EXAMENES RESUELTOS DE PROGRAMACION I

Comentarios por Jose Antonio Vaqué

EXAMEN DE Febrero de 1997, segunda semana

Soluciones no oficiales

1.- El esquema de datos UNION, se logra mediante:

a) Registros.
b) **Registros con variantes.**
c) Conjuntos.
d) Formaciones.

Texto base, apartado 12.5. Registros con variantes, página 343:
"Los esquemas unión pueden utilizarse en programas en Modula-2 definiéndolos como registros con variantes".

Solo hay que estudiar un poco de vez en cuando, o mejor todavía saber como montar un registro con variantes, y que eso es una unión.

2.- La transparencia referencial impone:

a) No pasar argumentos por referencia.
b) Emplear sólo funciones puras.
c) Emplear desarrollo descendente.
d) **No acceder a variables externas.**

Texto base, apartado 8.1.2. Funciones, argumentos, página 191:
"La transparencia referencial se garantiza si la realización de funciones no utiliza datos externos a ella".

Seguimos con los estudios, que no vienen nunca mal. Como denomina "transparencia referencia" a que la función siempre devuelva los mismos resultados, si se llama con los mismos parámetros, es de cajón que no debe utilizar variables externas para conseguirlo.

3.- Los vectores abiertos:

a) Pueden tener cualquier dimensión.
b) **Sólo se pueden utilizar como argumentos.**
c) Sólo pueden tener tipos simples como elementos.
d) Simplifica las condiciones de contorno.

Texto base, apartado 11.6.Argumentos de tipo vector abierto, páginas 31 y siguientes

Aunque parece que tiene dos respuestas posibles, no es así. La respuesta A se refiere a que no se puede pasar una matriz de dos dimensiones, solo vale para vectores (no confundir dimensión con número de elementos). La C no es cierta, vale cualquier tipo de vector, y la D no tiene nada que ver.

4.- Después de ejecutar el siguiente fragmento de código:

```
TYPE tp = RECORD pri, seg : CHAR; END;
VAR x : tp;
PROCEDURE nom(VAR x : tp; y : tp);
BEGIN
  WITH x DO pri := CHR(ORD(seg) + 3); END;
  WITH y DO seg := pri; END;
END nom;
x.pri := "c"; x.seg := "d";   nom(x, x);
```

El valor de la variable global x es:

a) **x.pri = "g" y x.seg = "d"**
b) x.pri = "g" y x.seg = "g"
c) x.pri = "c" y x.seg = "d"
d) Se produce un error de doble referencia

Texto base, Tema 7 Funciones y procedimientos, Páginas 157 y siguientes:

Este tipo de pregunta que suele salir en el examen, hay que pensar un poco, y esquivar las trampas. Sigamos un poco el programa:

- Se define una variable, compuesta por dos CHAR, y se inicializa con los valores “c” y “d”.
- Se llama a una función, cuyo primer argumento es VAR, por lo que los cambios en esta variable se reflejan fuera.
- Se cambia el primer componente de la primera variable con su valor + 3 (pasa de “c” a “g”, y como es VAR se trasmite hacia fuera.
- Se cambia el segundo componente de la segunda variable con el valor del primero, pero no se trasmite.

Viendo esto, vemos que solo se cambia el primer componente del registro. La trampa es llamar a la variable global **x**, y llamar a los parámetros del procedimiento **x** e **y**, lo que no influye, pero puede despistar.

5.- Dado el siguiente fragmento de código:

```
FOR v := 1 TO 10000 DO END;
```

a) El bucle se ejecuta 0 veces.
b) **El bucle se ejecuta 10000 veces.**
c) Es un bucle infinito.
d) Es un bucle no válido.

Texto base, apartado 5.3.4 Sentencia FOR, páginas 112 y siguientes

Un regalito de pregunta. El bucle FOR no ejecuta nada en su interior, pero es un bucle, que se ejecuta 10.000 veces, sin hacer nada. Se usa mucho, para introducir una pausa en el programa, cuando no se disponen de sentencias para ello, y solo se emplea para pruebas del programa.

6.- Dada la siguiente definición en notación BNF:

```
expresion ::= { letra*numero }
letra ::= { a | b }
numero ::= { 1 | 2 }
```

Cual de las siguientes respuestas es INCORRECTA:

a) **aa**
b) a*1b*2
c) Una expresión vacía
d) *2*2

Texto base, Tema 2.1. Notación BNF, Página 27 y siguientes

Una expresión se define como cero o más repeticiones de una letra, un asterisco y un número. Una letra es una repetición de cero o mas veces las letras a o b o ambas en cualquier orden. Un número es una repetición de cero o mas veces el 1 o el 2 o ambos en cualquier orden.

