UNIVERSIDADE DA CORUÑA

FACULTADE DE INFORMÁTICA
Departamento de Computación
Estructura de Datos e da Información

Ejercicios de punteros: SOLUCIONES

1. Suponga que tenemos las declaraciones:

```
type
    Indice = 0..9;
    ApuntIndice = ^Indice;
var
    i : Indice;
    ApuntI : ApuntIndice;
```

   a) ¿Qué contiene entonces `ApuntI`?

   Solución: No contiene nada porque no se ha inicializado a ningún valor.

   b) Si en seguida ejecutamos la codificación...

```
New(ApuntI);
ApntI^:= 2;
I:= 4;
```

   ..¿qué contendrá entonces `ApuntI`? ¿qué contendrá `ApuntI^`?

   Solución: `ApuntI` contiene la dirección de memoria donde está almacenado un 2. `ApuntI^` es la variable apuntada por `ApuntI` y, por lo tanto, contiene un 2.

2. Después de ejecutarse el fragmento de codificación

```
type
    Cosa = integer;
    ApuntadorACosa = ^Cosa;
var
    c, cc: Cosa;
    ApuntC, ApuntCC: ApuntadorACosa;
begin
    ApuntC:= NIL;
    New(ApuntCC);
```

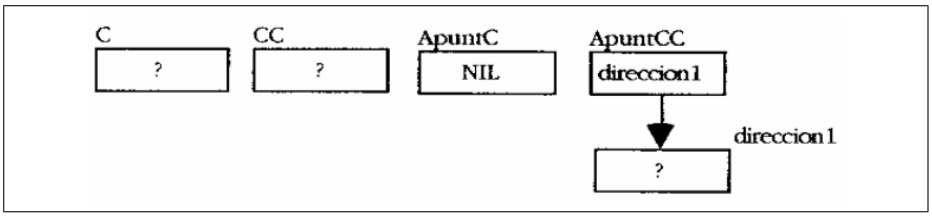
   ¿cuáles de las siguientes variables contienen basura?

   a) `ApuntC`
   b) `ApuntCC`
   c) `C`
   d) `CC`
   e) `ApuntC^`
   f) `ApuntCC^`

Solución: Contienen basura c, d, e y f

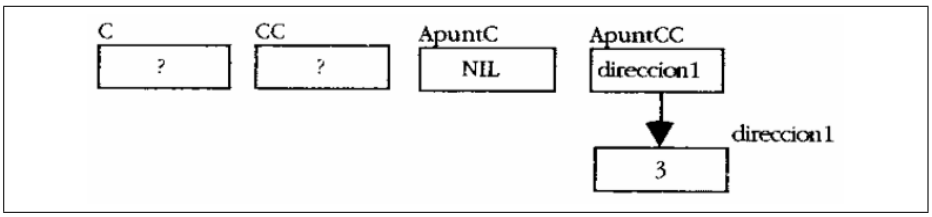
3. Preparar un diagrama de apuntadores que muestre la situación de inicialización después de ejecutarse el fragmento de codificación del ejercicio 2

   SOLUCION



4. Si después de ejecutar la codificación del ejercicio 2, el programa ejecuta la sentencia `ApuntCC^ := 3;` ¿cuál será entonces la situación de inicio? Dibuje un diagrama de apuntadores.

   SOLUCION



5. Suponga que

```
var
   Eso: ^char;
```

   ¿Será legal entonces llamar a `New(Eso^)`? ¿Y llamar a `New(Eso)`? Explique por qué.

Solución: La instrucción `New()` lleva como argumento una variable de tipo puntero. Tras su ejecución se reserva en memoria el espacio necesario para almacenar la variable apuntada por el apuntador. Además, será éste el que nos permita el acceso a esta zona de memoria. `Eso^` no es una variable de tipo puntero, sino que hace referencia a la variable apuntada por `Eso`, es decir, es de tipo `char`. Por lo tanto,`New(Eso^)` no es legal, mientras que `New(Eso)` sí lo es.

6. Suponga que

```
type
    Acertijo = real;
    ApAcert = ^Acertijo;
var
    a1, a2: ApAcert;
```

¿Cuáles de los siguientes enunciados serán legales en ese caso?

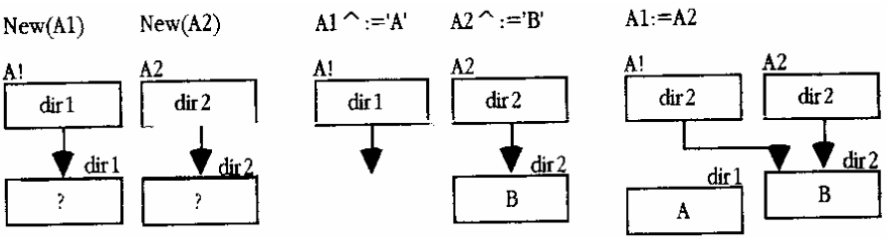
a) `A1 := 1.1;`
b) `A1 := 1.1^;`
c) `New(A1);`
d) `A1 := NIL;`
e) `A1^ := 1.1;`
f) `New(A1^);`
g) `A2:= A1;`
h) `A2:=^1.1;`
i) `A2:=^A1;`

Solución: Legal: 6*c*, 6*d*, 6*e* (legal si `A1^` existe), 6*g*; Ilegal: 6*a*, 6*b*, 6*f*, 6*h*, 6*i*.

