TEORIA

12

TEMAS de la CLASE

1 Tipo de dato PUNTERO

- Características
- Ejercitación

Alocación de Memoria

Hasta ahora hemos trabajado con el tipo de Alocación Estática de la memoria (stack)

Alocación Estática (stack) --> variables estáticas

- Las estructuras de datos hasta ahora vistas se almacenan estáticamente en la memoria física del ordenador.
- El espacio de memoria se reserva con anticipación y no cambia durante la ejecución del programa.
- Esto permite una comprobación de tipos en tiempo de compilación

Inconvenientes de la configuración estática

- Su rigidez, ya que estas estructuras no pueden crecer o decrecer durante la ejecución del programa.

Alocación de Memoria

Alocación Dinámica (Heap) ->variables dinámicas ó referenciadas

- Los espacios de memoria asignados a las variables dinámicas se reservan y se liberan durante la ejecución del programa.
- No hay espacio de memoria reservado

Ventajas de la configuración dinámica:

- Su flexibilidad, ya que las estructuras “dinámicas” pueden crecer o decrecer durante la ejecución del programa.

En Pascal --> punteros (pointer)

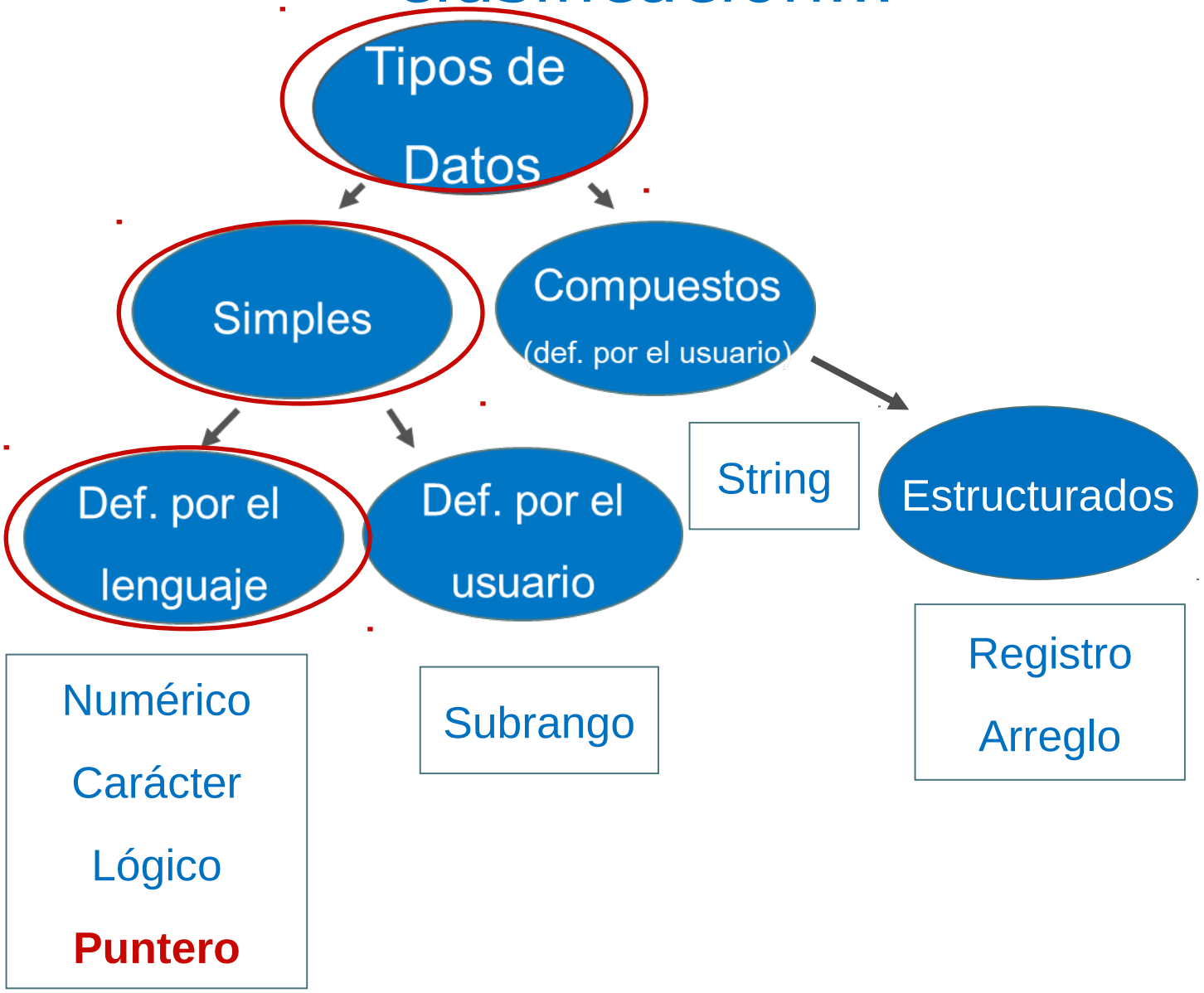
Tipo de dato Puntero

- Un puntero es un tipo de variable usada para almacenar la dirección en memoria dinámica de otra variable, en lugar de un dato convencional.