7.- La programación a la defensiva recomienda usar:

a) Constantes.
b) Tipos.
c) **Subprogramas.**
d) Variables.

Texto base, Tema 8.3.1. Programación a la defensiva*, Página 216:*
"…consiste en que cada programa o subprograma esté escrito de manera que desconfíe sistemáticamente de los datos o argumentos con que se invoca…"

La programación a la defensiva trata de que se comprueben los argumentos antes de empezar a operar con ellos, lo que no tiene nada que ver con las posibles respuestas que dan en el examen. O bien hay un error en la trascripción del examen, o como soy muy burro y no consigo ver la respuesta. Pongo esa por que es la que mas se asemeja.

8.- Dado el siguiente fragmento de código:

```
TYPE Tiporef = (a, b, c, d);
     Tipoconj = SET OF Tiporef;
VAR C1 : Tipoconj;
```

El número de valores posibles de C1 es:

a) 0
b) **16**
c) 4
d) 15

Texto base, Tema 9.6 Conjuntos, Página 230 y siguientes

El tipo de referencia de C1 es un conjunto, de 4 posibles valores. C1 puede contener uno de estos valores (en los conjuntos no se admiten repeticiones de elementos en su interior):

1. Estar vacío
2. a
3. b
4. c
5. d
6. a, b
7. a, c
8. a, d
9. b, c
10. b, d
11. c, d
12. a, b, c
13. a, b, d
14. a, c, d
15. b, c, d
16. a, b, a, d

9.- Dado el siguiente fragmento de código:

```
TYPE TPTdias = POINTER TO Tdias;
     TCdias = SET OF Tdias;
     TPTCdias = POINTER TO TCdias;
VAR pr1 : TPTdias; pc1 : TPTCdias;
NEW( pr1 ); NEW( pc1 );
```

La inclusión de un elemento en el conjunto se hará:

a) INCL(pc1, pr1);
b) **INCL(pc1^, pr1^);**
c) INCL(TCdias, Tdias);
d) IN(pc1^, pr1);

Texto base, Tema 13.3.1. Punteros, Página 363 y siguientes
Texto base, Tema 9.6.3 Operaciones entre conjuntos, Página 237:
"INCL(S, X) Incluye el elemento X en el conjunto S."

Un puntero se comporta como la variable a la que apunta, por lo tanto **pr1** es una variable de tipo puntero a un elemento posible de un conjunto, y pr1^ es un elemento posible de un conjunto, y similarmente **pc1** es una variable de tipo puntero a un conjunto, y pc1^ es una variable de tipo conjunto, por ello:

- INCL(pc1, pr1) No se puede usar, ya que no son conjunto y elemento, sino punteros
- INCL(pc1^, pr1^) Conjunto y elemento, correcto
- INCL(TCdias, Tdias) No se puede usar, ya que solo son tipos definidos, no variables
- IN(pc1^, pr1) Conjunto y variable puntero, no es posible.

10.- Elegir la afirmación CORRECTA:

a) Los campos de un tipo registro son de tipo simple.
b) Cualquier tipo estructurado de Modula-2 es un esquema tupla.
c) En los registros con variantes es imprescindible el nombre del campo discriminante.
d) Para agrupar datos del mismo o diferente tipo, en Modula-2, se hace mediante la estructura tipo registro.

EJERCICIO DE PROGRAMACIÓN

Codificar un subprograma en Modula-2 que dado un vector de longitud L de números complejos, obtenga una matriz de tamaño M x N con los mismos valores del vector, siendo L = M * N. La condición que debe cumplir la matriz obtenida es que sus filas estén ordenadas crecientemente según la parte imaginaria y las columnas según la parte real. Se dispone del procedimiento *PROCEDURE Ordenar(VAR vect : tipov; val : tipop);* que ordena el vector *vect* crecientemente según la parte real cuando *val = real* o según la parte imaginaria cuando *val = imag*.

