

# AGENDA DE LA CLASE

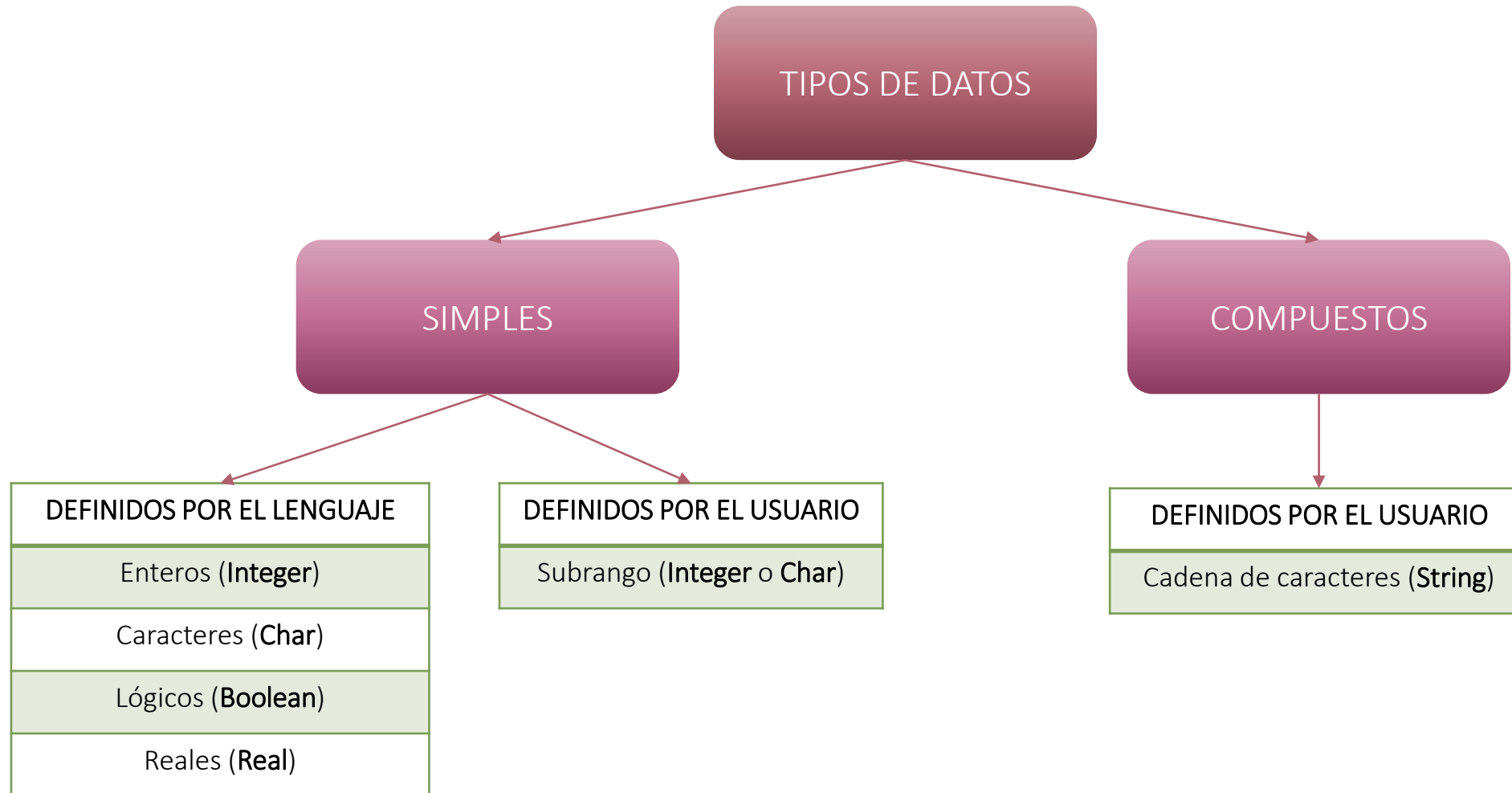
## TIPO DE DATO ESTRUCTURADO / ESTRUCTURA DE DATOS

