EXAMENES RESUELTOS DE PROGRAMACION I

Comentarios por Jose Antonio Vaqué

EXAMEN DE Febrero 1994 - 2ª Semana

Soluciones no oficiales

1.- Dado el subprograma:

```
PROCEDURE Potencia(Z : REAL; exponente : INTEGER) : REAL;
VAR
  C : REAL;
  K : INTEGER;
BEGIN
  C := 1.0;
  FOR K := 1 TO exponente DO
    C := C * Z;
  END;
  RETURN C;
END Potencia;
```

¿Cuál sería una posible sentencia de llamada a este subprograma?

a) Potencia(3.0, 2);
b) IF (Potencia(7.2, 2) < 25.) THEN …
c) Valor := Potencia(3.0, 2.);
d) Ninguna respuesta anterior es correcta.

Texto base, Tema 7. Funciones y Procedimientos

El apartado a) no es correcto, ya que no recoger el retorno de la función. El apartado b) es válido, ya que se compara el retorno de una función, que es una REAL en este caso, con otro REAL. El apartado c) no es correcto, ya que la llamada se realiza con dos REALES, cuando se piden REAL y ENTERO. El apartado d) no es correcto, ya que si hay respuesta.

2.- Dado el siguiente fragmento de programa con:

```
VAR
  i : INTEGER;
  vector : ARRAY [1..10] OF INTEGER;
…
FOR i := 10 TO 1 BY -1 DO
  IF vector[i] = vector[i-1] THEN
    vector[i] := 2 * vector[i];
  END;
END;
```

a) Hay errores de compilación.
b) Se producen errores en la ejecución del programa.
c) Todos los elementos del vector terminan siendo diferentes.
d) Ninguna respuesta anterior es correcta.

Texto base, Tema 11.2. Vectores

El programa define un vector de 10 elementos, y lo recorre mediante un bucle del 10 al 1. Hasta aquí todo correcto. Para cada recorrido del bucle, compara el contenido de un elemento con el anterior. Podemos suponer que se han asignado valores a los elementos del vector, o pensar que no tiene valor, y es indefinido. La cuestión es que cuando i valga 1, compararemos el elemento [1] con el elemento [0], que no está definido en el vector, por lo que se produce un error de ejecución.

3.- Dado el módulo:

```
DEFINITION MODULE Total;
  PROCEDURE Suma;
  PROCEDURE Media;
END Total.
```

a) Faltan los argumentos de los procedimientos.
b) Es completamente correcto.
c) Falta la declaración de algún tipo o variable.
d) Debe acabar en un punto y coma.

Texto base, Tema 14.2. Módulos en Modula-2

El módulo de definición solo puede contener definición de variables y de procedimientos, no su implementación, para ello están los módulos de implementación. Por ello este es un módulo de definición perfectamente válido.

4.- Dada la declaración de variables:

```
VAR
  dias, meses, annos : INTEGER;
  tipo, codigo : CHAR;
  pesetas, sueldo, gastos : REAL;
```

cual de las expresiones siguientes NO es correcta:

a) pesetas := (sueldo - gastos) * meses;
b) tipo := CHR(ORD(codigo));
c) dias := annos * 365 + meses * 30 + dias;
d) pesetas := sueldo * FLOAT(meses);

Texto base, Tema 3.5.1. Compatibilidad de tipos

El apartado a) intenta mezclar variables INTEGER y REAL. Los apartados b) y c) usan variables del mismo tipo. El apartado d) usa variables REAL, y una INTEGER convertida a REAL.

5.- Dada la declaración de variables:

```
VAR
  m, n : INTEGER;
  x, y : REAL;
  l, t : CHAR;
```

cual de las siguientes expresiones condicionales NO es correcta:

a) x = y + FLOAT(10)
b) m # 5 * n
c) x > ln(y)
d) l OR t

Texto base, Tema 5.2. Expresiones condicionales

En principio parece que todos son correctos, pero analizando bien los apartados:

a) Compara un valor REAL x, con otro REAL, resultado de sumar al REAL y el valor 10 convertido a REAL. O sea se compara si (REAL = REAL).
b) Compara si un INTEGER m, es diferente (#) a un INTEGER resultado de multiplicar 5 por un INTEGER. O sea se compara (INTEGER <> INTEGER).
c) Compara si un REAL x, es mayor que otro REAL resultado del logaritmo de un REAL y. O sea se compara si (REAL = REAL).
d) Una operación lógica OR, solo es válida si se opera entre dos valores BOOLEANOS, mientras que aquí se usan valores CHAR.

6.- La sentencia CASE es válida si el tipo del dato que determina la selección es:

a) ordinal, enumerado o subrango.
b) real, entero o carácter.
c) ordinal, real o enumerado.
d) booleana, ordinal o conjunto.