7. ¿Qué exhibe el siguiente programa?

```
Program QuePasa;
type
    ApuntadorC = ^Char;
var
    A1, A2: ApuntadorC;
begin
    New(A1);
    New(A2);
    A1^:='A';
    A2^:='B';
    A1:= A2;
    writeln(A1^);
    writeln(A2^);
end;
```

SOLUCION



Con `write(A1^)` y `write(A2^)` se imprimen por pantalla los contenidos de las variables apuntadas por A1 y A2, es decir, dos B. Fíjese en que la variable apuntada por A1 no deja de existir, pero tampoco es accesible desde ningún sitio. Lo más lógico habría sido hacer un `Dispose(A1)` antes de la asignación de punteros si la variable `A1^` ya no interesaba.

8. Dadas las siguientes definiciones y declaraciones

```
type
    TPEntero = ^integer;
    TPCaracter = ^char;
var
    P1, P2 : TPEntero;
    Q1, Q2 : TPcaracter ;
```

¿cuál es la salida de los siguientes fragmentos de código?

a)

```
new(P1);
new(P2);
new(Q1);
readln(Q1^);
P2^:= P1;
writeln('Q1^ igual a ', Q1^, ' Q2^ igual a ', Q2^);
```

Solución: El programa no compila. No está permitida la asignación `P2^:=P1` puesto que no es posible asignar a un tipo *integer* el tipo puntero *TPEntero*.
`Error: Incompatible types: got TPENTERO expected SMALLINT`

b)

```
new(P1);
P1:= P2;
P1:= 3.5 * P1^;
```

Solución: La primera asignación es válida, pero no es correcta, puesto que el puntero P2 está sin definir. Al llegar a la segunda asignación el compilador generar un error de tipos: se espera un tipo entero, pero entiende que `3.5*P1^` devuelve un tipo real.
`Error: Incompatible types: got S80REAL expected TPENTERO`
En cualquier caso, la asignación no es válida porque no están permitidas operaciones aritméticas con punteros.

c)

```
new(P1);
new(Q2);
P1^:= 48;
Q2^:= char(P1^);
P1:= Q2;
```

Solución: Existe un error de compilación en la sentencia `P1:=Q2;` puesto que se espera un tipo *TPCaracter* y se obtiene un tipo *TPEntero*
`Error: Incompatible types: got TPCARACTER expected TPENTERO`

9. Asumir las siguientes declaraciones:

```
var
    x: integer;
    P1, P2: ^integer;
    Q1, Q2: ^real;
```

¿Qué es incorrecto (si lo hay) en cada una de las sentencias?

a) `writeln(P1);`

Solución: Error de compilación: las variables de tipo puntero no se "leen" o "escriben" en sentencias de este tipo.
`Error: Can't read or write variables of this type`

b) `P1:= Q1;`

Solución: Error de compilación: las variables son de tipos distintos.
`Error: Incompatible types: got ^S64REAL expected ^SMALLINT`

c) `if P1^ = nil then Q1:= Q2;`

Solución: `P1` es una variable de tipo puntero; no así `P1^`, la variable dinámica asociada, de tipo entero. La comparación no se puede realizar.

d) `readln(P1^)`

Solución: Correcto.

e) `new(X)`

Solución: X no es una variable de tipo puntero.
`Error: pointer type expected, but got SMALLINT`

f)

```
begin
    P1^:= 17;
    new(P1);
end
```

Solución: No hay error de compilación, sino de ejecución puesto que no se ha hecho reserva alguna para el puntero P1, y su valor es indefinido.

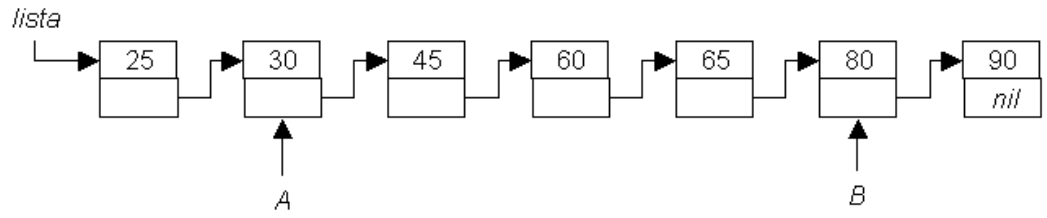
10. ¿Cuál es la salida del siguiente trozo de código?

```
program punteros;
type
    estudiante = record
        Letra: char;
        Edad: integer;
    end;
    punt_estu = ^estudiante;
var
    p1, p2: punt_estu;
begin
    new(p1);
    p1^.edad:=10;
    p1^.letra:='A';
    new(p2);
    p2^.edad:=11;
    p2^.letra:='C';
    writeln (p1^.letra, p1^.edad, p2^.letra, p2^.edad);
    p1^.edad:= p2^.edad;
    p1^.letra:= p2^.letra;
    writeln (p1^.letra, p1^.edad, p2^.letra, p2^.edad);
    p1:=p2; {ojo, esto es lícito, pero deja a p1^sin liberar}
    writeln (p1^.letra, p1^.edad, p2^.letra, p2^.edad);
end.
```