- CONCEPTO
- CLASIFICACIÓN

## TIPO DE DATO REGISTRO

- MOTIVACIÓN Y CONCEPTO
- DECLARACIÓN Y CARACTERÍSTICAS
- EJERCITACIÓN

# TIPOS DE DATOS VISTOS HASTA EL MOMENTO



# TIPO DE DATO ESTRUCTURADO - MOTIVACIÓN

## Problema:

- Se quiere representar la información de los estudiantes de una Facultad.



## Preguntas:

- Qué datos se necesitan para representar a un estudiante (DNI, Apellido y Nombre, Fecha de nacimiento, etc.).
- Todos esos datos agrupados representan al estudiante.
- Cantidad de estudiantes a representar.
- Qué tipo de datos visto hasta el momento se puede utilizar para representar a los estudiantes o al estudiante.



# ESTRUCTURA DE DATOS - CONCEPTO



- Una **estructura de datos** es un conjunto de variables (que podrían ser de distintos tipo) que poseen una relación lógica o conceptual entre sí y que se puede reconocer como un todo, bajo un nombre único.
- Nos permite representar objetos del mundo real que son más complejos que un número, un carácter o una palabra.

# ESTRUCTURA DE DATOS - CONCEPTO

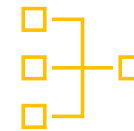


- Ejemplos.

- Representar la lista de empleados de una empresa.
- Representar la lista de productos para un sitio de compras on-line.
- Representar la carta de menú de un bar.



# ESTRUCTURA DE DATOS - CLASIFICACIÓN



Según sus elementos	Según su forma de acceso	Según su ocupación en memoria	Según la relación entre sus elementos
Homogénea	Directo	Estática	Lineal
Heterogénea	Secuencial	Dinámica	No Lineal

# ESTRUCTURA DE DATOS – CLASIFICACIÓN



- Según sus elementos:
  - **Homogénea**: todos los elementos que la componen pertenecen a mismo tipo de dato.
  - **Heterogénea**: los elementos que la componen pueden ser de distinto tipo de dato.



- Según su ocupación en memoria:
  - **Estática**: la cantidad de elementos que puede contener no varía durante el tiempo de ejecución del programa. Por lo tanto, la cantidad de memoria que necesita ocupar es siempre la misma durante todo el programa.
  - **Dinámica**: la cantidad de elementos puede variar, por lo tanto, puede variar la memoria ocupada en tiempo de ejecución.

# ESTRUCTURA DE DATOS – CLASIFICACIÓN



- Según su acceso:
  - **Directo:** se puede acceder a un elemento particular sin necesidad de pasar por otros elementos.
  - **Secuencial:** para llegar a un elemento, puede ser necesario pasar por otros elementos.



- Según su linealidad:
  - **Lineal:** cada elemento puede tener 0 o 1 sucesor, 0 o 1 predecesor o ambos.
  - **No lineal:** cada elemento puede tener 0, 1 o más elementos que le preceden, 0, 1 o más elementos que le suceden o ambos casos.



# REGISTROS

MOTIVACIÓN

CONCEPTO

DECLARACIÓN EN PASCAL




CARACTERÍSTICAS

OPERACIONES Y EJEMPLOS



# TIPO DE DATO REGISTRO - MOTIVACIÓN

## Problemática:

- Representar la información del estudiante. 
- Representar los datos de una llamada telefónica. 
- Representar los datos de un vehículo. 



## Que necesitamos:

- Qué datos nos interesa representar para un estudiante (*DNI, apellido y nombre, fecha de nacimiento, etc.*).
- Qué datos nos interesa de la llamada (*origen, destino, duración, etc.*)
- Qué datos nos interesa para un vehículo (*patente, modelo, marca, etc.*)



# ESTRUCTURA DE DATOS REGISTRO - CONCEPTO



- El tipo de dato registro (***record***) permite agrupar un conjunto de campos, con igual o diferente tipo de dato, bajo un nombre único.