- Mediante la variable de tipo puntero (en stack) se accede a esa otra variable, almacenada en la dirección de memoria dinámica que señala el puntero. Es decir, el valor de la variable de tipo puntero es una dirección de memoria.

- Se dice que el puntero apunta o señala a la variable almacenada en la dirección de memoria (heap) que contiene el puntero. Lo que nos interesa es el dato contenido en esa variable apuntada. No hay que confundir la variable apuntada con el puntero.

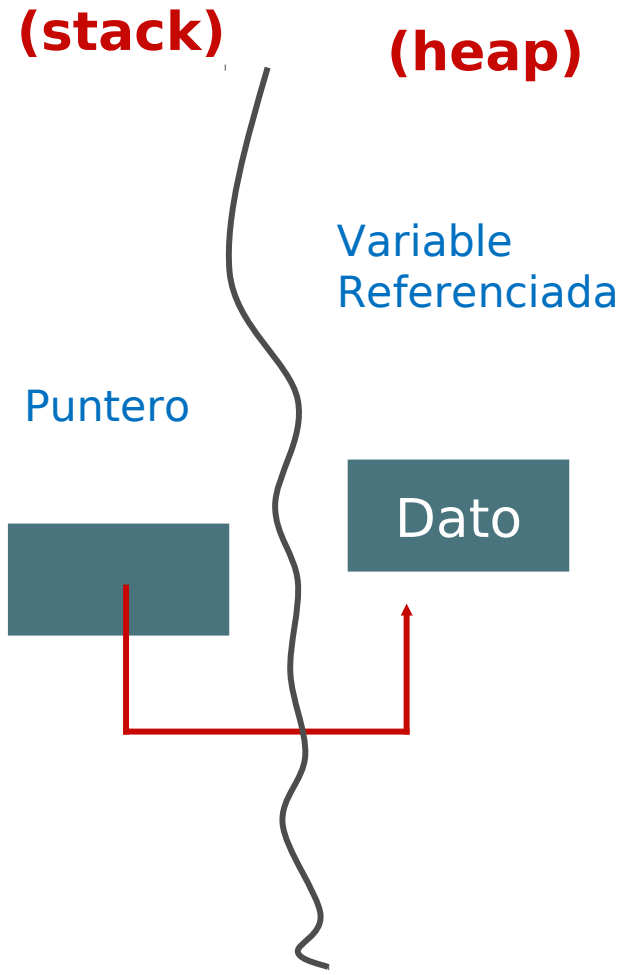
Punteros: Recordemos clasificación...



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).

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 string,

```
TYPE
    cadena30=string[30];
    PunteroAcadena30 = ^cadena30;
```

Y luego se puede declarar una variable:

```
VAR Puntero : PunteroAcadena30;
```

- Un dato de tipo puntero puede apuntar a una variable de cualquier tipo, incluso tipos estructurados.

Ejemplos de declaraciones en Pascal

```
{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
   peso : PtrReal; (o ^real)

    t : PtrString;
    frase : PtrString;
    s : TipoString;
    puntero : PtrCadena;
    p, q : PtrDatos;
```

Analicemos

¿Memoria estática?

¿Memoria dinámica?

Punteros - 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 declaradas a priori y que se van a crear y destruir en tiempo de ejecución.

Punteros - 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)**

Punteros - Operaciones

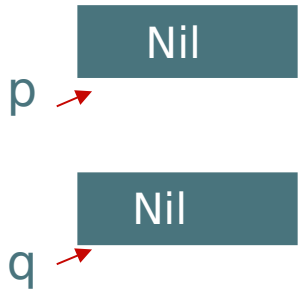
- Asignación de un valor a una variable puntero
- Asignación de valor al objeto “referenciado” o “apuntado” por el puntero
- Acceso a la información del objeto “referenciado” o “apuntado” por el puntero
- Eliminación de un objeto apuntado que no se necesita
- Operaciones de Entrada / Salida???
- Operaciones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

Punteros – 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;

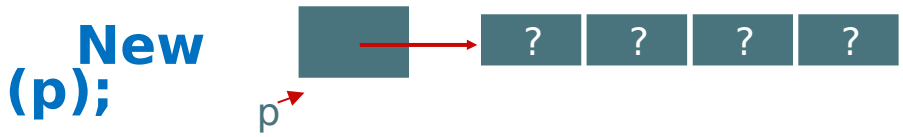
q := Nil;



