

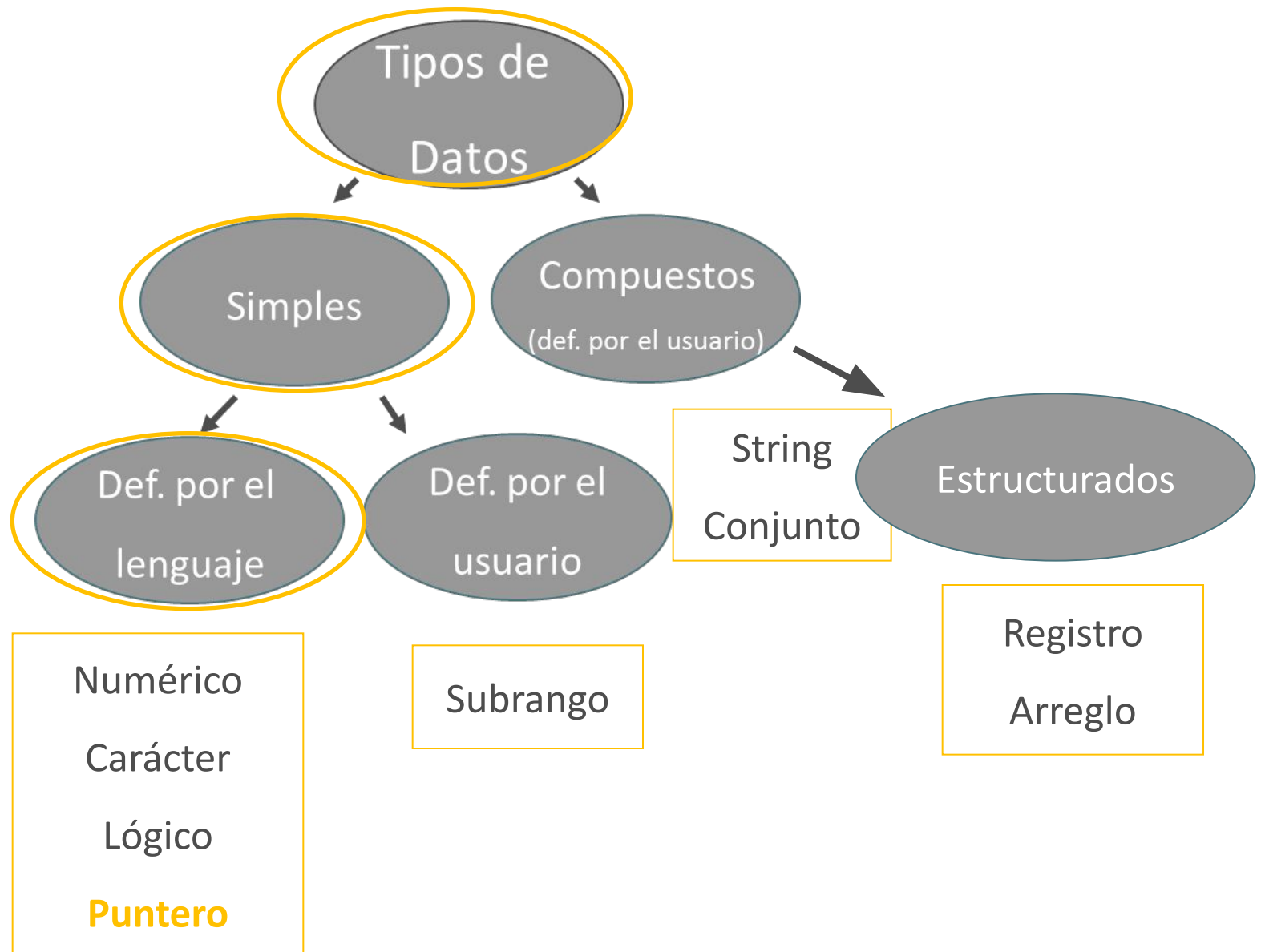
**PROGRAMACIÓN I**

AÑO 2025

# 1 Tipo de dato PUNTERO

- Características
- Ejercitación

# PUNTEROS: Recordemos clasificación...



# TIPO DE DATO PUNTERO

Un puntero es un tipo de dato simple que contiene la dirección de otro dato.