- Ejemplos.

## ***grúa:***

patente.  
marca.  
modelo.  
habilitación.  
km.  
peso.



## ***auto:***

marca.  
modelo.  
velocidad.  
categoría.



## ***empleado:***

DNI.  
apellido.  
nombre.  
estudios.  
CUITL/CUIT.



## ***piloto:***

DNI.  
apellido.  
nombre.  
horas de vuelo.  
CUITL/CUIT.  
fecha de nacimiento.



# TIPO DE DATO REGISTRO – DECLARACIÓN EN PASCAL



**type**

```
identificador = record;  
    campo1: tipo de dato;  
    campo2: tipo de dato;  
    campo3: tipo de dato;  
    ...  
    campoN: tipo de dato;  
end;
```

**var**

```
r1: identificador;  
r2: identificador;
```

## Consideraciones:



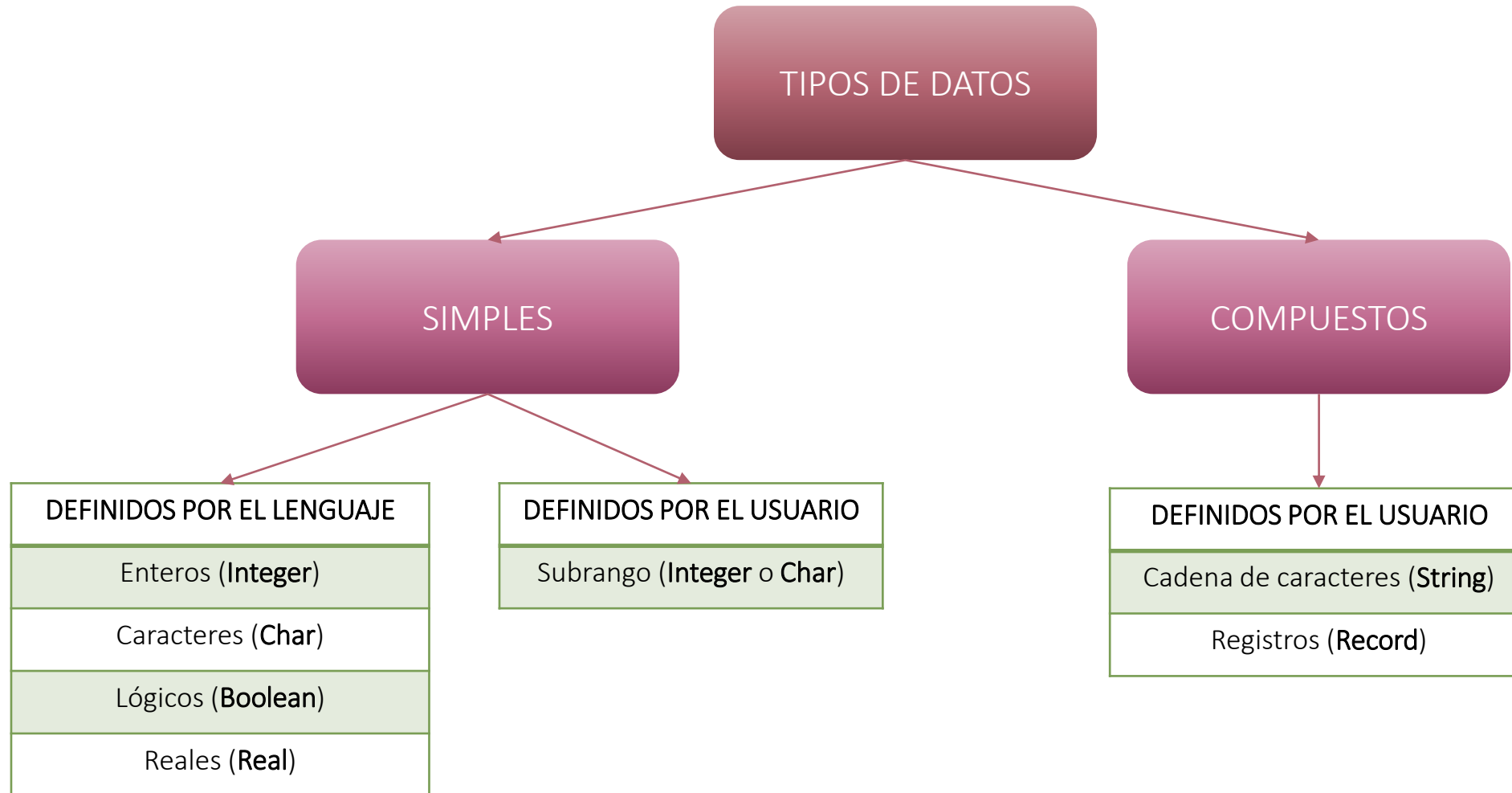
- “Identificador” es el nombre que se elige para el tipo registro (*record*).
- Se debe especificar la lista de campos que componen el registro y el tipo de dato de cada campo. Cada campo puede ser de cualquier tipo de dato conocido.
- Cada campo puede ser referenciado individualmente.
- Variables del tipo “identificador”.