Ejemplo:
con L = 4

V =

5 + 5j	2 + 3j	10 + 3j	1 + 2j

Matriz =

1 + 2j	10 + 3j
2 + 3j	5 + 5j

con M x N = 2 x 2

Creo que el planteamiento no es correcto, ya el procedimiento **Ordenar** está mal definido. Siempre ordena el vector completo, cuando primero hay que ordenarlo completo, para montar la matriz, y luego ordenar esta por columnas. Como esto es imposible de conseguir llamando a ordenar simplemente, o el examen sería muy sencillo si ordenar se comporta de dos formas diferentes, planteo un programa completo que resuelve el problema según otro planteamiento, y a esperar que no toque un ejercicio similar en el examen.

```
MODULE Examen;
FROM InOut IMPORT WriteString,Write,WriteInt,WriteLn,Read;
CONST M = 2; N = 3; L = M * N;
TYPE tipoi = RECORD
        preal,pimag : INTEGER;
        END;
     tipov = ARRAY [1 .. L] OF tipoi;
     tipom = ARRAY [1 .. M],[1 .. N] OF tipoi;
     tipop = (real,imag);
VAR  Vector : tipov;
     Matriz : tipom;
     i,j,k  : INTEGER;
(* --------------------------------- ORDENA EL VECTOR POR LA PARTE IMAGINARIA *)
PROCEDURE OrdenaVector(VAR vect : tipov);
VAR i,j:INTEGER;
BEGIN
   FOR i:=1 TO L-1 DO
      FOR j:=i+1 TO L DO
         IF (vect[i].pimag > vect[j].pimag) THEN
            CambiaVector(vect,i,j);
         END;
      END;
   END;
END OrdenaVector;
(* ------------------------------------- ORDENA LA MATRIZ POR SU PARTE REAL *)
PROCEDURE OrdenaMatriz(VAR matriz : tipom);
VAR i,j,k:INTEGER;
BEGIN
   (* Si es por la parte real, solo columnas *)
   FOR k:=1 TO M DO
      FOR i:=1 TO N-1 DO
         FOR j:=i+1 TO N DO
            IF (matriz[k,i].preal > matriz[k,j].preal) THEN
               CambiaMatriz(matriz,k,i,j);
            END;
         END;
      END;
   END;
```

```
END OrdenaMatriz;
(* -------------------------- INTERCAMBIA DOS ELEMENTOS DEL VECTOR ENTRE SI *)
PROCEDURE CambiaVector(VAR vect : tipov; e1,e2:INTEGER);
VAR auxiliar:tipoi;
BEGIN
   auxiliar := vect[e1];
   vect[e1] := vect[e2];
   vect[e2] := auxiliar;
END CambiaVector;
(* ------------------------- INTERCAMBIA DOS ELEMENTOS DE LA MATRIZ ENTRE SI *)
PROCEDURE CambiaMatriz(VAR matriz : tipom; col,e1,e2:INTEGER);
VAR auxiliar:tipoi;
BEGIN
   auxiliar       := matriz[col,e1];
   matriz[col,e1] := matriz[col,e2];
   matriz[col,e2] := auxiliar;
END CambiaMatriz;
(* ---------------------------------------------------------- IMPRIME EL VECTOR *)
PROCEDURE VerV(Vec:tipov);
VAR i:INTEGER;
BEGIN
   WriteLn;
   FOR i:=1 TO L DO
      WriteInt(i,2);WriteString(":[");
      WriteInt(Vec[i].preal,2);WriteString("+");
      WriteInt(Vec[i].pimag,2);WriteString("j]");WriteString("  ");
   END;
   WriteLn;
   WriteLn;
END VerV;
(* ---------------------------------------------------------- IMPRIME LA MATRIZ *)
PROCEDURE VerM(Mat:tipom);
VAR i,j:INTEGER;
BEGIN
   WriteLn;
   FOR i:=1 TO M DO
      FOR j:=1 TO N DO
         WriteInt(Mat[i,j].preal,3);WriteString("+");
         WriteInt(Mat[i,j].pimag,3);WriteString("j");
         WriteString("     ");
      END;
      WriteLn;
   END;
   WriteLn;
   WriteLn;
END VerM;
(* ---------------------------------------------------------- PROGRAMA PRINCIPAL *)
BEGIN
   (* Metemos los valores en el vector *)
   Vector[1].preal:= 5;  Vector[1].pimag:=5;
   Vector[2].preal:=22;  Vector[2].pimag:=3;
   Vector[3].preal:=10;  Vector[3].pimag:=3;
   Vector[4].preal:= 1;  Vector[4].pimag:=2;
   Vector[5].preal:= 5;  Vector[5].pimag:=1;
   Vector[6].preal:= 6;  Vector[6].pimag:=7;
   WriteString("Vector inicial: ");VerV(Vector);
   (* Ordenamos por la parte imaginaria *)
   OrdenaVector(Vector);
   WriteString("Odenado por parte imaginaria: ");VerV(Vector);
   (* Montamos la matriz *)
   k := 1;
   FOR i:=1 TO M DO
      FOR j:=1 TO N DO
         Matriz[i,j].preal:=Vector[k].preal;
         Matriz[i,j].pimag:=Vector[k].pimag;
         k:= k + 1;
      END;
   END;
   WriteString("Matriz inicial: "); VerM(Matriz);
   (* Ordenamos por la parte real *)
   OrdenaMatriz(Matriz);
   WriteString("Matriz final: "); VerM(Matriz);
END Examen.
```