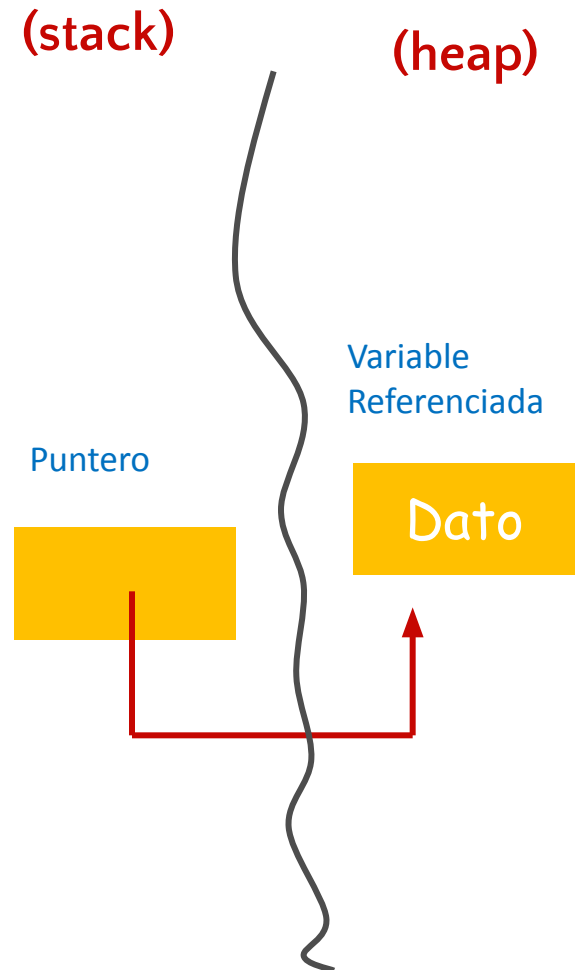
❑ Los punteros (en stack) pueden apuntar solamente a variables dinámicas, es decir, a datos que están almacenados en memoria dinámica (heap).

---

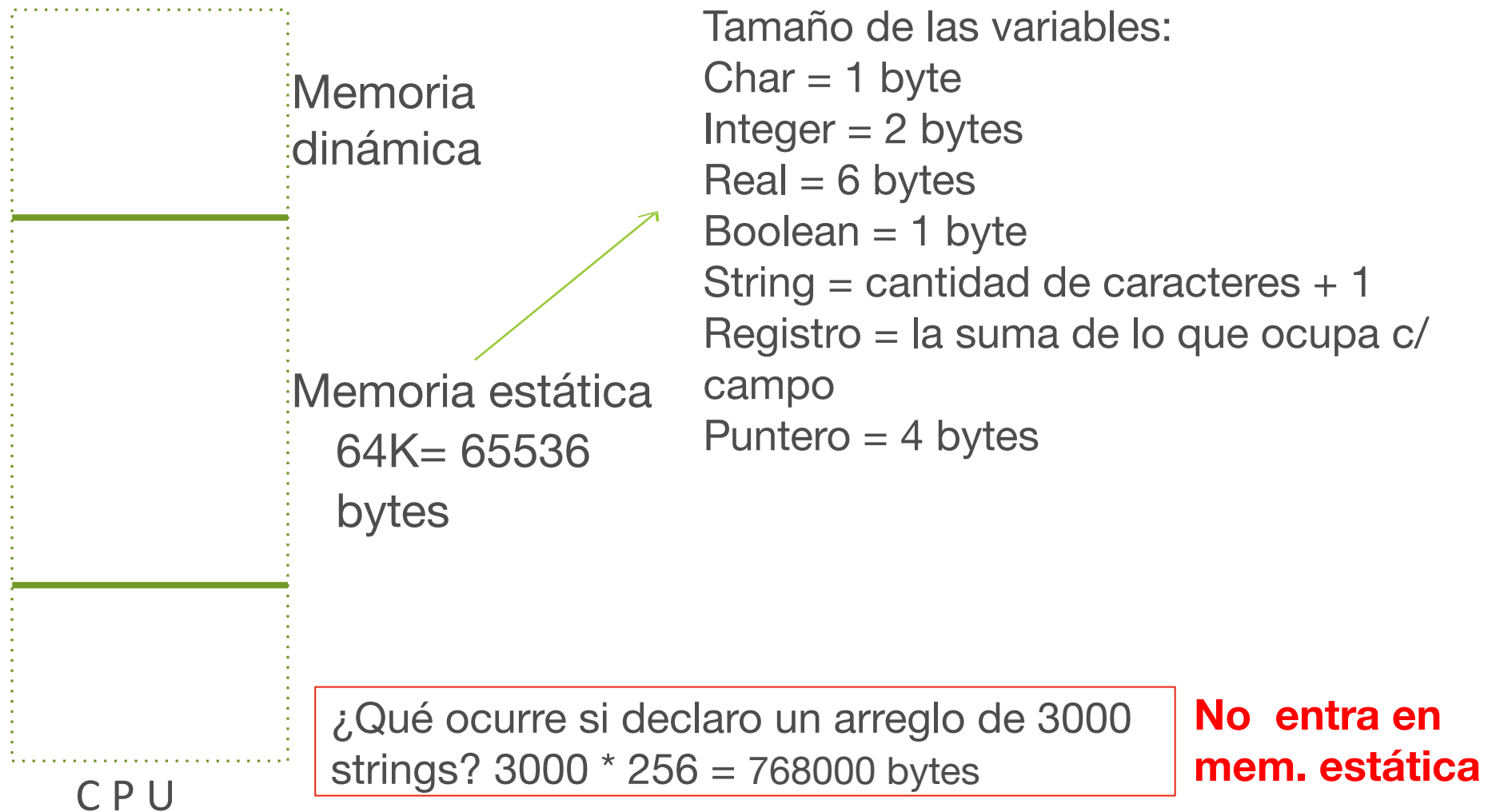
❑ Una variable de tipo puntero ocupa 4 bytes de memoria (stack) para su representación interna en Pascal.

