



Programación II

Gestión dinámica de la memoria: Punteros

Resumen sobre el uso de punteros y variables dinámicas

Operación	Código	Notas
Definición y declaración.	<pre>type pEntero= ^integer; var p: pEntero;</pre>	
Inicialización	<code>p:= NIL;</code>	Es conveniente igualar a NIL al inicio del programa (a menos que hagamos <code>new(p)</code>)
Creación de la variable dinámica	<code>new(p)</code>	No se puede utilizar la variable <code>p^</code> sin haber hecho el <code>new(p)</code> .
Destrucción de esta variable	<code>dispose(p)</code>	Siempre se debe realizar cuando no necesitamos la variable. Después se debe hacer <code>p:=NIL</code>
Acceso al contenido de la variable dinámica	<pre>p^:= valor; vble:= p^;</pre>	
Asignación de valores a punteros	<pre>p:= NIL; p:= q;</pre>	Son las dos únicas sentencias de asignación permitidas.
Comparación de punteros	<pre>p= q; p<>q; p= NIL; p <> NIL;</pre>	

NOTACIÓN	
<code>type pEntero= ^integer;</code>	<code>p</code> = puntero
<code>var p: pEntero;</code>	<code>p^</code> = contenido de la variable apuntada por <code>p</code>

Errores más comunes en el manejo de punteros

1. Errores de compilación más frecuentes.

- a) Las variables puntero sólo pueden apuntar a datos de un tipo particular. Por lo tanto, para que los punteros puedan compararse o asignarse entre sí tienen que ser del mismo tipo.
- b) Confundir el puntero (p) con la variable a la que apunta (p^{\wedge}).

2. Errores de ejecución más frecuentes.

- a) La variable referenciada por un puntero sólo existe cuando se inicia el apuntador mediante la asignación a una variable ya existente o mediante `new()`. Un error muy frecuente es intentar acceder a la variable referenciada cuando no existe. En este caso, estaremos intentando acceder a una dirección de memoria no válida y causará un error de ejecución –normalmente un *Segmentation Fault*.

■ Incorrecto:

```
type tPos=^integer;
var p: tPos;
begin
    p^:=...
```

■ Correcto.

```
type tPos=^integer;
var p, q: tPos;
begin
    new(p); p^:=... ó
    new(q); p:= q; p^:=...
```

Hay que tener en cuenta que los punteros acceden directamente a la memoria del ordenador y, por lo tanto, al acceder a posiciones de memoria no reservadas y escribir en ellas pueden ocurrir errores inesperados como escribir en el propio código del programa.

- b) Para evitar este problema haremos que el puntero contenga el valor `nil` siempre que no apunte a una variable. Así, podremos reconocer cuándo el puntero apunta o no a una variable con sólo preguntar por su valor.

- c) Cuidaremos las expresiones compuestas de los bucles. La mayoría de los compiladores de Pascal no evalúan en cortocircuito, por lo que la sentencia:

```
while (p<>nil) and (p^ > valor) do
```

producirá un error cuando p sea nil.

- d) Hay que tener cuidado cuando tenemos varios punteros que apuntan a la misma variable, ya que la modificación de la variable por parte de uno de ellos implicará que también cambiará el contenido para los demás.
- e) Además, si uno de ellos libera la variable los demás quedarán desreferenciados (referencias perdidas).
- f) La memoria de un ordenador es grande pero no ilimitada, y puede acabarse si constantemente creamos nuevas variables sin liberar el espacio de las que ya no necesitamos. En este sentido, otro tipo de errores que no causan error de ejecución están relacionados con el **dispose**:
- 1) Dejar variables a las que ya no apuntan ningún puntero, sin haber hecho un **dispose**. Supone una pérdida de capacidad de memoria para ese programa.
 - 2) Hacer **dispose(p)** y no preocuparnos de que p apunte a alguna dirección válida o a **nil**.
 - 3) Hacer un **dispose(p)** y acceder posteriormente a p.

Paso de punteros como parámetros

■ Ejemplo: Paso de punteros por valor (o nombre).

Diseñar una rutina *EsNulo* que dado un puntero de tipo *tPEntero* devuelva verdadero o falso en función de si el contenido del puntero es el valor NULO.

```
function EsNulo(p: tPEntero): boolean;
begin
    if p<>nil then
        EsNulo:= false
    else
        EsNulo:= true;
end;
```

Aunque un código más correcto sería:

```
function EsNulo(p: tPEntero): boolean;
begin
    EsNulo:= (p = NIL);
end;
```

■ Ejemplo: Paso de punteros por variable o referencia.

Escribir una rutina *Intercambiar* que, dados dos punteros de tipo *PEntero*, intercambie sus contenidos sólo en el caso de que no sean punteros nulos. En este caso, pasaremos los punteros por *referencia*.

```
procedure Intercambiar (var p,q: tPEntero);
    var t: tPEntero;
begin
    if not EsNulo(p) and not EsNulo(q) then begin
        t:= p;
        p:= q;
        q:= t;
    end;
end;
```

Un error común es no pasar el puntero por referencia cuando hace falta. Por ejemplo, supongamos que queremos crear un procedimiento que dado un puntero cree la variable dinámica asociada a él:

```
procedure CrearVariable (var p: tPEntero);  
begin  
    new(p);  
end;
```

¿Cómo hay que pasar el parámetro? ¿Qué ocurriría si lo pasamos por valor y luego hacemos $p^{\wedge}:=3$? Daría un error de ejecución.