Punteros – Asignación de un valor a una variable puntero

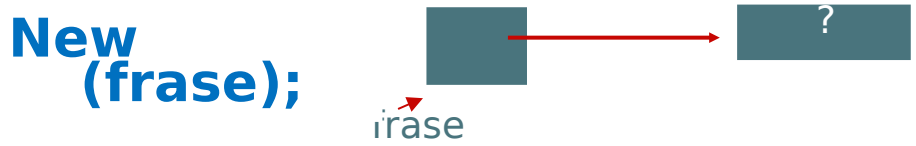
```
Type
 TipoString = string[20];
 Datos = record
          Nombre: TipoString;
                    Apellido:
TipoString;
          Edad: integer;
         Altura: real
        end;
 PtrDatos = ^datos;
 PtrReal = ^real;
 PtrString = ^TipoString;

Var
 p, q : PtrDatos;
 peso : PtrReal;
 frase : PtrString;
```

RECORDAR Cada variable puntero ocupa 4 bytes



Al ejecutar el New, se reserva espacio para el registro (54 bytes)

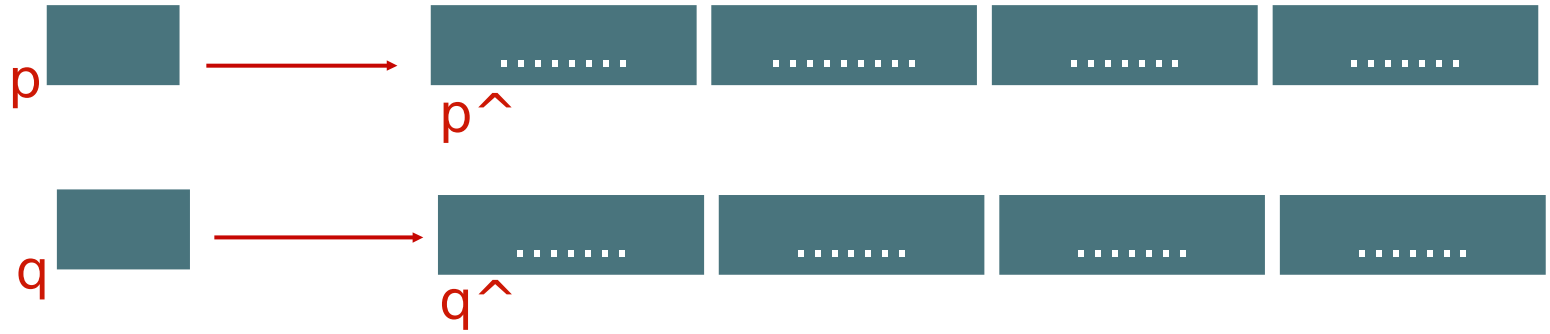
Al ejecutar el New, se reserva espacio para el real (8 bytes)



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

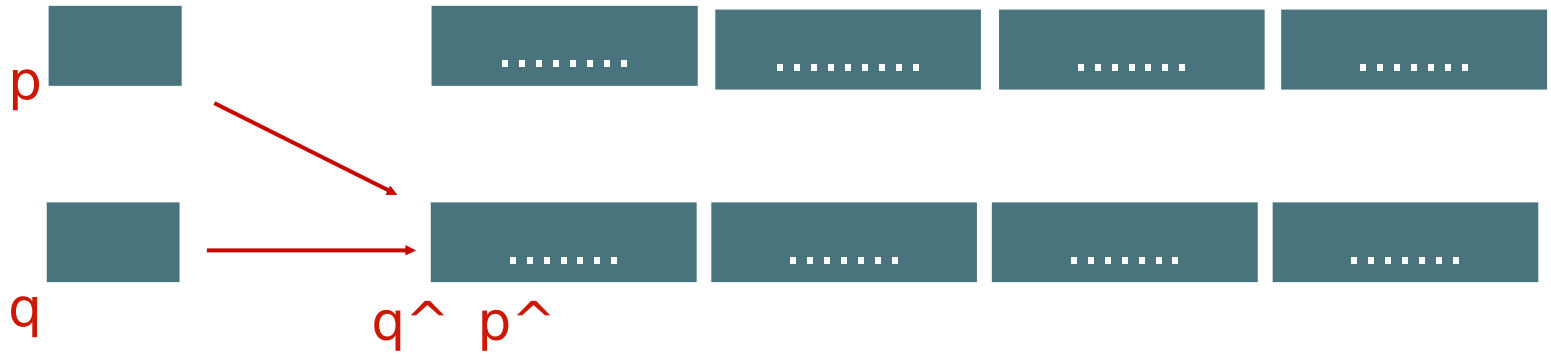
Punteros – Asignación de un valor a una variable puntero

