EXAMENES RESUELTOS DE PROGRAMACION I

Comentarios por Jose Antonio Vaqué

EXAMEN DE septiembre 1994

Soluciones no oficiales

1.- ¿Cuál de las siguientes sentencias es incorrecta?:

```
VAR
  A : ARRAY [0..3] OF ARRAY [1..5] OF CHAR;
  B : ARRAY [2..6] , [0..4] OF CHAR;
…
A[0, 2] := B[3, 1];
…
```

a) La declaración de A.
b) No hay ningún error.
c) La declaración de B.
d) La sentencia de asignación.

Texto base, Tema 11.3. Formaciones anidadas. Matrices.

Las dos definiciones de variables son correctas, según se ve en la página 294, y luego se asigna un elemento válido de B a un elemento válido de A, usados tal y como aparece en la página 295. Por lo tanto, no hay errores.

2.- Dadas las siguientes declaraciones:

```
TYPE Tipo1 = RECORD
  CASE i : INTEGER OF
    -1 : x : INTEGER |
     1 : z : REAL;
  END;
END;
VAR a : Tipo1;
```

Qué valores toman las componentes de a tras ejecutar:

```
a.i := 1;
a.z := 13.0;
a.x := 5;
```

a) a.z indeterminado y a.x vale 5
b) a.z vale 13.0 y a.x vale 5
c) a.z indeterminado y a.x indeterminado
d) a.z vale 13.0 y a.x indeterminado

Texto base, Tema 12.5 Registros con variantes.

Aunque en el texto base no está demasiado claro, las variables que están en la parte variante no comparten las mismas posiciones de memoria, sino que se definen todas una tras otra. Lo que se está definiendo es una estructura igual a esta:

```
TYPE Tipo1 = RECORD
    i : INTEGER ;
    x : INTEGER;
    z : REAL;
  END;
```

El discriminante no es mas que otra variable mas, que se usa para calificar que parte es la que DEBERIA usarse, pero ambas se pueden usar indiscriminadamente.

3.- Una “Lavadora con diferentes programas de lavado” es una máquina:

a) Virtual.
b) Automática.
c) De control manual.
d) <u>Programable.</u>

Texto base, Tema 1.1.1., Máquinas programables, página 4:
“Las máquinas automáticas actúan por si solas, sin necesidad de operador, aunque pueden responder a estímulos externos…Otras máquinas automáticas se denominan programables, y su comportamiento no siempre es el mismo.”

Primero decir que esto va a juicio del equipo docente, y no tiene mucha lógica en el examen, ya que su definición de máquina programable no es muy buena, incluso diría que es muy mala.

Por un lado, a la definición de máquina automática le sigue el ejemplo de un ascensor, que responde a los botones que se le pulsan, de forma mecánica y prefijada. La lavadora también responder de forma mecánica y prefijada a la selección del “programa de lavado” que elijamos, y la confusión puede venir del término “programa de lavado”, que no tiene nada que ver con un programa de ordenador.

Pero ponen como ejemplo de máquina programable un tocadiscos, ya que se puede elegir el orden de los discos cuando queramos. Por eso la lavadora debe ser también una máquina programable, ya que podemos elegir el programa que deseemos en la lavadora cada vez.

4.- Dado el siguiente procedimiento:

```
PROCEDURE Calcular(X : INTEGER; VAR B : INTEGER);
BEGIN
  B := X DIV 2;
  X := X * X + 3 * B + 1;
END Calcular;
```

El valor de Z tras ejecutar:

```
Z := 5;
Calcular(Z, Z);
```

será:

a) 32
b) <u>2</u>
c) 5
d) Ninguna respuesta anterior es correcta

Texto base, Tema 7.4.2. Paso de argumentos por referencia.

Cuando entramos en el procedimiento, el valor de X se toma por valor, mientras que el de B se toma por referencia. Cualquier cambio que hagamos en B se reflejará al salir del procedimiento. Lo primero que se hace es B:= X DIV 2, y ese valor se modifica en la variable en la que se efectúa la llamada. Por eso Z pasa a valer 2. Lo que luego se hace con X ya no influye para nada.

5.- Dados los siguientes módulos de programa:

```
DEFINITION MODULE Modulo1;
  TYPE Tipo1;
  PROCEDURE Leer;
END Modulo1.
IMPLEMENTATION MODULE Modulo1;
  PROCEDURE Leer;
    VAR cadena : Tipo1;
  BEGIN
    cadena := “datos”;
  END Leer;
END Modulo1.
```

a) <u>La definición del Tipo1 es incorrecta.</u>
b) La variable cadena deberá estar declarada como ristra global en el módulo principal.
c) La definición completa del Tipo1 se debe dar en el módulo principal.
d) Falta el argumento del procedimiento Leer.

Texto base, Tema 9.1. Definición de tipos. Página 223:
“Declaración_de_tipos ::= TYPE { Definición_de_tipo; }
Definición_de_tipo ::= Identificador = Esquema_de_tipo”

Una definición de tipo, en la notación BNF deja claro que debe ser TYPE nombre = tipo y eso no se cumple en el módulo de definición.