Texto base, Tema 10.2.1. Sentencia CASE, página 262:

"Si el tipo de valor que determina la selección es un tipo ordinal: INTEGER, CARDINAL, CHAR, enumerado o subrango, se dispone en MODULA-2 de la sentencia CASE."

No se puede añadir nada a esto.

7.- Dentro de una función:

a) Puede existir más de una sentencia RETURN.
b) La sentencia RETURN se utiliza para devolver resultados erróneos.
c) La sentencia RETURN es la última línea de código de una función.
d) Ninguna respuesta anterior es correcta.

Texto base, Tema 72.1.Definición de funciones. Página 162:

"La sentencia RETURN se puede insertar en cualquier punto de la parte ejecutable de la función. Además, es posible utilizar mas de una sentencia RETURN en una misma función."

Nuevamente no se puede añadir nada a esto.

8.- ¿Qué hace el siguiente bucle?:

```
FOR k := 1 TO N - 1 DO
  Write('');
  WriteLn;
END;
```

a) Imprimir una línea de apóstrofos.
b) Imprimir N - 1 líneas con un apóstrofo cada una.
c) Imprimir N - 1 apóstrofos seguidos.
d) <u>Da error de compilación.</u>

Texto base, Tema 2.3.3. Caracteres. Página 32:
"el carácter apóstrofo solo se puede representar entre comillas, y viceversa".

Pequeña trampa, ya que parece que la respuesta es la b). Hay que estar muy atentos a este tipo de "trampas" en el examen, semejante a la de la anterior pregunta 2.

9.- Dado el siguiente procedimiento:

```
PROCEDURE Operar(VAR a, b : INTEGER; c : INTEGER);
BEGIN
  a := a + b;
  c := a;
END Operar;
```

y las variables globales X, Y, Z. Con la llamada Operar(X, Y, Z) …

a) Las variables X e Y cambian pero Z no cambia.
b) Las tres variables X, Y y Z cambian.
c) <u>La variable X cambia, pero Y y Z no cambian.</u>
d) Las variables X y Z cambian, pero Y no cambia.

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

Aunque en el texto no está nada claro, en el ejemplo de la página 172 si deja claro que VAR afecta a todas las variables que la sigan, no solo a la primera, por lo que afecta a las variables **a** y **b** del procedimiento. Esto puede despistarnos, pensando que cambian **X** e **Y**, mientras que **Z** no se cambiaría. Pero si analizamos el código, vemos que solo se asigna valor a las variables **a** (pasada por referencia) y **c** (pasada por valor), por lo que realmente solo cambiará **X**. Nuevamente una pequeña "trampa".

10.- Cual es el resultado de la expresión:

```
FLOAT(TRUNC(FLOAT(33 DIV 5 + 2) / 3.) DIV 3) + 1
```

a) 3.6
b) <u>Hay algún error en la expresión</u>
c) 1.0
d) 2.0

Texto base, Tema 2.5. Expresiones aritméticas

Analizando paso a paso, comenzando por los paréntesis mas interiores y operando, tenemos:

FLOAT(TRUNC(FLOAT(**<u>33 DIV 5 + 2</u>**) / 3.) DIV 3) + 1	-> 33 DIV 5 + 2 -> (33 DIV 5) + 2 -> 6 + 2 = 8
FLOAT(TRUNC(**<u>FLOAT(8) / 3.</u>**) DIV 3) + 1	-> FLOAT(8) / 3.0 -> 8.0 / 3.0 = 2.666666
FLOAT(**<u>TRUNC(2.66666.) DIV 3</u>**) + 1	-> TRUNC(2.66666.) DIV 3 -> 2 / 3 = 0
<u>FLOAT(0) + 1</u>	-> FLOAT(0) + 1 -> 0.0 + 1

No se pueden sumar un REAL y un INTEGER, por lo que hay un error. También es mas sencillo si vemos que FLOAT(algo) mas 1 es un REAL mas un INTEGER.

11.- Dadas las siguientes declaraciones:

```
TYPE
  TipoMaterial = (lapiz, goma, clip, anuario, carpeta);
  Escritorio = SET OF TipoMaterial;
VAR MesaOficina : Escritorio;
```

la sentencia: MesaOficina := Escritorio{carpeta..lapiz}

a) <u>Asigna el conjunto vacío a la variable MesaOficina.</u>
b) Asigna todos los elementos del conjunto a la variable MesaOficina.
c) Asigna los elementos carpeta y lapiz a la variable MesaOficina.
d) Ninguna respuesta anterior es correcta.

Texto base, Tema 9.6.Conjuntos. Página 235:
"En el caso de que el ordinal del primer elemento sea superior al del último, el rango que se está indicando es un rango vacío."

