EXAMENES RESUELTOS DE PROGRAMACION I

Comentarios por Jose Antonio Vaqué

EXAMEN DE Septiembre de 2000

SOLUCIONES OFICIALES

1 Nos encontramos con estas sentencias

```
dato := 3;
dato := 4.3;
```

¿Qué estamos manejando?

A.- Dos campos variantes con el mismo identificador
B.- Un error monumental
C.- Una variable multitipo
D.- Una redefinición

Se está intentando asignar a una variable un datos de tipo INTEGER o CARDINAL, y luego uno de tipo REAL, lo que no es posible en Modula.

2 El dato "hola" es

A.- Una constante cuyo identificador es "hola"
B.- Es un ARRAY de CHAR
C.- Es un valor constante
D.- Es un CHAR

Al estar entre comillas, sabemos que es un valor constante de tipo cadena, no tiene nada que ver con el identificador (el nombre que se le da a la variable), ni es un ARRAY ni un CHAR.

3 Dado el siguiente fragmento de código

```
DEFINITION MODULE Juegos
PROCEDURE InicioJuego;
PROCEDURE FinJuego;
END Juegos.
```

InicioJuego y FinJuego manipulan la variable jugadores.
¿Dónde debe inicializarse esta variable?

A.- En las sentencias de inicialización del módulo de implementación
B.- En las sentencias de inicialización del modulo de definición
C.- En el bloque ejecutivo de los procedimientos
D.- En la parte ejecutiva de los módulos que los usen

Cualquier variable de un módulo que no esté declarada en el DEFINITION, solo se puede usar dentro del IMPLEMENTATION. El lugar adecuado para su inicialización es en la parte de inicialización del módulo. La respuesta B es absurda, ya que no se puede hacer nada en un DEFINITION. La respuesta C puede parecer adecuada, pero si miramos, la variable se usa en los dos módulos, por lo que debe inicializarse en otro lugar. La respuesta D tampoco es adecuada, ya que en ese caso se podría usar antes de su correcta inicialización.

4 Dado el siguiente fragmento de código

```
VAR A:INTEGER
PROCEDURE A( ):INTEGER;
VAR A: INTEGER;
BEGIN
RETURN A;
END A;
```

A.- Error, identificador reutilizado
B.- Correcto, es una redefinición de identificadores
C.- Correcto, la reglas de visibilidad deciden
D.- Error por paralaje de identificadores

Pregunta con trampa gorda. La variable A global y la que se usa dentro de la PROCEDURE no colisionan, se aplican las reglas de visibilidad y arreglado, pero no se pueden tener dos variables iguales dentro del mismo bloque, y el Modula considera a las funciones como variables (para poder pasarlas como argumentos en procedimientos, lo que no se explica en el libro). Yo no he visto en el libro que se mencione que las funciones y las variables no pueden tener el mismo nombre, peor el compilador no deja, por lo que asumo será así.

5 La sentencia de asignación:

A.- Siempre necesita una variable a su derecha
B.- Existe en todos los lenguajes de programación
C.- Siempre necesita una variable a su izquierda
D.- Existe en todos los modelos de cómputo

Esta es fácil, si ponemos cosa := valor; podemos poner en valor otra variable, una constante, una función, o una expresión, pero cosa siempre debe ser una variable. Aunque parezca raro, no tiene porque existir en todos los lenguajes (los hay muy raros), y la respuesta D no tiene mucho sentido.

6 Los aspectos de estilo:

A.- Muestran la creatividad de cada programador
B.- No dependen del lenguaje utilizado
C.- Sólo tiene en cuenta el encolumnado y los comentarios
D.- Los fija el lenguaje utilizado

Texto base, apartado 4.4 aspectos de estilo, página 80 y siguientes..

Aunque parece que A sea la elección, lo que demuestran es la disciplina del mismo, de fijar unos criterios y seguirlos. Además, suelen venir impuestos por la costumbre general o por los propios jefes.

7 ¿Cual de las siguientes afirmaciones es CORRECTA?:

A.- Los tipos opacos son tipos abstractos de datos
B.- Los datos encapsulados son tipos opacos
C.- Los tipos anónimos son datos encapsulados
D.- Los tipos abstractos son tipos opacos

Texto base, apartado 14.3 Tipos abstractos de datos, página 410 y siguientes.
Texto base, apartado 14.4.2 Tipos opacos, página 414 y siguientes.
Texto base, apartado 14.4.3 Datos encapsulados, página 416 y siguientes.

Descaramos la B, ya que un dato encapsulado no tiene tipo de dato asociado. Descartamos C, ya que un tipo anómino no tiene nada que ver con esto. Descartamos D, ya que puedo definir tipos abstractos que no sean opacos.

8 Respecto a la sentencia:
P^ := P^.siguiente DIV 23;
Se puede decir:

A.- Hay incompatibilidad de tipos en la operación DIV
B.- Es correcta
C.- No es correcta la operación P^.siguiente
D.- Hay incompatibilidad de tipos en la asignación

Un puntero solo puede recibir una dirección, no se le puede asignar un valor.