Si se tiene



Y si se hace

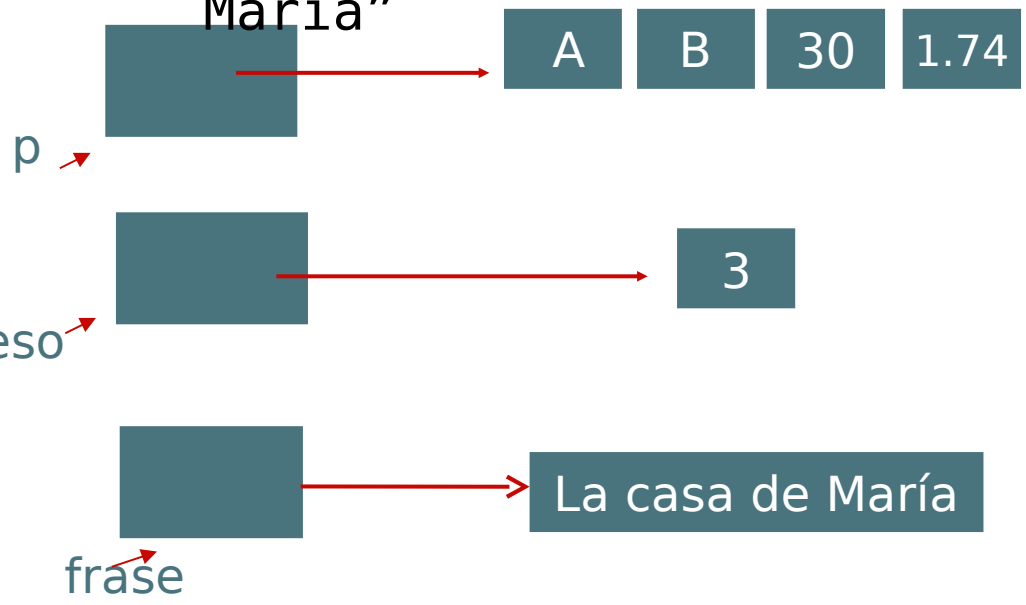
p := q



Punteros – 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, q : PtrDatos;
  peso : PtrReal;
  frase : PtrString;
```

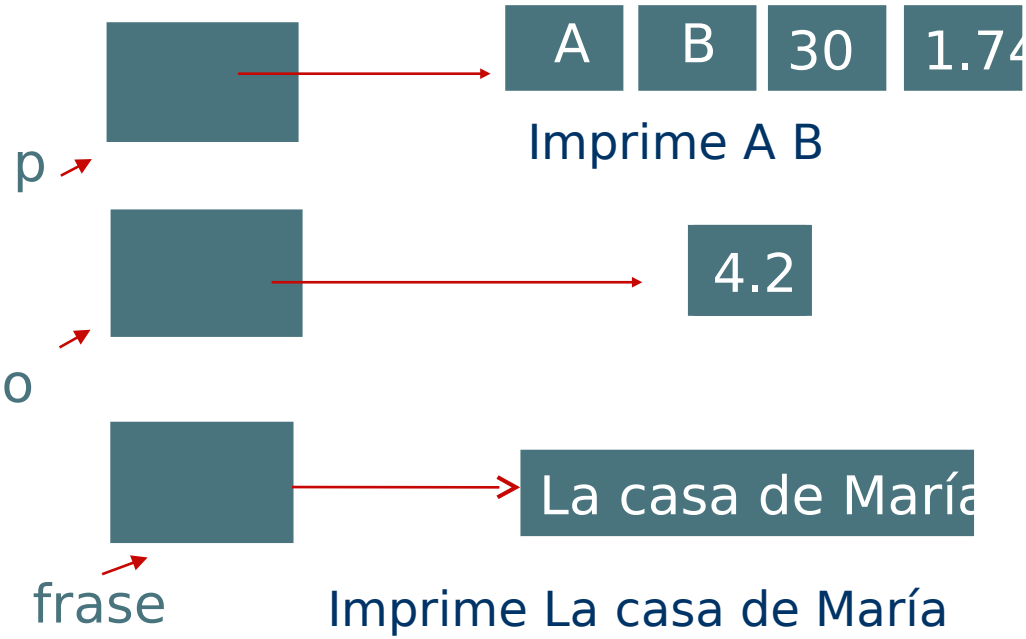
```
Begin
New(p);                New(peso);
   New(frase);
read (P^.nombre);
read (P^.apellido);
p^.edad := 30;
p^.altura := 1.74;
peso^ := 3;