# ESTRUCTURA DE DATOS REGISTRO - CARACTERÍSTICAS



- Características.
  - Según sus elementos es *heterogénea*.
    - Cada campo que contiene el registro puede ser del mismo o de distinto tipo de dato.
  - Según el acceso a sus elementos es de *acceso directo*.
    - Cada campo se puede referenciar directamente sin necesidad de referenciar los otros.
  - Según su ocupación en memoria es *estática*.
    - Al declarar una variable de tipo registro, su tamaño en memoria puede ser determinado realizando la suma de sus campos, algo que no varía en tiempo de ejecución.
  - Según la relación entre sus elementos.
    - En esta estructura de datos podemos decir que no aplica el concepto.
    - No existe una relación de orden entre los campos, solamente el concepto que juntos componen.

# TIPOS DE DATOS VISTOS + REGISTROS



# ESTRUCTURA DE DATOS REGISTRO - OPERACIONES



- Operaciones sobre la variable del tipo registro (*record*).
  - Asignación ( $:=$ )
  - Acceso a cada campo en particular (.)
- Operaciones sobre los campos que componen el registro (*record*).
  - Como cada campo del registro puede tener su propio tipo de dato, entonces las operaciones que se puedan realizar sobre éstos campos son las permitidas para el tipo de dato que posea el campo correspondiente.

# ESTRUCTURA DE DATOS REGISTRO - OPERACIONES



- Única operación permitida sobre la variable del tipo registro (*record*).
  - Asignación ( $:=$ )
- Ejemplo.

**type**

```
estudiante = record;  
  legajo: integer;  
  nombre: string[20];  
  apellido: string[20];  
  fecha_insc: string[10];  
  fecha_nac: string[10];  
end;
```

**var**

```
e1: estudiante;  
e2: estudiante;
```

**{Asignación}**

**begin**

```
...  
e1 := e2;  
...  
...
```

**end.**

*La asignación es válida si  
ambas variables son del  
mismo tipo de registro.*



# ESTRUCTURA DE DATOS REGISTRO - OPERACIONES



- Acceso a los campos de la variable del tipo registro (*record*).
  - A través del símbolo de punto (.)
- A partir de aquí, las operaciones en un registro serán aplicadas a sus campos.
- Ejemplos.

```
type
    estudiante = record;
        legajo: integer;
        nombre: string[20];
        apellido: string[20];
        fecha_insc: string[10];
        fecha_nac: string[10];
    end;

var
    e: estudiante;

    {Asignación de valor a sus campos}

begin
    ...
    e.legajo := 123456;
    ...
    e.fecha_insc := "01/02/2020";
    ...
    read(e.nombre);
    ...
    if (e.fecha_insc > "01/02/2020") then
        ...
    ...
end.
```