Como carpeta es superior a lápiz en la enumeración, el resultado es un conjunto vació, tal y como se indica en el ejemplo que sigue a la definición anterior.

12.- La complejidad del siguiente fragmento de programa:

```
n := 1;
WHILE n < 100 DO
  IF (n MOD 10) = 0 THEN
```

```
    n := 3 * n;
  ELSIF (n MOD 5) = 0 THEN
    n := 2 * n;
  ELSE
    n := n + 1;
  END;
END;
```

a) Es O(1).
b) Es O(n log n).
c) <u>Es O(n).</u>
d) Ninguna respuesta anterior es correcta.

Texto base, Tema 6.4.3. Crecimiento asintótico

A pesar de que es materia de la asignatura PROGRAMACION II, en el libro se da una pequeña introducción, que no creo sirva para responder a esta pregunta, pero en líneas generales, como el número de pasos de este programa depende del número de veces que se ejecute el bucle WHILE, ya que en su interior no se ejecuta ningún otro bucle, podemos decir que es del orden de n.

13.- Dado el siguiente esquema:

```
Iniciar;
WHILE (NOT Encontrado) AND (NOT Final) DO
  Pasar al siguiente elemento;
  Verificar encuentro;
END;
```

a) Es un esquema de inserción.
b) Es un esquema de recorrido.
c) Es una simplificación de las condiciones de contorno.
d) <u>Ninguna respuesta anterior es correcta.</u>

Texto base, Tema 11.4.2.Búsqueda secuencias. Página 298

Nuevamente es un criterio de equipo docente que esto sea claramente un esquema de BUSQUEDA SECUENCIAL, a pesar de que es también un esquema de recorrido. Solución, aprenderse estos temas de memoria.

4.- Dado el siguiente fragmento de programa:

```
A := 7;
B := 15;
C := INC(A + B) + DEC(A);
```

el valor que toma C es:

a) 29
b) 27
c) 30
d) Ninguna respuesta anterior es correcta.

Texto base, Tema 7.3.3. Procedimientos predefinidos

El operador INC no es una función, sino un procedimiento, por lo que no tiene valor de retorno que se pueda asignar a una variable. Además, el valor a incrementar se pasa por referencia, por lo que no se puede incrementar la suma de dos variables, solo una única variable. Por esto dará un error de compilación.

15.- Indicar que conjunto de sentencias NO es correcto, para la siguiente declaración:

```
TYPE puntero = POINTER TO REAL;
VAR p1, p2 : puntero;
```

a) NEW(p1); p2 := p1;
b) <u>NEW(p2); p2^ := NIL; p1^ := p2^;</u>
c) NEW(p1); p1^ := 7.5; p2 := p1;
d) Todas las respuestas anteriores son correctas.

Texto base, Tema 13.3.1. Punteros

Aunque en el texto no está muy claro, el valor NIL solo se puede usar para averiguar si un puntero apunta a algo o no, y no es posible asignar ese valor a un puntero. Para ello se usa DISPOSE o DEALLOCATE. Así puede usarse en sentencias como IF (p2 = NIL) THEN, pero no en p2:=NIL o en p2^:=NIL

EJERCICIO

Escribir un programa en Modula-2 que controle las operaciones diarias de Caja de un establecimiento, con las siguientes características:

- Al comenzar el día la cajera introducirá la fecha.
- Por cada cliente se introducirá código de artículo, cantidad y precio unitario, tantas veces como compras distintas haga el cliente. Para acabar con un cliente se indica mediante código de artículo igual a 0.
- Se imprimirá un ticket con la información: número de ticket (secuencial cada día a partir del número 1), una línea por cada artículo (código, cantidad, precio unitario, precio total), una línea con el IVA (15%) y un total general.
- El final del día se indicará con un código de artículo negativo y a continuación se imprimirá la fecha y el total bruto recaudado.