---

❑ Cada variable de tipo puntero puede apuntar a un único tipo de dato (en Heap).



# ALOCACIÓN ESTÁTICA Y DINÁMICA



# TIPO DE DATO PUNTERO: DECLARACIÓN

- En Pascal, un tipo de dato puntero se define:

```
TYPE TipoPuntero= ^TipoVariableApuntada;
```

- Por ejemplo para definir un tipo puntero a un producto:

```
TYPE producto = record  
                                codigo: integer;  
                                stock: integer;  
                                precio: real;  
end;  
  
PunteroAProducto = ^producto;
```

Y luego se puede declarar una variable:

```
VAR Puntero : PunteroAProducto;
```

- Un dato de tipo puntero puede apuntar a una variable de cualquier tipo.

# TIPO DE DATO PUNTERO: EJEMPLOS DE DECLARACIÓN

*{declaración de tipos}*

**type**

TipoCadena = **array** [1..10] **of** char;

PtrCadena = ^TipoCadena;

PtrReal = ^real;

TipoString = string[20];

PtrString = ^TipoString;

Datos = **record**

    Nombre: TipoString;

    Apellido: TipoString;

    Edad: integer;

    Altura: real

**End;**

PtrDatos = ^datos;

*{declaración de variables}*

**var**

    precio : PtrReal; (o ^real)

    t : PtrString;

    frase : PtrString;

    s : TipoString;

    puntero : PtrCadena;

    p, q : PtrDatos;

Analicemos  
¿Memoria estática?  
¿Memoria dinámica?

**Importante:** Pascal permite que las funciones retornen punteros, por ejemplo,  
Function Verificar (s: TipoString): PtrString;

# TIPO DE DATO PUNTERO: OBSERVACIONES IMPORTANTES

- ❑ Una variable de tipo puntero ocupa una cantidad de memoria estática fija (4 bytes), independiente del tipo de dato al que apunta.
- ❑ Un dato referenciado o apuntado, como los ejemplos vistos, no tienen memoria asignada, o lo que es lo mismo no existe inicialmente espacio reservado en memoria para este dato.
- ❑ Para poder emplear variables dinámicas es necesario usar el tipo de dato PUNTERO que permite referenciar nuevas posiciones de memoria que no han sido utilizadas a priori y que se van a crear y destruir en tiempo de ejecución.



# TIPO DE DATO PUNTERO: OPERACIONES

- Las variables dinámicas son por definición aquellas que se crean cuando se necesitan y se destruyen cuando ya han cumplido con su cometido.
- En Pascal la creación y destrucción de variables dinámicas se realiza mediante los siguientes procedimientos:

**New (puntero)**

**Dispose (puntero)**

# TIPO DE DATO PUNTERO: OPERACIONES

- ❑ Asignación de un valor a una variable puntero
- ❑ Asignación de valor al objeto “referenciado” o “apuntado” por el puntero
- ❑ Acceso a la información del objeto “referenciado” o “apuntado” por el puntero
- ❑ Operaciones de Entrada / Salida???
- ❑ Operaciones de comparación
- ❑ Eliminación de un objeto apuntado que no se necesita

# TIPO DE DATO PUNTERO - OPERACIONES

- Asignación Nula a una variable puntero

## Type

```
TipoString = string[20];  
Datos = record  
    Nombre: TipoString;  
    Apellido: TipoString;  
    Edad: integer;  
    Altura: real  
end;  
PtrDatos = ^datos;
```