frase^  :=  “La  casa  de
   María”
```



Punteros – Acceso a la información de la variable referenciada

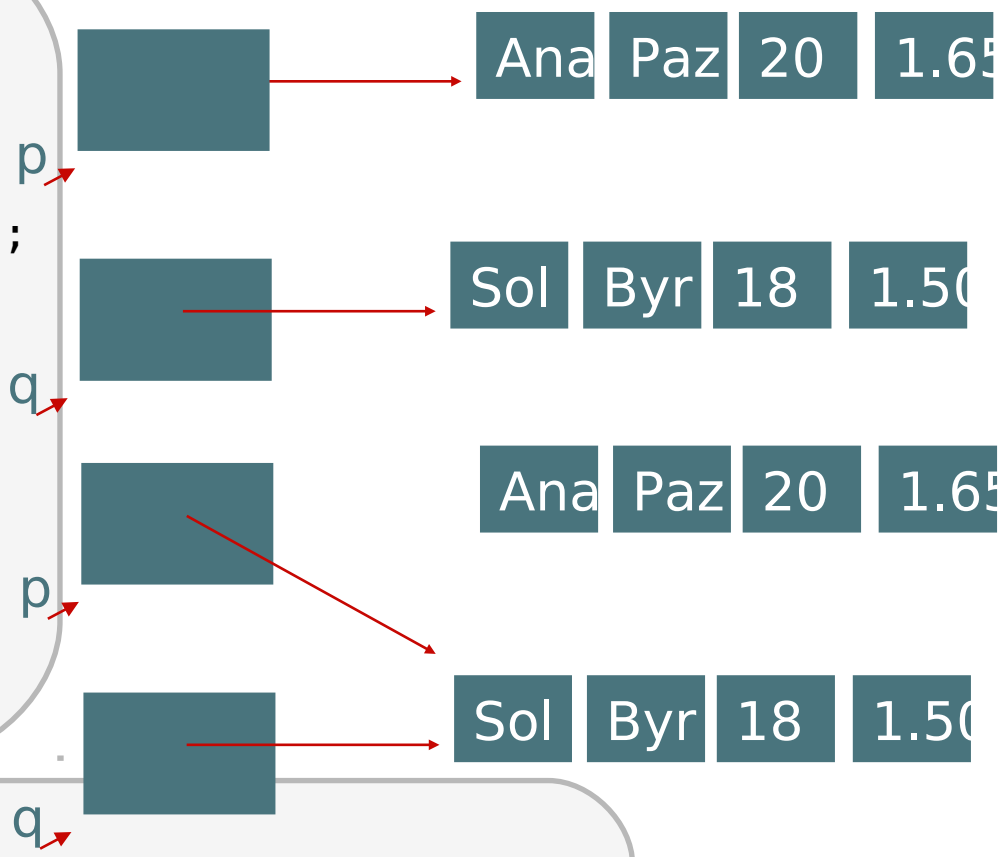
```
Type
 TipoString = string[20];
 Datos = record
                        Nombre:
TipoString;

Apellido:TipoString;
          Edad: integer;
          Altura: real
          end;
 PtrDatos = ^datos;
 PtrReal = ^real;
 PtrString = ^TipoString;

Var
   p, q : PtrDatos;
   peso : PtrReal;
   frase : PtrString;
```

```
Begin
……..
write (P^.nombre);
write  (P^.apellido);
peso^:=peso^     +
  1.2;
write( frase^);
```



Punteros – Asignación de punteros

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

Var
  p, q : PtrDatos;
```

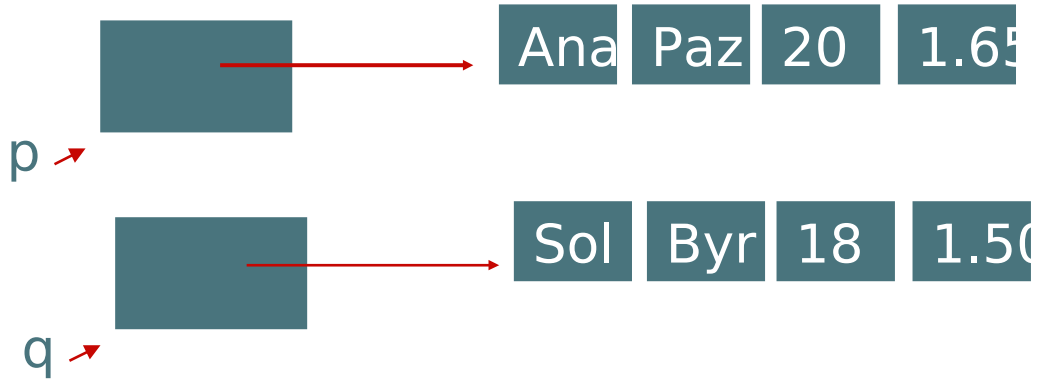



```
New (p);
Read(p^.nombre,p^.apellido,p^.edad,
   p^.altura);
New (q);
Read(q^.nombre,     q^.apellido,     q^.edad,
   q^.altura);
 p := q
```

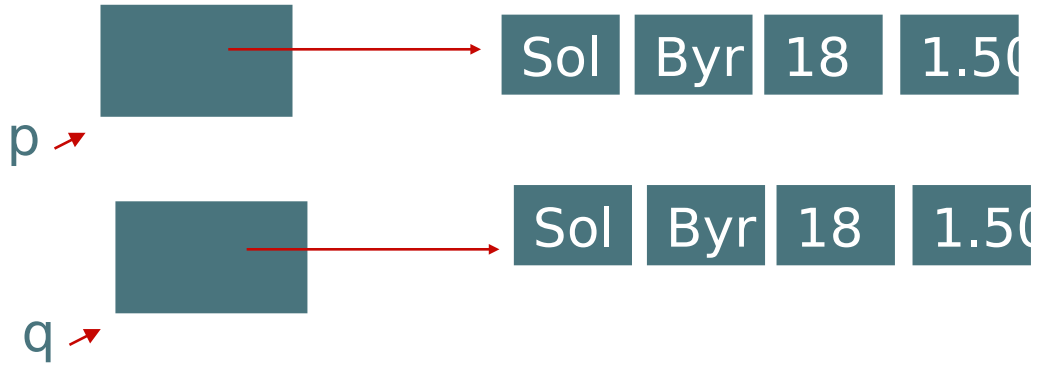
Punteros – Asignación a la variable referenciada