Solución:

```
A10C11
C11C11
C11C11
```

11. Sea la siguiente figura que representa una lista:



y sean:

```
type
    tInfo = integer;
    tEnlace = ^tnodo;
    tNodo = record
        info: tInfo;
        sig: tEnlace;
    end;
    tColeccion = tEnlace;
var
    lista: tColeccion;
    A, B: tEnlace;
```

a) Dar los valores de las siguientes expresiones:

1) `A^.info`

Solución: 30

2) `B^.sig^.info`

Solución: 90

3) `Lista^.sig^.sig^.info`

Solución: 45

b) Decir si se verifican las siguientes igualdades:

1) `Lista^.sig = A`

Solución: Verdadero

2) `A^.sig^.info = 60`

Solución: Falso

3) `B^.sig = nil`

Solución: Falso

c) Indicar si la sintaxis de las siguientes sentencias son correctas o no, y explicar cuál es el problema, si lo hay.

1) `Lista^.sig : = A^.sig`

Solución: Correcto, pero perderíamos elementos de la colección.

2) `Lista^.sig := B`

Solución: Correcto, pero perderíamos elementos de la colección.

3) `Lista^.info := B`

Solución: Incorrecto, se espera un tipo tInfo (entero) y se intenta asignar un tipo tPuntero.

4) `B := A^.sig^.info`

Solución: Incorrecto, se espera un tipo tPuntero y se intenta asignar un tipo tInfo (entero) .

5) `Lista:=B^.sig^.sig`

Solución: Correcto. Se asigna `nil` (puntero nulo) a `Lista`

6) `B := B^.sig^.sig^.sig` (nil no tiene siguiente)

Solución: Incorrecto. No existe la variable dinámica `B^.sig^.sig^`

d) Escribir una sentencia para cada una de las siguientes acciones:

1) Hacer que `Lista` apunte al nodo que contiene `45`

Solución: `Lista:=Aˆ.sig;`

2) Hacer que `B` apunte al último nodo de la lista

Solución: `B:= Bˆ.sig;`

3) Hacer que `Lista` apunte a una lista vacía

Solución: `List:= nil;`

e) Mostrar lo que escribe el siguiente segmento de código:

```
var
    Ptr : tEnlace;
...
New(lista);
New(Ptr);
Listaˆ.info:=2;
Ptrˆ.info:=5;
Lista:=Ptr;
Ptrˆ.info:=7;
writeln(Ptrˆ.info, Listaˆ.info);
```

Solución: 77

f) Mostrar lo que escribe el siguiente segmento de código:

```
var
   Ptr : tEnlace;
...
New(lista);
Lista^.info := 10;
New(Ptr);
Ptr^.info := 18;
Ptr^.sig := nil;
Lista^.sig := Ptr;
New(Ptr);
Ptr^.info := 20;
Ptr^.sig:=lista;
Lista := Ptr;
while Ptr <> NIL do begin
   writeln(Ptr^.info);
   Ptr := Ptr^.sig;
end; (*while*)
```

Solución: 20
10
18

12. Dadas las declaraciones siguientes:

```
type
    PteroNumero = ^NodoNumero;
    NodoNumero = record
                    Datos: integer;
                    Sig: PteroNumero;
                 end;
var
    P1, P2: PteroNumero;
    P3: ^integer;
```

y suponiendo que se han ejecutado previamente las instrucciones

```
new(p1); new(p2); new(p3);
```

¿Qué resultado se visualizará en cada uno de los siguientes fragmentos (en caso de error, indicar cuál):

a)

```
p1^.Datos := 123;
p2^.Datos := 456;
p1^.sig := P2;
writeln(p1^.Datos);
writeln (p1^.sig^.Datos);
```

Solución: 123 y 456

b)

```
p1^.Datos := 12;
p2^.Datos := 34;
p1:=p2;
writeln(p1^.Datos);
writeln (P2^.Datos);
```

Solución: `34 y 34`

c)

```
p1^.Datos := 12;
p2^.Datos := 34;
p1^.sig := p2;
writeln(p2^.Datos);
writeln (p2^.sig^.Datos);
```

Solución: `34 e incorrecto porque el puntero no está definido`

d)

```
p1^.Datos := 12;
p2^.Datos := 34;
p3^.Datos := 34;
p1^.sig := p2;
p2^.sig := p3;
writeln(p1^.Datos);
writeln(p2^.Datos);
writeln(p3^.Datos);
```

Solución: `12, 34 y 34`