EXAMENES RESUELTOS DE PROGRAMACION I

Comentarios por Jose Antonio Vaqué

EXAMEN DE septiembre 1994, Reserva

Soluciones no oficiales

1.- Determinar los valores reales válidos en Modula-2:

a) 56.2F-54 y 2.2
b) 5. Y -0.78
c) 4,78 y 2.0E2
d) .234 y 1.

Texto base, Tema 2.4.3. El tipo REAL

En a) se usa F en lugar de E para el exponente. En c) se usa una coma en lugar del punto. En d) un número comienza por punto, cuando debe comenzar por un número.

2.- Dado el procedimiento:

```
PROCEDURE Uno(VAR x, y, z : INTEGER);
BEGIN
  IF x < y THEN
    z := y - x;
    IF x < z THEN
      y := z - x;
    END;
  END;
END Uno;
```

el resultado después de las siguientes sentencias:

```
A := 1;
B := 2;
Uno(A, A, B);
```

será:

a) A = 1; B = 2
b) A = 1; B = 1
c) Hay errores en la declaración del procedimiento
d) Hay errores en la llamada al procedimiento

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

Como las variables que acepta el procedimiento son por valor, no por referencia, no son afectadas por el proceso en su interior, y mantienen sus valores (además, aunque se pasasen por referencia, en el interior no se les altera, ya que 1 no es menor que 1, y no se ejecuta lo de dentro de los IF).

3.- Dado el siguiente fragmento de código:

```
VAR
  p1, p2 : POINTER TO REAL;
  numero : INTEGER;
…
NEW(p1);
NEW(p2);
p1^ := 33.3;
p2^ := 34.5;
p1 := p2;
numero := p1^;
…
```

a) La variable numero toma el valor 33.5
b) La variable numero toma el valor 34.5
c) La variable numero toma el valor 33
d) Ninguna respuesta anterior es correcta

Texto base, Tema 3.5.1 Compatibilidad de tipos

En este fragmento **p1** y **p2** apuntan a dos números REAL, y como no se puede asignar a un número INTEGER como es **numero** un valor REAL, daría un error de ejecución. Una trampa del examen, ya que por inercia darías la respuesta b).

4.- La instrucción WriteInt(100000 DIV 3, 2):

a) Da error por la operación.
b) Imprime el valor 33 en pantalla.
c) Da error porque faltan espacios para imprimir.
d) Imprime el valor 33333 en pantalla.

Texto base, Tema2.4.1 El tipo INTEGER, página 34:

"El rango depende del computador concreto utilizado … Los rangos mas comúnmente utilizados son los siguientes:

-32.768 … 0 … 32.767
-2.147.483.648 … 0 … 2.147.483.647"

Si estamos dentro del primer rango, la respuesta sería la a), pero si estamos dentro del segundo, la respuesta sería la b). Faltaría una respuesta que fuera "Depende del rango del computador para el tipo INTEGER". En concreto, en FST se usa el primer rango, y dará un error de compilación.

5.- En la sintaxis de Modula-2 el término VAR:

a) Es un identificador predefinido.
b) Es una palabra clave.
c) Es un identificador con uso particular.
d) Ninguna respuesta anterior es correcta.

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

Las "palabras clave" son lo que normalmente se denomina "palabras reservadas", pero este libro usa un vocabulario un tanto especial.

6.- Un nombre cualificado sirve para invocar a:

a) Identifica el nombre de un módulo que se va a importar.
b) Identifica a las variables definidas en un módulo.
c) <u>Un elemento de un módulo importado con anterioridad.</u>
d) Identifica a un tipo de datos definidos en un módulo.

Texto base, Tema 14.2.4 Uso de Módulos, página 405:

> "Existe otra forma de importación en un módulo, que se escribe simplemente:
>
> IMPORT nombre_de_modulo
>
> Con esta forma de importación se puede usar cualquier elemento definido en ese módulo, pero no directamente por su nombre, sino designándolo mediante la combinación:
>
> nombre_de_módulo.nombre_de_elemento
>
> Esta notación se llama nombre cualificado, y tiene apariencia similar a la referencia a campos de un registro, ya que también se usa el punto (.) para ligar los dos identificadores usados para la designación de un elemento particular."

Se refiere a poder usar cualquier elemento del módulo importado, no solo las variables o los tipos.

7.- Indicar cual de los siguientes esquemas de bucle no puede ejecutarse cero veces:

a) El esquema LOOP.
b) El esquema FOR.
c) El esquema WHILE.
d) **<u>El esquema REPEAT.</u>**

Texto base, Tema 10.1.1 Sentencia REPEAT

Como la condición de salida del bucle está obligatoriamente al final del REPEAT, deberá ejecutarse siempre como mínimo una vez. La condición de salida de un LOOP esta dentro del bucle, y no tiene por que llegar al END final del LOOP, aunque se quede en la sentencia anterior.

8.- El modelo de programación lógica corresponde a:

a) La programación imperativa.
b) <u>La programación declarativa.</u>
c) La programación más adecuada y conveniente.
d) La programación basada en operaciones lógicas.