9 A descomponer un problema en subproblemas se conoce como

A.- Reutilización
B.- Redefinición
C.- Refinamiento
D.- Reasignación

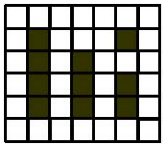
Reutilizar es volver a utilizar, redefinir es volver a definir, reasignar es volver a asignar. Aunque no hayamos leído el libro, solo quedaba esta.

10 La complejidad de un algoritmo:

A.- Depende del programador
B.- Depende del anidamiento de bucles
C.- Depende del invariante
D.- Aumenta con el uso del LOOP

Aunque la complejidad depende del programador, los noveles suelen hacer programas más complejos y los veteranos solemos hacerlos mas simples, realmente depende mas del anidamiento de bucles y las llamadas a procedimientos que de otra cosa.

EJERCICIO DE PROGRAMACIÓN



Desarrollar en MODULA - 2 en un módulo distinto del principal el juego de los “barcos”. Distribuir en un tablero de tamaño 6 por 6 cuadrículas, cuatro barcos de tamaños 4, 3, 2, y 1 cuadrícula. No pueden estar en contacto entre sí ni con los laterales del tablero. Desarrollar el dato tablero con las operaciones: BuscarSitio(tablero, tamano, cuadriculas), SituarBarco(tablero, tamano, cuadriculas) Los barcos son longitudinales y continuos. En la figura se puede ver un ejemplo de distribución de barcos. Crear otra distribución.

Auque el enunciado te dicen un tablero de 6x6, el ejemplo es de 7x6, ya que un tablero de 6x6 no se puede meter todos los barcos sin tocar los bordes o tocarse entre si, por lo que el tablero debe ser de 7x7 obligatoriamente.

```
DEFINITION MODULE Pers;
   CONST LMAX = 25;                      (* Longitud máxima de un dato *)
   TYPE TBuscar = (BNombre, BTelefono);  (* Tipos de búsquedas *)
        Persona = RECORD                 (* Para guardar una persona *)
           nom, ap1, ap2, dir, tel : ARRAY [0..LMAX] OF CHAR;
        END;
   PROCEDURE Alta(dato:Persona);
   PROCEDURE Baja(reg:INTEGER);
   PROCEDURE Buscar(tipo:TBuscar;que:ARRAY OF CHAR;VAR reg:INTEGER):BOOLEAN;
END Pers.
```

```
IMPLEMENTATION MODULE Pers;
FROM InOut IMPORT WriteString,WriteInt,WriteLn;
   CONST NMax = 50;                                (* Máximo de personas que guardamos *)
   VAR VPos : INTEGER;                             (* Puntero *)
       VPersona : ARRAY [1 .. NMax] OF Persona;    (* Las personas que manejamos *)
   PROCEDURE Alta(dato:Persona); (* Alta de una persona *)
   BEGIN
      IF (VPos = NMax) THEN
         WriteString("Estructura llena, no puede introducir otro elemento");WriteLn;
      ELSE
         INC(VPos);   VPersona[VPos] := dato;
      END;
   END Alta;
   PROCEDURE Baja(reg:INTEGER); (* BAJA de una persona *)
   VAR i:INTEGER;
   BEGIN
      IF (reg < 1) OR (reg > VPos)  THEN
         WriteString("Elemento fuera de rango, no se puede borrar");WriteLn;
      ELSE
         FOR i:=reg TO VPos-1 DO
            VPersona[i] := VPersona[i+1];
         END;
         DEC(VPos);
      END;
   END Baja;
   PROCEDURE Buscar(tipo:TBuscar;que:ARRAY OF CHAR):BOOLEAN; (* Buscar una persona *)
   VAR i:INTEGER;  e:BOOLEAN;
   BEGIN
      i:=0;  e:=FALSE;
      REPEAT
         INC(i);
         CASE tipo OF
            BNombre   : e:= Comparar(que,VPersona[i].nom); |
            BTelefono : e:= Comparar(que,VPersona[i].tel);
         END;
      UNTIL (e = TRUE) OR (i = VPos);
      IF (e = TRUE) THEN
         WriteString("Encontrado en posición: ");WriteInt(i,2); WriteLn;
         WriteString("   Nombre....: ");WriteString(VPersona[i].nom);WriteLn;
         WriteString("   Apellido 1: ");WriteString(VPersona[i].ap1);WriteLn;
         WriteString("   Apellido 2: ");WriteString(VPersona[i].ap2);WriteLn;
         WriteString("   Dirección.: ");WriteString(VPersona[i].dir);WriteLn;
         WriteString("   Teléfono..: ");WriteString(VPersona[i].tel);WriteLn;
      END;
      RETURN e;
   END Buscar;
   PROCEDURE Comparar(uno,dos:ARRAY OF CHAR):BOOLEAN; (*Comparar dos cadenas *)
   VAR i,max1,max2:INTEGER;  e:BOOLEAN;
   BEGIN
      max1 := HIGH(uno);   max2 := HIGH(dos);   e := TRUE;   i:= 0;
      LOOP
         IF (uno[i] <> dos[i]) THEN   e := FALSE;   EXIT;   END;
         INC(i);
         IF (i > max1) OR (i > max2) THEN   EXIT;   END;
      END;
      RETURN e;
   END Comparar;
BEGIN
   VPos := 0; (* Inicializamos el puntero a cero *)
END Pers.
```