- **Asignación de objetos “referenciados”**

Si se tiene p y q como se muestra



y se hace **p^ := q^**



(punteros distintos apuntando a valores iguales)

Otras operaciones con Punteros

- **Operaciones de Entrada / Salida :**

No podemos leer y escribir punteros. Si podemos leer y escribir los objetos que ellos referencian dependiendo del tipo apuntado como ya se vió.

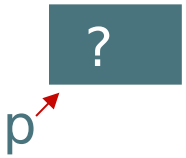
- **Operaciones de comparación :**

Pueden aparecer en expresiones relacionales como: p = q y p <> q

Operación de Eliminación de la variable referenciada

- **Eliminación de un objeto apuntado que no se necesita ➔ Dispose (p)**

➡ Si hacemos dispose (p); el efecto es que se “rompe” el enlace entre p y p^. No es posible volver a trabajar con el dato direccionado por p, por lo tanto, ese espacio de memoria puede ser “reutilizado”.

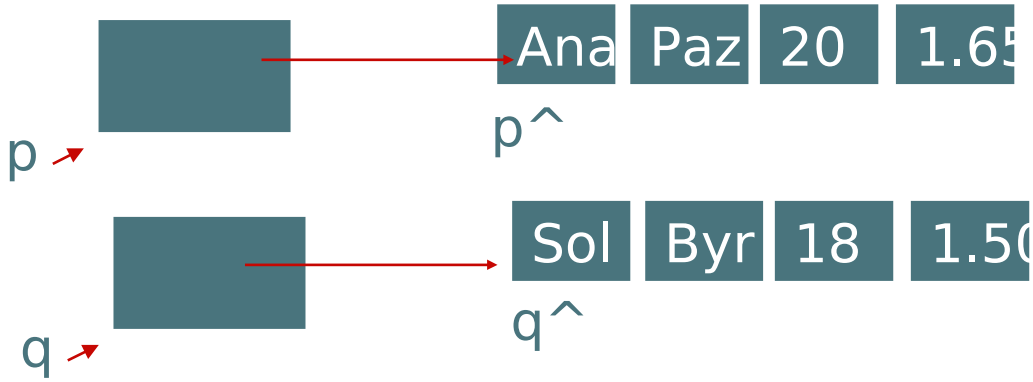


➡ El contenido del puntero p queda indeterminado. No se lo puede utilizar a menos que se lo asigne nuevamente.

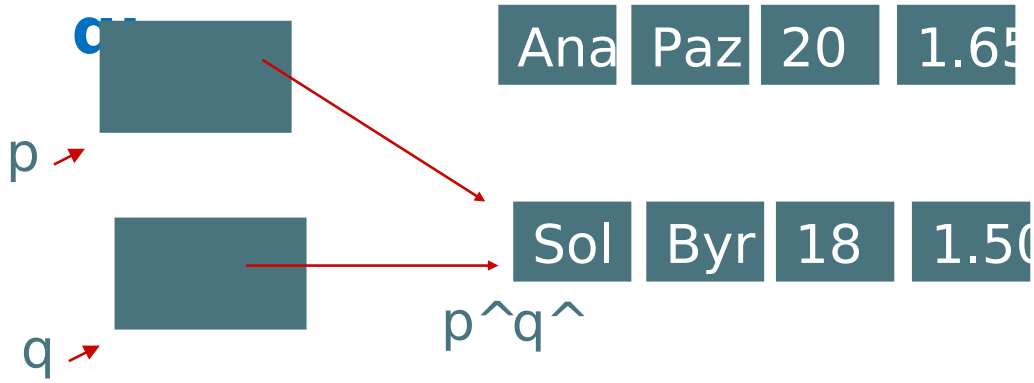
Efecto de la operación Dispose

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

Si se tiene:



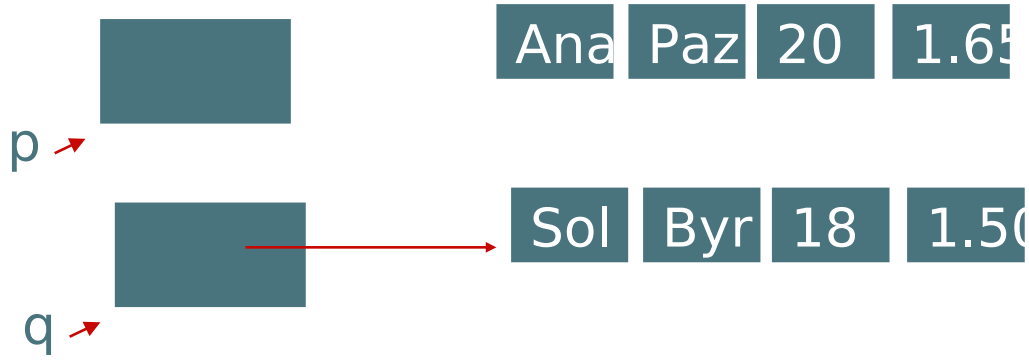
Y se hace **p := q** el efecto es:



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

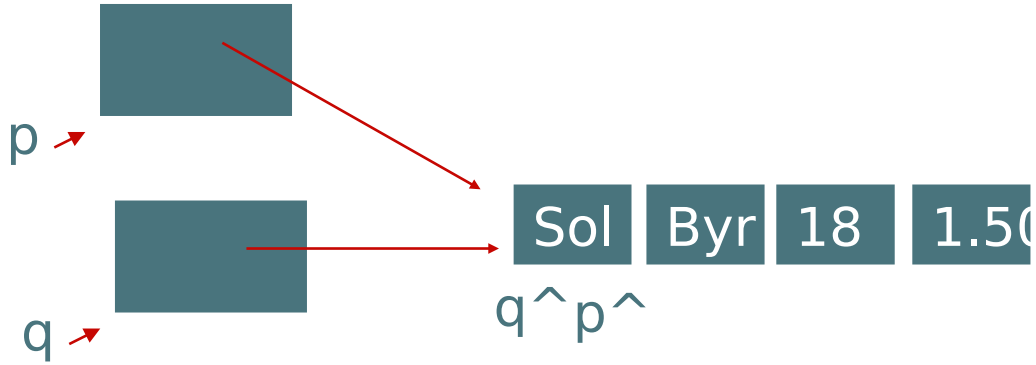
Efecto de la operación Dispose

Si se tiene:



Y se hace:

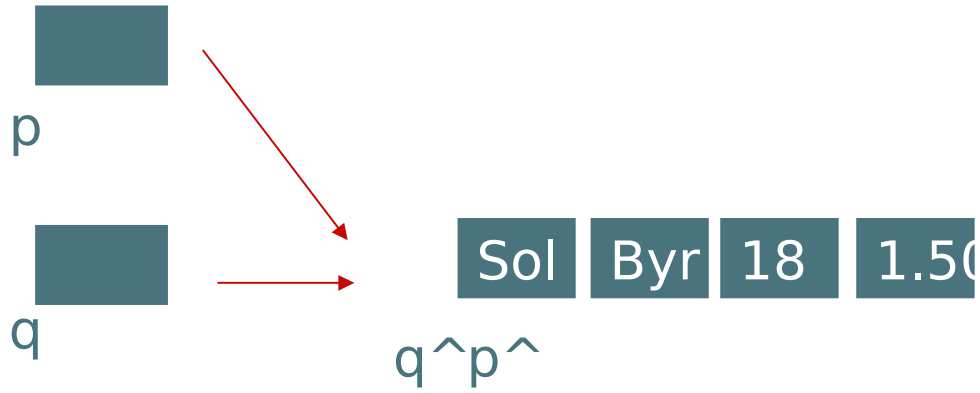
```
Dispose
   (p);
p := q;
```



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

Efecto de la operación Dispose

Supongamos que:



¿Qué ocurre si se hace

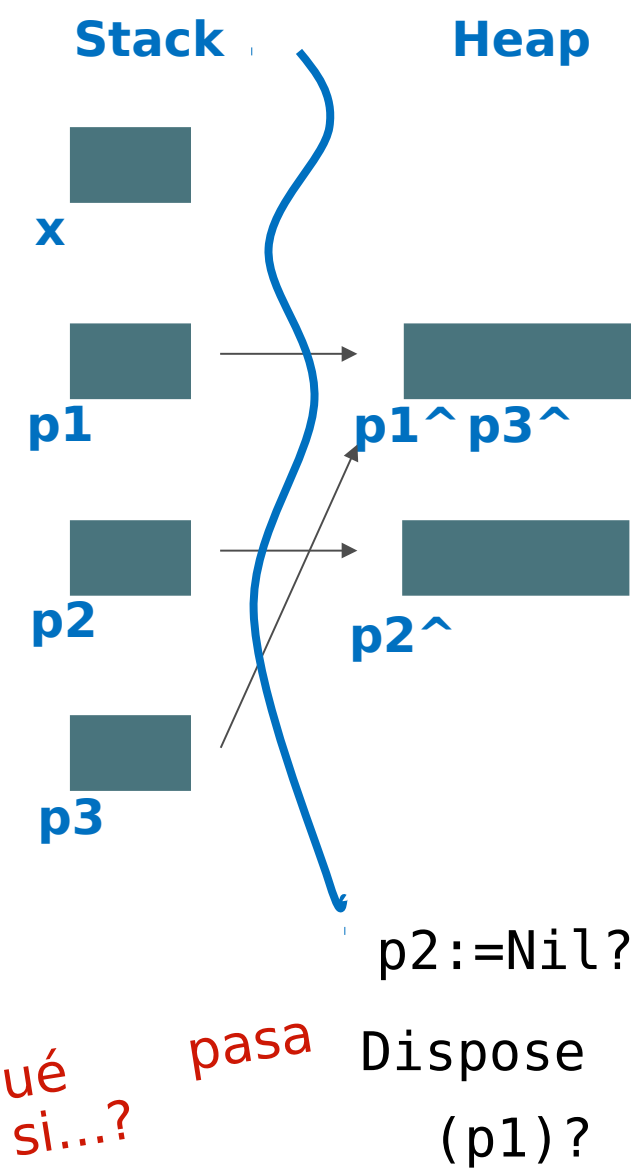
Dispose (p)?