## Var

```
p, q : PtrDatos;
```

p := Nil;

q := Nil;

p 

q 

# TIPO DE DATO PUNTERO - OPERACIONES

- Asignación de un valor a una variable puntero

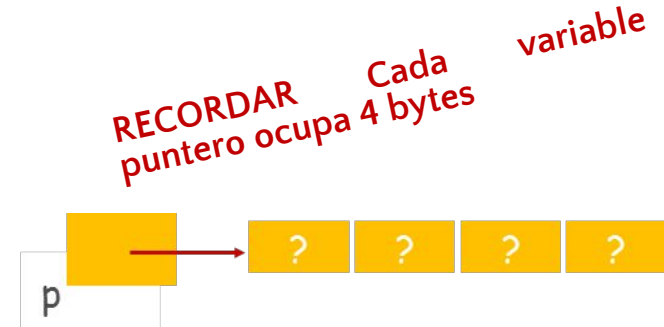
## Type

```
TipoString = string[20];  
Datos = record  
    Nombre: TipoString;  
    Apellido: TipoString;  
    Edad: integer;  
    Altura: real  
end;  
PtrDatos = ^datos;  
PtrReal = ^real;  
PtrString = ^TipoString;
```

## Var

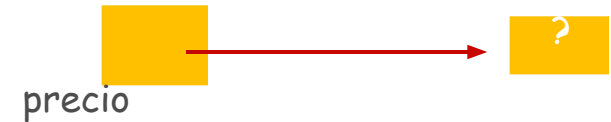
```
p      : PtrDatos;  
precio : PtrReal;  
frase  : PtrString;
```

## New (p);



Al ejecutar el New, se reserva espacio para el registro (50 bytes)

## New (precio);



Al ejecutar el New, se reserva espacio para el real (6 bytes)

## New (frase);



Al ejecutar el New, se reserva espacio para el string (21 bytes)

# TIPO DE DATO PUNTERO - OPERACIONES

- Asignación de un valor a una variable puntero

Si se tiene



Se puede hacer

**$p := q$**



# TIPO DE DATO PUNTERO - OPERACIONES

## ■ Asignación de valor a la variable apuntada

### Type

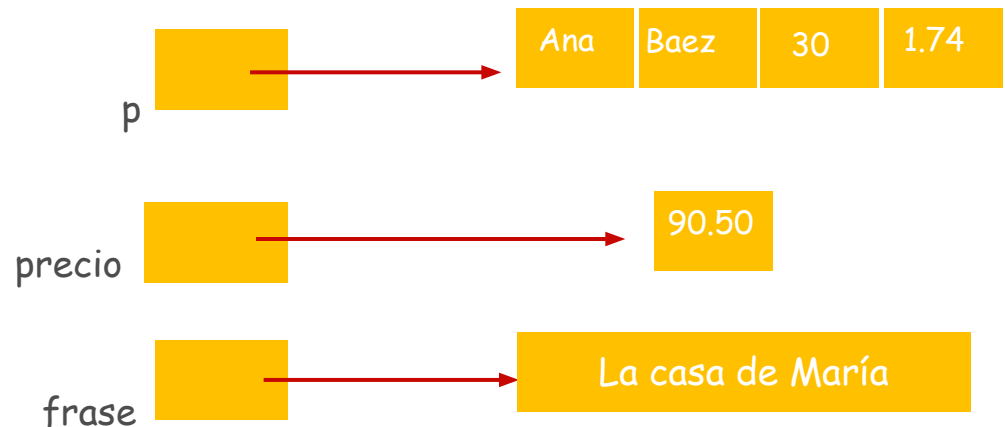
```
TipoString = string[20];  
Datos = record  
    Nombre: TipoString;  
    Apellido: TipoString;  
    Edad: integer;  
    Altura: real  
end;  
PtrDatos = ^datos;  
PtrReal = ^real;  
PtrString = ^TipoString;
```

### Var

```
p      : PtrDatos;  
precio : PtrReal;  
frase  : PtrString;
```

### Begin

```
New(p); New(peso); New(frase);  
read (p^.nombre);      read (P^) MAL  
read (p^.apellido);  
p^.edad := 30;  
p^.altura := 1.74;  
precio^ := 90.50; read (precio^) BIEN  
frase^ := 'La casa de María'; read (frase^) BIEN
```



# TIPO DE DATO PUNTERO - OPERACIONES

## ■ Acceso a la información de la variable referenciada

### Type

```
TipoString = string[20];  
Datos = record  
    Nombre: TipoString;  
    Apellido: TipoString;  
    Edad: integer;  
    Altura: real  
end;  
PtrDatos = ^datos;  
PtrReal = ^real;  
PtrString = ^TipoString;
```