Texto base, Tema 1.5.3 Modelo de programación lógica, página 18:

> *"Este modelo abstracto de cómputo corresponde plenamente a lo que se denomina programación declarativa"*

Pues ya sabes, a saberse de memoria estos trozos del libro

9.- Los registros con variantes se distinguen de los registros sin variantes por:

a) Realizar esquemas de tipo tupla.
b) Agrupar varios datos del mismo tipo.
c) Agrupar varios datos de tipos diferentes.
d) <u>Ninguna respuesta anterior es correcta.</u>

Texto base, Tema 12.5 Registros con variantes, página 343:

> *"Un registro con variantes tiene unos campos fijos, más una colección de variantes, cada una de las cuales cosiste en un grupo particular de campos. "*

La a) no es válida, ya que ambos son registros, y los registros son tuplas. La b) y c) no son ciertas, ya que pueden contener registros del mismo tipo, o de diferente tipo. Solo queda la d).

10.- Dado el siguiente fragmento de código:

```
VAR p1, p2 : POINTER TO INTEGER;
…
NEW(p1);
NEW(p2);
```

después de la asignación:

```
p1 := p2;
```

a) La variable a la que apuntaba p2 se pierde.
b) <u>p1 y p2 pasarán a apuntar a lo que apuntaba p1.</u>
c) La variable a la que apuntaba p1 se pierde.
d) El contenido de la variable apuntada por p2 pasa a ser 0.

Texto base, Tema 13.3. Variable dinámicas.

Lo que está claro es que la b) es correcta. De la a), podemos decir que puede ser o no cierta, ya que anteriormente puede haberse igualado p2 a otra variable, o incluso haber liberado el puntero p2. Como no sabemos lo que ha pasado, no podemos asegurar que a) sea cierto.

EJERCICIO

Realizar un programa en Modula-2 con dos opciones (P) ponerse en cola y (S) siguiente de la cola. Con la opción P se deberá introducir un nombre de 5 letras y un número del 0 al 100 y la opción S sacará estos datos por pantalla del primero en la cola. El número máximo de nombres que se pueden guardar en cola simultáneamente es de 50. La gestión de la cola es sacar los nombres en el orden en que se introducen.

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

CONST MAXELE = 50;

TYPE TIPOCADENA = ARRAY[0 .. 5] OF CHAR;
     REGISTRO = RECORD
        nombre : TIPOCADENA;
        numero : INTEGER;
     END;
     TIPOCOLA = ARRAY [0 .. MAXELE] OF REGISTRO;

VAR  Opcion  : CHAR;
     Cola    : TIPOCOLA;
     Puntero : CARDINAL;
     Nombre  : TIPOCADENA;
     Numero  : INTEGER;

PROCEDURE meter();      (* Meter un elemento en la cola *)
VAR i : CARDINAL;
    c : TIPOCADENA;
    n : INTEGER;
BEGIN
     IF (Puntero >= MAXELE) THEN
        WriteString("Pila llena, no caben mas elementos"); WriteLn;
     ELSE
         FOR i:=0 TO 5 DO c[i] := " "; END;    (* Borramos la cadena *)
         i := 0;  WriteString("Introduzca la cadena (Máximo 5 elementos): ");
         REPEAT  (* Pedimos los datos *)
            Read(c[i]); Write(c[i]); INC(i);
         UNTIL (i = 5) OR (c[i-1]=" ");
         WriteLn; WriteString("Número asociado: "); ReadInt(n); WriteLn;
         FOR i:=0 TO 5 DO Cola[Puntero].nombre[i] := c[i]; END; (* Los metemos *)
         Cola[Puntero].numero := n;
         INC(Puntero);
      END;
END meter;

PROCEDURE sacar();      (* Sacar un elemento de la cola *)
VAR i : CARDINAL;
BEGIN
     IF (Puntero = 0) THEN
        WriteString("Pila vacia"); WriteLn;
     ELSE
        DEC(Puntero);
        WriteString("Sacamos un elemento con texto: ");
        FOR i:=0 TO 5 DO Write(Cola[Puntero].nombre[i]); END;
        WriteString("   y valor: "); WriteInt(Cola[Puntero].numero,5); WriteLn;
     END;
END sacar;

BEGIN
     Puntero := 0; (* Iniciamos el valor dle puntero *)

     REPEAT  (* Pedimos que opción se desea *)
           WriteString("Introduzca Opción (P)oner, (S)acar, (F)in: ");
            Read(Opcion);WriteLn;
           IF (Opcion = "P") THEN meter(); END;
           IF (Opcion = "S") THEN sacar(); END;
     UNTIL (Opcion = "F");
END Examen.
```

No es nada elegante, ya que utiliza variables globales, por lo que solo puede tener una cola, pero funciona. Primero crea tipos para almacenar los elementos que se utilizan, luego van dos funciones, una para meter un elemento y otra para sacarlo. Se ha añadido una tercera, que sirve para ver la cola sin alterarla, imprescindible para probar el programa. Si quitamos esa función, cabe en mucho menos.

Lo que hay que tener siempre presente son las verificaciones antes de intentar meter o sacar elementos, ya que si no el programa fallará. Como mejoras, habría que considerar:

a) Poder meter datos en mayúsculas o minúsculas.
b) Poner la cola como otro parámetro más en las funciones, para que se puedan usar varias colas simultáneamente.
c) Usar una función mejor para introducir los nombres, ya que no verifica nada.
d) Añadir una tercera función, que sirva para ver la cola sin alterarla, imprescindible para probar el programa.