6.- ¿Cuál de las siguientes sentencias no es correcta? (el _ representa un espacio en blanco).

a) WriteString(‘’);
b) Write(‘’);
c) Write(“_”);
d) WriteString(“_”);

Texto base, Tema 2.6.3.Procedimientos Write y WriteString

Tanto las cadenas como los caracteres individuales se pueden encerrar entre comillas simples o dobles. La sentencia WriteString imprime una cadena. La cadena vacía o una cadena de un espacio en blanco son cadenas válidas. La sentencia Write imprime un solo carácter. Un espacio en blanco es un carácter, pero una cadena bacía no contiene ningún carácter en su interior, por lo que no es un carácter válido.

7.- ¿Es correcto el siguiente fragmento de programa?:

```
CONST INTEGER = 10;
NUMERO = INTEGER DIV 3;
VAR NIVEL : INTEGER;
```

a) No, la declaración de la constante INTEGER es incorrecta.
b) No, la declaración de la variable NIVEL es incorrecta.
c) Sí, es correcta.
d) No, la declaración de la constante NUMERO es incorrecta.

Texto base, Tema 3.2 El vocabulario de Modula-2.

Un elemento de Modula-2 puede usar como identificador un tipo de dato, ya que no son palabras reservadas del lenguaje. Por ello la redefinición de INTEGER como constante es correcta. Una constante se puede usar para definir el valor de otra, por ello la definición de NUMERO como constante es correcta (al ir justo tras otra constante, no hay que volver a indicar CONST en su definición). Una variable se define como VAR nombre : tipo_de_dato, pero en este caso INTEGER ya no es un tipo de dato, sino que es una constante, por lo que se puede usar en estas definiciones.

8.- Dado el siguiente fragmento de código:

```
TYPE
  Ptr = POINTER TO Dato;
  Dato = RECORD
    x : INTEGER;
    sgte : Ptr;
  END;
VAR a, b : Ptr;
BEGIN
NEW(a);
NEW(b);
a^.x := 7;
b^.x := 13;
b^.sgte := a;
a^.sgte := b;
...
```

El valor de la expresión a^.sgte^.sgte^.x es:

a) La expresión no tiene valor definido.
b) Las asignaciones son incorrectas.
c) 7.
d) 13.

Texto base, Tema 13.3. Variable dinámicas.

Parece complejo, pero es muy sencillo. Si seguimos el programa, vemos que hace lo siguiente:

TYPE Ptr = POINTER TO Dato; Dato = RECORD x : INTEGER; sgte : Ptr; END;	se define una estructura que contiene un entero, y un puntero a si misma, para que un elemento pueda apuntar a otro y realizar una estructura encadenada de registros.
VAR a, b : Ptr; BEGIN	Se definen dos punteros con esta estructura
NEW(a); NEW(b);	Se crean los dos punteros
a^.x := 7; b^.x := 13;	Se asigna 7 al entero de una variable, y 13 al entero de la segunda
b^.sgte := a; a^.sgte := b;	Se hace que el puntero a siguiente de cada una de las estructuras apunte a la otra
a^.sgte^.sgte^.x	Mediante a^.sgte^ se llama al registro al que apunta a, que es b. Luego se llama al registro que se apunta mediante b^.sgte^, que nuevamente es a. Finalmente queda a.x que es el valor x del registro de la variable a, que es 7.

Para verlo mas sencillo, podemos decir que “a” apunta a un registro que contiene (7, apunto a “b”). Mientras que “b” apunta a un registro que vale (13, apunto a “a”).

9.- El uso de comentarios en un programa es:

a) Necesario para no tener errores de compilación.
b) Conveniente para aumentar las líneas de programa.
c) Imprescindible para documentar el programa.
d) Una pesadez innecesaria.

Texto base, Tema 2.7.1, página 46: “Estos comentarios sirven solo como documentación del programa fuente…”

Pregunta tonta, ya que las otras tres opciones son indiotas.

10.- El valor de RES tras ejecutar el siguiente fragmento será:

```
RES := 1;
FOR i := 1 TO 6 BY 2 DO
  LOOP
    j := 1;
    REPEAT
      RES := RES * j;
      INC( j );
    UNTIL j = 3
  END;
  IF (RES MOD i) = 0 THEN
    EXIT;
  ELSE
    RES := RES DIV i;
  END;
END;
```

a) El uso del REPEAT no es correcto.
b) 0.
c) <u>El uso del EXIT no es correcto.</u>
d) 27.

Texto base, Tema 10.1.2, Sentencias LOOP y EXIT

El bucle LOOP no contiene ningún EXIT en su interior, sino tras el END (no confundir con el END del REPEAT, ya que esta instrucción no usa el END, sino el UNTIL), y por tanto el EXIT no se usa correctamente, ya que queda fuera del bucle.