### Var

```
p, q : PtrDatos;  
precio : PtrReal;  
frase : PtrString;
```

### Begin

```
.....  
write (P^.nombre);  
write (P^.apellido);  
  
precio^:=precio^ + 5.10;  
write ( frase^);
```



# TIPO DE DATO PUNTERO - OPERACIONES

- Asignación de punteros

## Type

```
TipoString = string[20];
```

```
Datos = record
```

```
    Nombre: TipoString;
```

```
    Apellido: TipoString;
```

```
    Edad: integer;
```

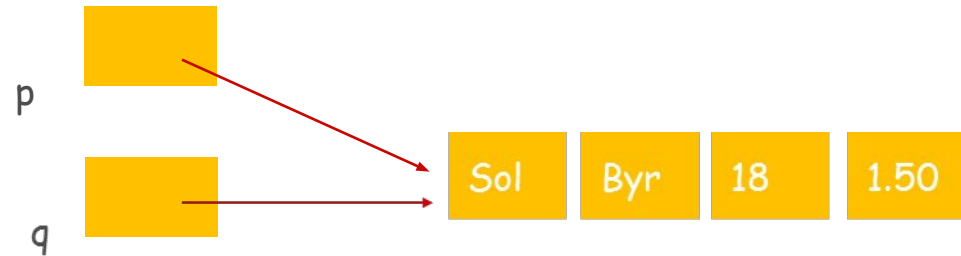
```
    Altura: real
```

```
end;
```

```
PtrDatos = ^datos;
```

## Var

```
p, q : PtrDatos;
```



```
New (q);
```

```
Read(q^.nombre, q^.apellido, q^.edad, q^.altura);
```

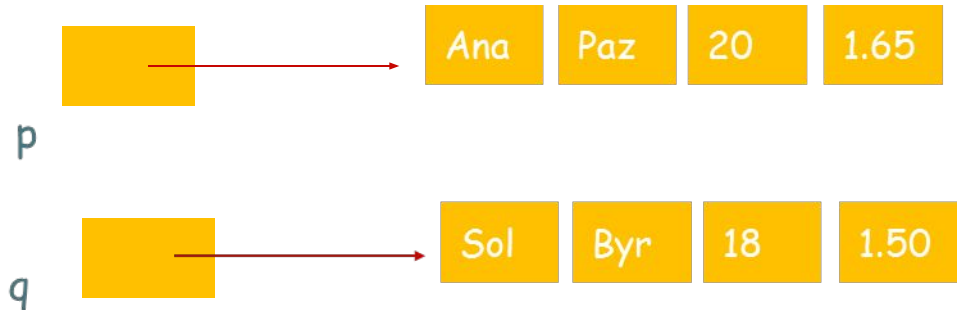
```
p := q
```



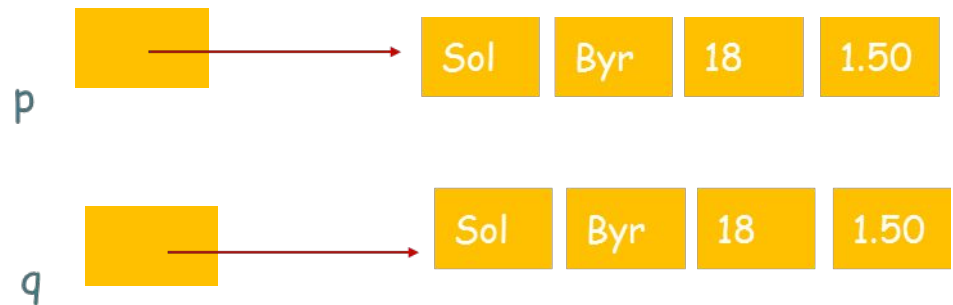
# TIPO DE DATO PUNTERO - OPERACIONES

- Asignación a la variable referenciada (asignación de objetos “referenciados”)