*El valor que sea asignado o la operación realizada al campo del registro dependerá de su tipo de dato.*

# ESTRUCTURA DE DATOS REGISTRO – SITUACIONES DE PRÁCTICA



- Completar los datos para de un registro realizando la lectura de sus valores desde el teclado. Los valores son ingresados por el usuario.
  - La lectura de un registro completo, es una tarea que es conveniente modularizar. Es decir, tener un proceso que solamente se concentre es la obtención de estos valores.

**type**

```
estudiante = record;  
  legajo: integer;  
  nombre: string[20];  
  apellido: string[20];  
  fecha_insc: string[10];  
  fecha_nac: string[10];  
end;
```

**var**

```
  e: estudiante;  
begin  
  ...  
  leer_estudiante(e);  
  ...  
end.
```



```
Procedure leer_estudiante (var e: estudiante);  
begin  
  readln(e.legajo);  
  readln(e.apellido);  
  readln(e.nombre);  
  readln(e.fecha_nac);  
  readln(e.fecha_insc);  
end.
```

*La lectura de los campos no necesita respetar el orden en que se encuentran declarados los campos.*

*Todos estos campos aceptan la operación de "read" o "readln".*



**Error:** Un registro (*record*) no admite la operación de *read* ni *readln*.

- read(e)
- readln(e)



# ESTRUCTURA DE DATOS REGISTRO – SITUACIONES DE PRÁCTICA



- Imprimir en pantalla los datos de un registro.
  - La impresión de un registro completo, es una tarea que es conveniente modularizar.

**type**

```
estudiante = record;  
  legajo: integer;  
  nombre: string[20];  
  apellido: string[20];  
  fecha_insc: string[10];  
  fecha_nac: string[10];  
end;
```

**var**

```
  e: estudiante;  
begin  
  ...  
  imprimir(e);  
  ...  
end.
```

```
Procedure imprimir(e: estudiante);  
begin  
  writeln(e.legajo);  
  writeln(e.apellido);  
  writeln(e.nombre);  
  writeln(e.fecha_nac);  
  writeln(e.fecha_insc);  
end.
```



*Todos estos campos aceptan la operación de "write" o "writeln".*



**Error:** Un registro (*record*) no admite la operación de *write* ni *writeln*.

- write(e)
- writeln(e) **×**

# ESTRUCTURA DE DATOS REGISTRO – SITUACIONES DE PRÁCTICA



- ¿Un registro puede contener un campo de otro tipo registro?
  - Si.
  - ¿Cómo se ve afectado el acceso?

**type**

```
fecha = record;  
  día: 1..31; {subrango}  
  mes: 1..12; {subrango}  
  año: 1900..3000; {subrango}  
end;
```

```
estudiante = record;  
  legajo: integer;  
  nombre: string[20];  
  apellido: string[20];  
  fecha_insc: fecha;  
  fecha_nac: fecha;  
end;
```

**var**

```
e: estudiante;
```

**{Acceso y asignación a un campo de tipo registro}**

**begin**

```
...  
e.fecha_insc.día:= 1;  
e.fecha_insc.mes:= 2;  
e.fecha_insc.año:= 2020;
```

**end.**



**Error:** ahora el campo es de tipo registro (*record*) no admite la asignación de un valor directamente.

- e.fecha\_insc := "01/02/2020" ❌

# ESTRUCTURA DE DATOS REGISTRO – SITUACIONES DE PRÁCTICA



- ¿Un registro puede contener un campo de otro tipo registro?
  - Si.
  - ¿Cómo se ven afectadas las operaciones de lectura e impresión en estos campos?

