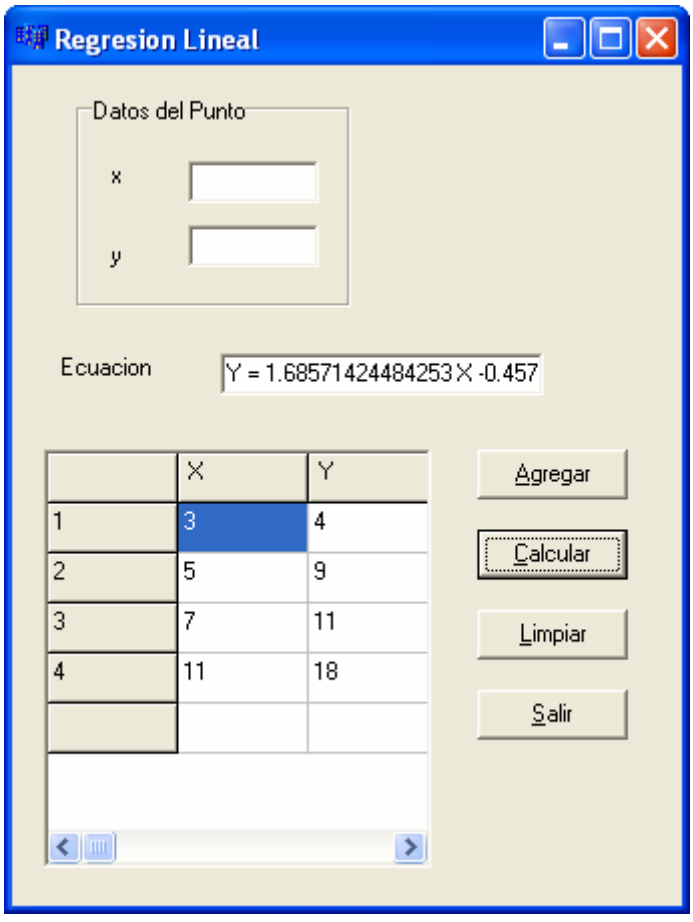
SX = X1 + X2 + ... + Xn
SY = Y1 + Y2 + ... + Yn
SXX = X1^2 + X2^2+ ... + Yn^2
SXY = X1*Y1 + X2*Y2 + ... + Xn*Yn

En la Unidad UnitPunto.h declaramos la estructura y la funcion para calcular la Ecuacion

```
//---------------------------------------------------------------------------
#ifndef UnitPuntoH
#define UnitPuntoH
//---------------------------------------------------------------------------
#endif
#define MAX 100

struct Punto{
    float x;
    float y;
};

void calculoEcuacion(Punto P[], int n, float &M, float &B);
```

En el Archivo UnitPunto.cpp se implementan las funciones declaradas en el archivo UnitPunto.h

```
//---------------------------------------------------------------------------
#pragma hdrstop
#include "UnitPunto.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)

void calculoEcuacion(Punto P[], int n, float &M, float &B)
{
  int i;
  float sx=0,sy=0,sxx=0,sxy=0,den;
  for(i=0;i<n;i++)
  {
    sx=sx+P[i].x;
    sy=sy+P[i].y;
    sxy=sxy+P[i].x*P[i].y;
    sxx=sxx+P[i].x*P[i].x;
  }
  den = n*sxx - sx*sx;
  M = (n*sxy-sx*sy)/den;
  B = (sxx*sy - sxy*sx)/den;
}
```



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

#include <vcl.h>
#pragma hdrstop

#include "Unit1.h"
#include "UnitPunto.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
Punto V[MAX];
int n;
float M, B;

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

void __fastcall TForm1::FormCreate(TObject *Sender)
{
  GrdPuntos->ColCount=3;
  GrdPuntos->RowCount=2;
  GrdPuntos->Cells[1][0]=" X  ";
  GrdPuntos->Cells[2][0]=" Y  ";
  n=0;
}
//---------------------------------------------------------------------------

void __fastcall TForm1::btnAgregarClick(TObject *Sender)
{
  if(Edx->Text!="" && Edy->Text!="")
  {
    V[n].x=Edx->Text.ToDouble();
    V[n].y=Edy->Text.ToDouble();
    GrdPuntos->Cells[0][n+1]=n+1;
    GrdPuntos->Cells[1][n+1]=V[n].x;
    GrdPuntos->Cells[2][n+1]=V[n].y;
    n++;
    GrdPuntos->RowCount++;
    Edx->Clear();
    Edy->Clear();
    Edx->SetFocus();
  }
  else
     ShowMessage("No se ingresaron completamente los datos");
}
```

```
//---------------------------------------------------------------------------
void __fastcall TForm1::btnCalcularClick(TObject *Sender)
{
  String respuesta;
  calculoEcuacion(V,n,M,B);
  respuesta="Y = "+FloatToStr(M)+" X ";
  if(B>0) respuesta=respuesta+"+ ";
  if(B!=0) respuesta=respuesta+FloatToStr(B);
  EdEcuacion->Text=respuesta;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::btnSalirClick(TObject *Sender)
{
   Close();
}
//---------------------------------------------------------------------------
void __fastcall TForm1::btnLimpiarClick(TObject *Sender)
{
   GrdPuntos->ColCount=3;
   GrdPuntos->RowCount=2;
   GrdPuntos->Cells[0][1]="";
   GrdPuntos->Cells[1][1]="";
   GrdPuntos->Cells[2][1]="";
   Edx->Clear();
   Edy->Clear();
   EdEcuacion->Clear();
   n=0;
   Edx->SetFocus();
}
//---------------------------------------------------------------------------
```

Ejercicios Propuestos

1) Ingresar n artículos de los cuales se conoce el código, descripción, cantidad en stock y costo. Además que tenga las siguientes operaciones.
- Dado un código Mostrar los datos de un articulo
- Dado un código eliminar el articulo

2) Ingresar el nombre, nota del Examen1, nota del Examen2 y nota de Trabajos de n alumnos si los pesos de los exámenes son 0.25 y del trabajo 0.5 encontrar el promedio ponderado de cada alumno y reportarlos en orden de Merito, además reportar el Promedio general.

3) Ingresar los datos de n clientes de un banco (código, nombre, saldo) y reportar el cliente (o los clientes) que tienen mayor saldo y el cliente (o los clientes) que tienen mayor saldo.

4) Se tienen los nombres y promedios de n alumnos. Calcular la nota promedio y la desviación estándar de las notas.

5) Un número complejo tiene una parte real y una parte imaginaria. Hacer un programa que permita ingresar 2 números complejos y permita sumar, restar, multiplicar y dividir 2 números complejos.

6) Los términos de un polinomio los podemos colocar como una estructura (coeficiente y exponente). Ingresar el grado y los términos de un polinomio y Escriba una funcion que encuentre el valor numérico del polinomio.

7) Hacer un programa para ingresar los datos de n trabajadores (código, nombre, valor por hora, horas trabajadas). Hacer las siguientes operaciones:
- Mostrar los Trabajadores ordenados por nombre,
- Mostrar los Trabajadores ordenados por sueldo
- Ingresar un codigo de un trabajador y mostrar todos sus datos.

8) Se tiene una serie de fracciones (numerador y denominador).Calcule la suma de todas estas fracciones y muestre el resultado. La respuesta debe estar simplificada.

9) Ingresar n cadenas, y reportarlas ordenadas ascendentemente según la longitud de la cadena.

10) De n alumnos se tiene los siguientes datos: codigo, nombre apellidos, fecha de ingreso. Se pide ordenar por fecha de ingreso.