Si se tiene p y q como se muestra



y se hace  $\hat{p} := \hat{q}$

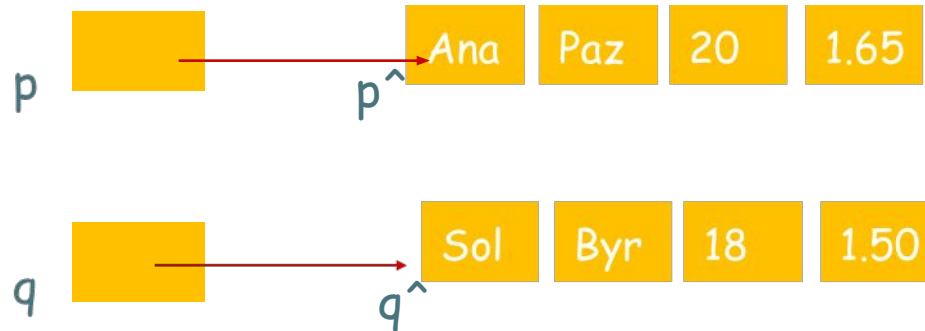


( punteros distintos apuntando a valores iguales )

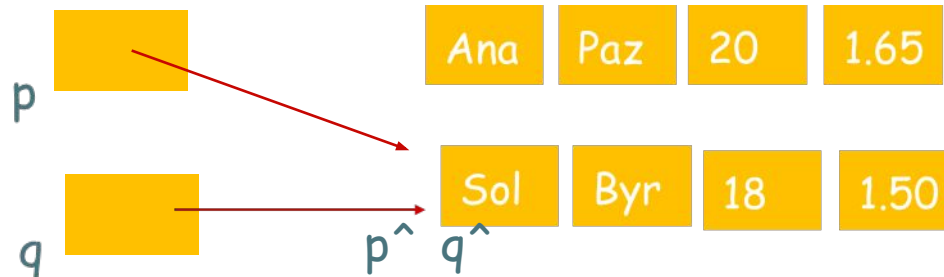
# TIPO DE DATO PUNTERO - ASIGNACIÓN

¿Qué ocurre cuando se asignan variables de tipo puntero?

Si se tiene:



Y se hace  **$p := q$** ; el efecto es:



➡ El espacio de memoria referenciado por  $p$  sigue “ocupado”, pero no es posible referenciarlo.

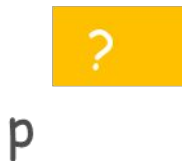
# TIPO DE DATO PUNTERO - OPERACIONES

- Eliminación de la variable referenciada □ **Dispose (variable de tipo puntero)**

Si se tiene



- ➔ Y se hace **dispose (p)**; el efecto es que se “rompe” el enlace entre  $p$  y  $p^{\wedge}$ . No es posible volver a trabajar con el dato direccionado por  $p$ , por lo tanto, ese espacio de memoria puede ser “reutilizado”.



- ➔ El contenido del puntero  $p$  queda indeterminado. No se lo puede utilizar a menos que se lo asigne nuevamente.

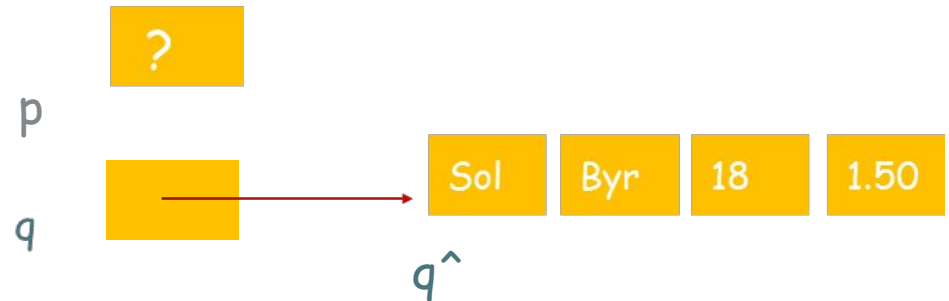
# TIPO DE DATO PUNTERO – EFECTO DE LA OPERACIÓN DISPOSE

¿Qué ocurre cuando se usa el procedimiento Dispose y cuando no se lo usa?

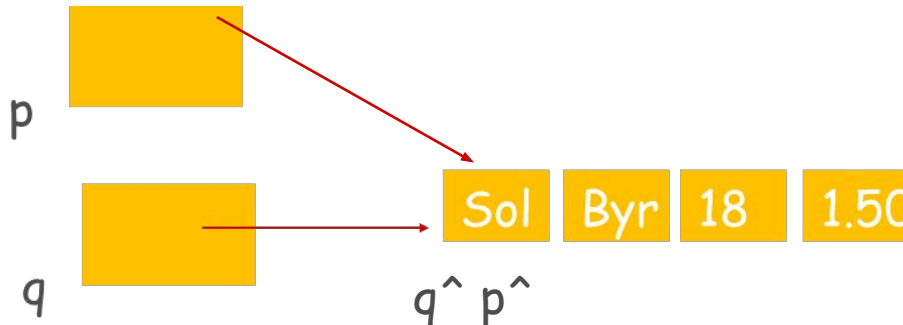
Si se tiene:



Y se hace **Dispose (p)**: el efecto es:



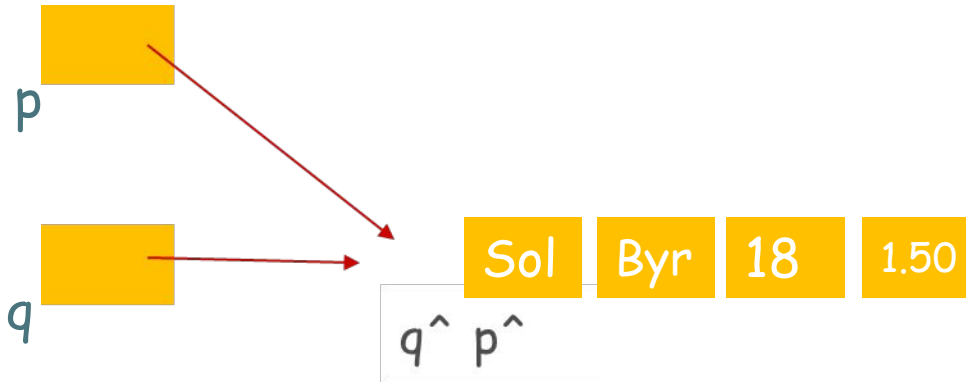
Y se hace  **$p := q$** : el efecto es:



Como el espacio de memoria referenciado por  $p$  fue “liberado”, entonces puede ser reutilizado.

# TIPO DE DATO PUNTERO – EFECTO DE LA OPERACIÓN DISPOSE

Supongamos que:



¿Qué ocurre si se hace **Dispose (p)**?



El espacio de memoria referenciado por ese puntero será “liberado”, por lo tanto, NINGÚN otro puntero que esté referenciando esa dirección podrá utilizarla.

# TIPO DE DATO PUNTERO - OPERACIONES

- Operaciones de comparación

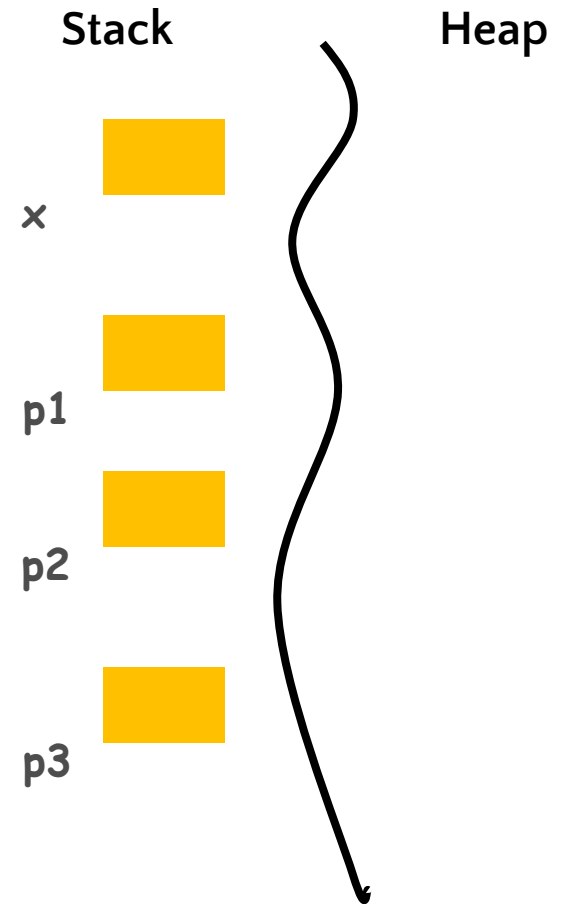
Pueden aparecer en expresiones relacionales como:  $p = q$     y     $p \neq q$

- Operaciones de Entrada / Salida

No se puede leer y/o escribir una variable puntero. Si se puede leer y/o escribir los objetos que ellos referencian dependiendo del tipo apuntado como ya se vió.