**type**

```
fecha = record;  
  dia: 1..31; {subrango}  
  mes: 1..12; {subrango}  
  año: 1900..3000; {subrango}  
end;
```

```
estudiante = record;  
  legajo: integer;  
  nombre: string[20];  
  apellido: string[20];  
  fecha_insc: fecha;  
  fecha_nac: fecha;  
end;
```

**var**

```
e: estudiante;
```

**Procedure** leer\_estudiante (**var** e: estudiante);

**begin**

```
  readln(e.legajo);  
  readln(e.apellido);  
  readln(e.nombre);  
  readln(e.fecha_nac.dia);  
  readln(e.fecha_nac.mes);  
  readln(e.fecha_nac.año);  
  readln(e.fecha_insc.dia);  
  readln(e.fecha_insc.mes);  
  readln(e.fecha_insc.año);
```

**end.**



*Es correcto, pero se puede mejorar.*

*Modularizando.*



Error: Un registro (**record**) no admite la operación de **read** ni **readln**.

- read(e.fecha\_insc)
- readln(e.fecha\_insc)



# ESTRUCTURA DE DATOS REGISTRO – SITUACIONES DE PRÁCTICA



- ¿Un registro puede contener un campo de otro tipo registro?
  - Si.
  - ¿Cómo se ven afectadas las operaciones de lectura e impresión en estos campos?

**type**

```
fecha = record;  
  dia: 1..31; {subrango}  
  mes: 1..12; {subrango}  
  año: 1900..3000; {subrango}  
end;
```

```
estudiante = record;  
  legajo: integer;  
  nombre: string[20];  
  apellido: string[20];  
  fecha_insc: fecha;  
  fecha_nac: fecha;  
end;
```

**var**

```
e: estudiante;
```

```
Procedure leer_estudiante (var e: estudiante);  
begin  
  readln(e.legajo);  
  readln(e.apellido);  
  readln(e.nombre);  
  leer_fecha(e.fecha_nac);  
  leer_fecha(e.fecha_insc);  
end.
```

```
Procedure leer_fecha (var f: fecha);  
begin  
  readln(f.dia);  
  readln(f.mes);  
  readln(f.año);  
end.
```



El mismo concepto se aplica a la impresión del registro.

# TIPO DE DATO REGISTRO – TRUCO PRÁCTICO - **WITH**



## Motivación:

- Al trabajar con registros que poseen un número considerable de campos, el acceso a cada uno de ellos, por ejemplo, para un proceso de lectura o de impresión en pantalla, se vuelve algo tedioso para el programador.
- Se puede hacer uso de la sentencia **WITH**.
- Se referencia una sola vez a la variable registro y luego se pueden utilizar sus campos sin necesidad de anteponer la referencia.



```
Procedure leer_fecha (var f: fecha);  
begin  
  with f do begin  
    readln(dia);  
    readln(mes);  
    readln(año);  
  end;  
end.
```

# ESTRUCTURA DE DATOS REGISTRO – EJERCITACIÓN



- Se leen artículos que una juguetería posee para la venta on-line. De cada artículo se lee: código, descripción, fecha de fabricación, edad recomendada y precio. Se pide informar la descripción para los artículos cuyo precio sea menor a \$500. La lectura finaliza cuando se lee el código 0 (cero).