EJERCICIO

Realizar en Modula-2 un programa que dados los resultados de los 10 partidos de fútbol de una jornada de liga, actualice la tabla de clasificación, en la que se guarda para los 20 equipos su nombre (compuesto de un carácter) y el número de puntos hasta ese momento.

```
MODULE Examen;
FROM InOut IMPORT Write, WriteString, WriteLn, WriteInt, Read;

TYPE EQUIPO = RECORD       (* Para guardar datos de un equipo *)
     Nombre : CHAR;  Puntos : INTEGER;
     END;
     PARTIDOS = RECORD      (* Para guardar datos de un partido *)
     Local : CHAR;   Visitante : CHAR;   Resultado : CHAR;
     END;

VAR  Tabla : ARRAY [1..20] OF EQUIPO;     (* Tabla de equipos *)
     Partido : ARRAY [1..10] OF PARTIDOS; (* Tabla de partidos de una jornada *)
     Nombres : ARRAY [0..19] OF CHAR;     (* Nombres de los equipos *)
     Local : CHAR;                        (* Equipo local *)
     Visitante : CHAR;                    (* Equipo visitante *)
     Resultado : CHAR;                    (* Resultado del partido *)
     i : INTEGER;       aux : CHAR;       (* Auxiliares *)

(* Busca un equipo y si es correcto lo coloca en su lugar *)
PROCEDURE Buscar(equipo:CHAR;numero,tipo:INTEGER) : BOOLEAN;
VAR j : INTEGER; salida : BOOLEAN;
BEGIN
     salida := FALSE;
     FOR j:=1 TO 20 DO   (* Mira que exista el equipo *)
         IF (equipo = Tabla[j].Nombre) THEN salida := TRUE; END;
     END;
     FOR j:=1 TO 10 DO   (* Mira que no se haya usado ya *)
         IF (equipo = Partido[j].Local)     THEN salida := FALSE; END;
         IF (equipo = Partido[j].Visitante) THEN salida := FALSE; END;
     END;
     IF (salida) THEN
        IF (tipo = 0) THEN Partido[numero].Local:= equipo;     END;
        IF (tipo = 1) THEN Partido[numero].Visitante:= equipo; END;
     END;
     RETURN (salida);
END Buscar;
PROCEDURE Sumar(equipo:CHAR;puntos:INTEGER); (* Sumar los puntos a un equipo *)
VAR j : INTEGER;
BEGIN
     FOR j:=1 TO 20 DO
         IF (equipo = Tabla[j].Nombre) THEN
             Tabla[j].Puntos:=Tabla[j].Puntos + puntos;
          END;
     END;
END Sumar;

BEGIN
     Nombres := "ABCDEFGHIJKLMNOPQRST"; (* Inicializamos los nombres *)
     FOR i:=1 TO 20 DO                  (* Inicializamos la tabla *)
         Tabla[i].Nombre := Nombres[i-1]; Tabla[i].Puntos:=0;
     END;
     LOOP
         REPEAT
            WriteString("Comenzar Nueva jornada (S/N): ");Read(aux);
         UNTIL (aux = 'S') OR (aux = 'N');
         IF (aux = 'N') THEN EXIT; END;
         FOR i:=1 TO 10 DO                  (* Inicializamos los partidos *)
            Tabla[i].Nombre := Nombres[i-1]; Tabla[i].Puntos:=0;
         END;
         FOR i:=1 TO 10 DO
            WriteLn;WriteLn;WriteString("Datos del partido ");WriteInt(i,2);
            WriteLn;
            REPEAT
               WriteString("Equipo Local.....: ");Read(Local);WriteLn;
            UNTIL (Buscar(Local,i,0));

            REPEAT
               WriteString("Equipo Visitante.: ");Read(Visitante);WriteLn;
            UNTIL (Buscar(Visitante,i,1));
            REPEAT
               WriteString("Resultado (1-X-2): ");Read(Resultado);WriteLn;
            UNTIL (Resultado = '1') OR (Resultado = 'X') OR (Resultado = '2');
            IF (Resultado = '1') THEN Sumar(Local,3); END;
            IF (Resultado = 'X') THEN Sumar(Local,1); Sumar(Visitante,1); END;
            IF (Resultado = '2') THEN Sumar(Visitante,3); END;
         END;
     END;
END Examen.
```

No es nada elegante, ya que le faltan algunas verificaciones, pero funciona. Primero crea una tabla con los 20 equipos, sus nombres y los puntos a cero. Necesita usar una cadena auxiliar con los nombres, para ocupar solo una línea, aunque deberíamos asumir que ya existen los equipos.

Luego para cada jornada, inicializa un vector con la estructura de la jornada, que usará luego para ir guardándose los resultados de la jornada.

Va pidiendo nombres de los equipos local y visitante, verificando que los equipos existan, y que no se hayan usado en la misma jornada. Luego pide el resultado, y suma los puntos a cada equipo.

No hay sitio para mas en una hoja, ni para imprimir la tabla de resultados al final. Como mejoras, habría que considerar:

a) Poder meter datos en mayúsculas o minusculas.
b) Nombres de equipos completos, no de un solo carácter.
c) Guardarse todas las jornadas del campeonato.
d) Verificar que no se repitan jornadas una vez introducidas
e) Imprimir las tablas clasificatoria de cada jornada y la final, ordenadas por los puntos de cada equipo.