# TIPO DE DATO PUNTERO – EJERCITACIÓN - ¿Qué imprime?

```
program punteros;  
Type  
  pint= ^integer;  
  
var  x : integer;  
     pun1, pun2, pun3: pint;  
begin  
  x:= 10 ;  
  new (pun1) ;  
  new(pun2) ;  
  pun1^ := x ;  
  pun2^ := pun1^ + 1 ;  
  x:= 20;  
  pun1^:= x ;  
  pun3 := pun1 ;  
  pun1^ := pun1^ + pun2^;  
  writeln ('Elemento en pun1: ', pun1^);  
  writeln ('Elemento en pun2: ', pun2^);  
  writeln ('Elemento en pun3: ', pun3^);  
  new (pun1); new (pun3); pun1:= pun3;  
  pun1:= nil;  
  dispose (pun2); dispose (pun1);  
End.
```



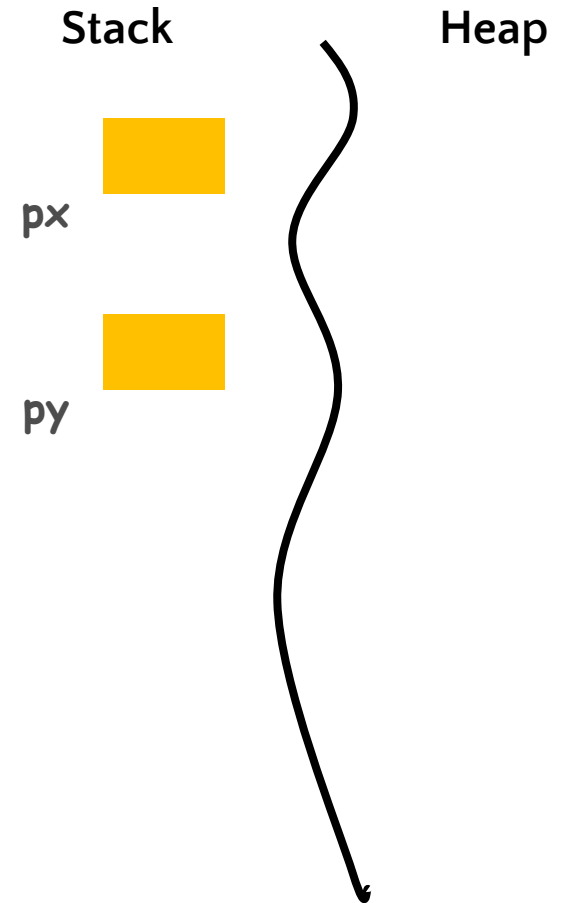
# TIPO DE DATO PUNTERO – EJERCITACIÓN ¿Qué imprime?

```
program punterosC;
type
  cadena = string[50];
  puntero_cadena = ^cadena;

procedure cambiar(var pun1: puntero_cadena;
                  pun2: puntero_cadena);

begin
  pun1:= pun2;
end;

var
  px, py: puntero_cadena;
begin
  new(px);
  px^:= 'Hoy es lunes';
  writeln('El contenido de px^: ', px1^);
  cambiar(py, px);
  writeln('El contenido de px^: ', px^);
  writeln('El contenido de py^: ', py^);
end.
```





## ¿Cuánta memoria es ocupada por PE?

**Program** ejercicio;

**Type**

Estudiante = **record**

Nombre : string [20];

Calificacion : integer;

**end;**

PunEstudiante = ^ Estudiante;

**Var**

PE: PunEstudiante;

**Begin**

...

**end.**

**Sólo 4 bytes,  
en memoria  
estática.**

Tamaño de las variables:

Char = 1 byte

Integer = 2 bytes

Real = 6 bytes

Boolean = 1 byte

String = cantidad de caracteres + 1

Registro = la suma de lo que ocupa c/  
campo

Puntero = 4 bytes

# TIPO DE DATO PUNTERO – EJERCITACIÓN

**Program** ejercicio;

**Type**

Estudiante = **record**

Nombre : string [20];

Calificacion : integer;

**end;**

PunEstudiante = ^ Estudiante;

**Var**

PE: PunEstudiante;

**Begin**

new (PE);

...

**end.**

**Se reservan 23 bytes (21 + 2) en tiempo de ejecución, en la heap.**

# TIPO DE DATO PUNTERO – EJERCITACIÓN

Analicemos el siguiente ejemplo:

```
Type cadena = string [255];
```

```
    pcadena = ^ cadena;
```

```
    puntero1 = array[1...3000] of cadena;
```

```
    puntero2 = array[1..3000] of pcadena;
```

```
Var pun1: puntero1; pun2: puntero2;
```

```
    i: integer;
```

```
begin
```

```
    for i:= 1 to 3000 do
```

```
        begin
```

```
            read (nom);
```

```
            pun1 [i] := nom;
```

```
            new (pun2[i]); pun2[i]^:= nom;
```

```
        end;
```

```
end.
```

¿Cuánta memoria ocupan pun1, pun2 e i?

Luego de la ejecución del for, ¿cuánta memoria hay ocupada?