Dispose (q)?

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

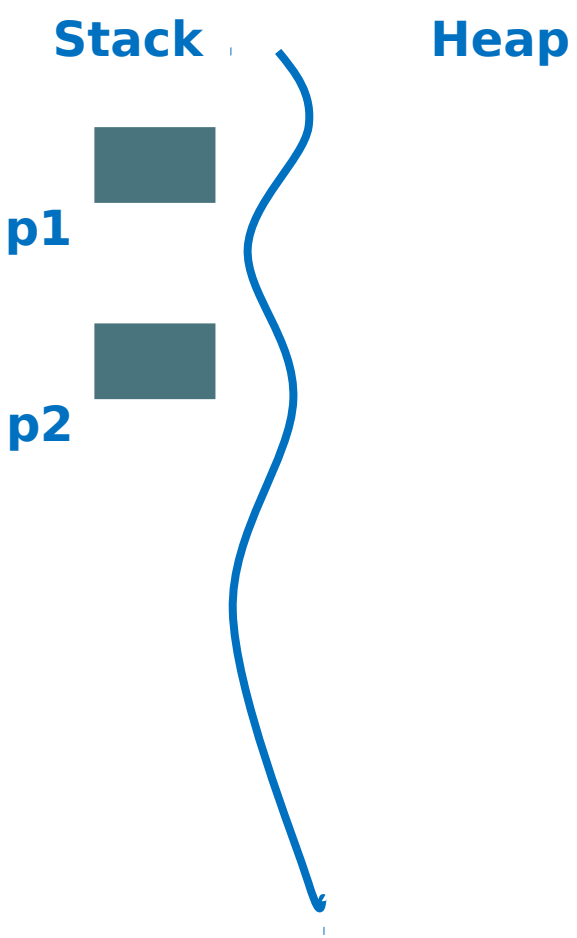
Ejercitación I

```
Type
  pint= ^integer;

var  x : integer;
       p1, p2, p3: pint;
begin
 x:= 5;
 new (p1) ;
 new(p2) ;
 p1^ := x ;
 p2^ := p1^ + 1 ;
 x:= 20;
 p1^:= x ;
 p3 := p1 ;
 p1^ := p1^+p2^;
  writeln ('Elemento en p1: ',
p1^);
  writeln ('Elemento en p2: ',
p2^);
  writeln ('Elemento en p3: ',
p3^);
End.
```



Ejercitación II: ¿Qué imprime cada writeln?

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

procedure       cambiar(var       pun1:
puntero_cadena;
                                  pun2:
puntero_cadena);
begin
  pun1:= pun2;
end;

var
  p1, p2: puntero_cadena;
begin
  new(p1);
  p1^:= 'Hoy es lunes';
    writeln('El  contenido  de  p1^:  ',
p1^);
  cambiar(p2, p1);
    writeln('El  contenido  de  p1^:  ',
p1^);
```



Se leen datos correspondientes a los productos de un supermercado. La lectura finaliza con nombre igual a ZZZ. Obtener un listado con los nombres de los productos con precio entre 25 y 50 pesos e informar la cantidad de productos que cumplen con esa condición.

Nota: el producto se guarda en memoria dinámica.

¿Declaración de tipos?

¿Declaración de variables?

```
program punteros;
type
  cad10 = string[10];
  producto= record
              nombre: cad10;
              precio: real;
            end;
  ptrproducto = ^ producto;
var
  pprod: ptrproducto;  total : integer;

procedure leerproducto (var p:producto);
 begin
   readln (p.nombre);
   if p.nombre <> 'ZZZ' then readln (p.precio);
 end;

begin
 total:=0;
 new (pprod);
 leerproducto (pprod^);
 while ( pprod^.nombre <> 'ZZZ') do begin
    if  (pprod^.precio  >=25)  and  (pprod^.precio
   <=50) then begin
      writeln ('producto: ', pprod^.nombre);
      total:= total + 1
   end;
   leerproducto (pprod^)
 end;
 write ('Total:', total);
end.
```