 # Registros

- Estructuras de datos: permiten al programador definir un tipo al que se asocian diferentes datos que tienen valores lógicamente relacionados bajo un nombre único.

  Clasificación:

  ¬ Elementos: depende de si son del mismo tipo de dato o no.
  ¬ Tamaño: hace referencia a si la estructura puede variar su tamaño.
  ¬ Acceso: hace referencia a cómo se pueden acceder los elementos.
  ¬ Linealidad: hace referencia a cómo se encuentran almacenados los datos.

  Registro: tipo de dato estructurado, permite agrupar diferentes clases de datos en una estructura única.

registro.pas

```
program registros;
const
//
type
    nombre = record
    dato1: tipo;
    dato2: tipo;
    dato3: tipo;
    //
    end;
// Procedimientos y/o funciones
var // Variables del programa principal
 registro: nombre;
begin
    // Programa principal
end;
```

La única operación permitida es la asignación de variables entre sí

- La única forma de acceder a los campos es con . (punto)

accesoRegistros.pas

```
program accesoRegistros;
type
    perros = record
        nombre: string;
        raza: string;
        edad: integer;
    end;
procedure leerInfo(var p: perros);
    begin
        read(p.nombre);
        read(p.raza);
        read(p.edad);
    end;
```

Los registros **pueden modularizarse solo en procesos** (ya que las funciones utilizan datos de tipo simple)
Para leer un dato de un registro que está dentro de otro, hay que especificar de la misma manera que antes, con puntos, pero agregando sus respectivas direcciones.

accRegistrosEnRegistros.pa

```
program accesoRegistrosEnRegistros;
type
    perros = record
        nombre: string;
        raza = record;
            golden: integer;
            cachorro: integer;
            policia: integer;
        edad: integer;
    end;
procedure leerInfo(var p: perros);
    begin
        read(p.nombre);
        read(p.raza.golden);
        read(p.raza.cachorro);
        read(p.raza.policia);
        read(p.edad);
    end;
```

 # Cortes de control

Los cortes de control se utilizan para leer información hasta algún punto. Es necesario tener la información ordenada de alguna forma y volver a leerla al finalizar el bucle.
Por ejemplo:

Se pide realizar un programa que lea perros hasta leer uno de raza `XXX` y al final informe la cantidad de perros de cada raza. La lectura se encuentra ordenada por raza.

corteControl.pas

```
Program corteControl;
Type
    perro = record
    raza: string;
    nombre: string;
    edad: integer;
    end;
// Suponiendo que se dispone del procedimiento de lectura de información "leer"
var
   ani: perro;
   cant: integer;
   actual:string;
begin
    leer(ani); // Leer una vez
    while (ani.raza <> ´XXX´) do
    begin
        while ((ani.raza <> ´XXX´)and(ani.raza = actual)) do
            begin
                cant:= cant + 1;
                leer(ani); // Volver a leer
            end;
        if (cant ≥ max) then
        begin
            max:= cant;
            razaMax:= actual;
        end;
    end;
end.
```