*{declaración de tipos}*

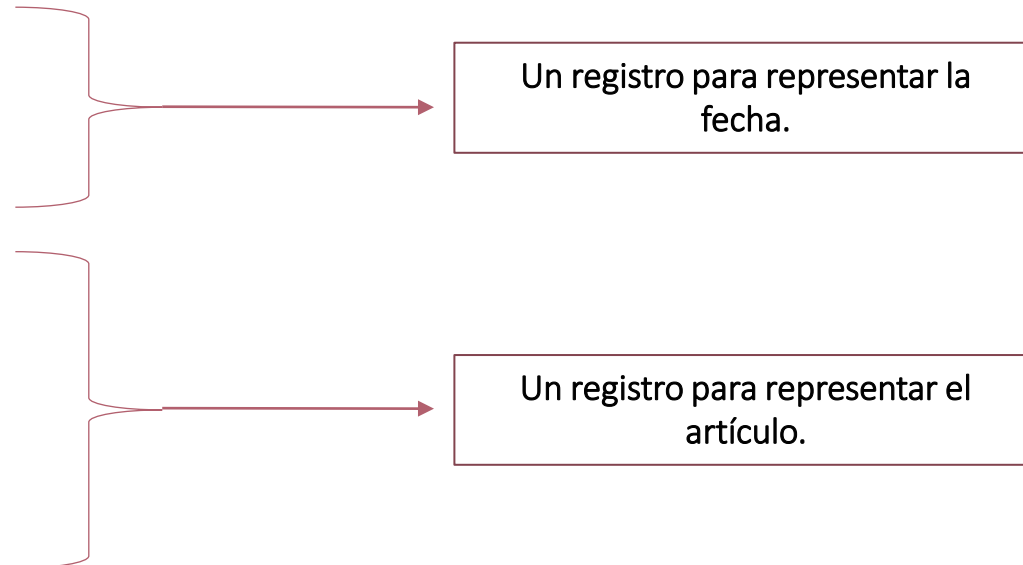
**type**

*{subrangos}*

```
rango_edades = 0..20; rango_dias = 1..31; rango_mes = 1..12;  rango_anios = 1900..3000;  
cadena = string[150];
```

```
fecha = record  
  dia: rango_dias;  
  mes: rango_mes;  
  anio: rango_anios;  
end;
```

```
articulo = record  
  cod: integer;  
  desc: cadena;  
  fecha_fab: fecha;  
  edad: rango_edades;  
  precio: real;  
end;
```





# ESTRUCTURA DE DATOS REGISTRO – EJERCITACIÓN



- Se leen artículos que una juguetería posee para la venta on-line. De cada artículo se lee: código, descripción, fecha de fabricación, edad recomendada y precio. Se pide informar la descripción para los artículos cuyo precio sea menor a \$500. La lectura finaliza cuando se lee el código 0 (cero).

```
{proceso que lee los datos de un artículo}
procedure leer_articulo (var a: articulo);

    procedure leer_fecha (var f: fecha);
    begin ... end;

begin
    with a do begin
        writeln;
        write(' - Codigo: ');
        readln(cod);
        if (cod <> 0) then begin
            write(' - Descripcion: ');
            readln(desc);
            {invocación al proceso que lee la fecha}
            leer_fecha(fecha_fab);
            write(' - Edad: ');
            readln(edad);
            write(' - Precio: ');
            readln(precio);
        end;
    end;
end;
```

{proceso que lee los datos de la fecha}

```
procedure leer_fecha (var f: fecha);
begin
    writeln(' - Fecha de fabricacion.');
    with f do begin
        write(' - Dia: ');
        readln(dia);
        write(' - Mes: ');
        readln(mes);
        write(' - Anio: ');
        readln(anio);
    end;
end;
```

# ESTRUCTURA DE DATOS REGISTRO – EJERCITACIÓN



- Se leen artículos que una juguetería posee para la venta on-line. De cada artículo se lee: código, descripción, fecha de fabricación, edad recomendada y precio. Se pide informar la descripción para los artículos cuyo precio sea menor a \$500. La lectura finaliza cuando se lee el código 0 (cero).

```
{variables del programa principal}
var
  a: articulo;
begin {programa principal}
  {inicializaciones}
  leer_articulo(a);
  while (a.cod <> 0) do begin
    if (a.precio < 500) then begin
      writeln;
      writeln('---- Artículo con precio menor a 500: ', a.desc);
      writeln;
    end;
    leer_articulo(a);
  end;
  writeln;
  writeln('---- Presione enter para finalizar ----');
  readln;
end.
```