NOTA: Se puede suponer que las sentencias para la escritura de los tickets escriben directamente en la impresora y no en la pantalla.

```
MODULE Examen;
FROM InOut IMPORT Write, WriteString, WriteLn, ReadInt, WriteInt;
FROM RealInOut IMPORT ReadReal,WriteReal;

VAR  Dia, Mes, Anio : INTEGER;
     Codigo, Cantidad, Precio : INTEGER;
     Numero, TLin, Suma, Total : INTEGER;
     Nuevo : BOOLEAN;
(* Imprimer la cabecera en impresora *)
PROCEDURE cabecera(numero,dia,mes,anio:INTEGER);
BEGIN
     WriteString("** Ticket número: ");WriteInt(numero,5);WriteString("  Fecha: ");
     WriteInt(dia,2);Write('.');WriteInt(mes,2);Write('.');WriteInt(anio,4);WriteLn;
     WriteString("**");WriteLn;
     WriteString("**Codigo Cantidad Precio Total");WriteLn;
END cabecera;
(* Imprime una linea en la impresora *)
PROCEDURE linea(codigo,cantidad,precio:INTEGER ;VAR total,suma:INTEGER);
BEGIN
     total:=cantidad * precio;  suma:= suma + total;
     WriteString("**");WriteInt(codigo,6);WriteInt(cantidad,9);
     WriteInt(precio,7);WriteInt(total,7);WriteLn;
END linea;
(* Imprime total del ticket en la impresora *)
PROCEDURE totales(VAR importe, total:INTEGER);
BEGIN
     WriteString("** ------------- Total: ");WriteInt(importe,7);WriteLn;
     total := total + importe;  importe := 0;
END totales;

BEGIN
     REPEAT  (* Primero pedimos la fecha *)
           WriteString("Introduzca el día (1..31): ");ReadInt(Dia);WriteLn;
     UNTIL (Dia >= 1) AND (Dia <= 31);
     REPEAT
           WriteString("Introduzca el mes (1..12): ");ReadInt(Mes);WriteLn;
     UNTIL (Mes >= 1) AND (Mes <= 12);
     REPEAT
           WriteString("Introduzca el año (2003..2005): ");ReadInt(Anio);WriteLn;
     UNTIL (Anio >= 2003) AND (Anio <= 2005);

     Numero:=0; Total:=0; Nuevo:=TRUE;
     LOOP
           IF (Nuevo) THEN WriteLn;WriteString("----- NUEVO TICKET -----");WriteLn; END;
           WriteString("Código de Artículo (0=Fin Ticket / -1=Fin Día): ");
           ReadInt(Codigo);WriteLn;
           IF (Codigo < 0) THEN (* Fin del día *)
              IF (NOT Nuevo) THEN totales(Suma,Total); END; (*Por si hay uno a medias *)
              EXIT;
           ELSIF (Codigo = 0) THEN (* Fin del ticket *)
              totales(Suma,Total); Nuevo:=TRUE;
           ELSE (* Pedir datos del ticket *)
              IF (Nuevo) THEN
                 INC(Numero);  Nuevo:=FALSE; cabecera(Numero,Dia,Mes,Anio);
              END;
              REPEAT
                 WriteString("Introduzca Cantidad: ");ReadInt(Cantidad);WriteLn;
              UNTIL (Cantidad <> 0); (* Las cantidades pueden ser negativas *)
              REPEAT
                 WriteString("Introduzca Precio: ");ReadInt(Precio);WriteLn;
              UNTIL (Precio >= 0); (* Pero el precio solo positivo o sin cargo *)
              linea(Codigo,Cantidad,Precio,TLin,Suma);
         END
     END;
     (* Los totales del día *)
     WriteLn;WriteLn;WriteString("TOTAL DEL DIA: ");WriteInt(Total,10);
END Examen.
```

Este ejercicio es bastante largo, y en el tiempo que dan en el examen, será difícil acordarse de todos los detalles a tener en cuenta, para que funcione mas o menos bien, como verificar que no cierren el día con un ticket abierto.

Para poder separar un poco la impresión del ticket de la salida por pantalla, he creado tres procedimientos que deben realizar la salida por impresora de la cabecera, de las líneas y del total, aunque siguiendo lo que pone en el enunciado, lo hacen por pantalla. Para diferencias estas líneas, las hago comenzar con dos asteriscos.

Utilizo una variable de control denominada “Nuevo”, que me indica si he comenzado o no un nuevo ticket, para que me imprima el cartel en pantalla de ----- NUEVO TICKET ----- cuando comience uno, para que me saque por la impresora la cabecera del ticket antes de comenzar uno, y para que si acabo el día con un ticket abierto, me lo cierre antes totalizándolo.

Como posibilidades adicionales, podríamos desarrollar lo siguiente:

a) Una rutina de impresión de números con ceros por delante, para que imprimiera de forma mejor las fechas, o con puntos de separación de miles, para que imprima las cantidades mas elegantemente.

b) Una rutina que trate las fechas como una variable, mejorar su introducción como 5 o 6 cifras, que el programa monte como dia-mes-año, y verificando que sea una fecha coherente, montando una cadena con el formato DD.MM.AAAA (con el mes en números), o DD.mmm.AAAA (con el mes en letras).

c) Sacar los ticket a un fichero de texto, que simule la impresora, para que no se mezclen las salidas por pantalla y las que van a la impresora.

d) Guardarse todos los tickets, para poder emitirlos juntos al final de día, tal y como exige Hacienda en España.