Introducción a la programación - Ejercicios Tema 4

DISIA - Facultad de Informática UCM (2009-2010)
Ing. Técnica en Informática de Gestión - 1º B
Profesores: Federico Peinado Gil y Pablo Moreno Ger

Ejercicio 1.- Traza y muestra la salida por pantalla del programa siguiente:

```
program prueba;
var
   A,B,C: Integer;

procedure UNO(A: Integer; Var B: Integer; C: Integer);
begin
     C:= A + B;
     B:= A;
     A:= C;
     write('ESCRIBE EL Procedure : ');
     writeln('A = ',A,' B = ',B,' C = ',C);
end;  (* uno *)

begin
     A:= 1;  B:= 2;  C:= 3;
     UNO(A,B,C);
     write('ESCRIBE EL PROGRAMA PRINCIPAL : ');
     writeln( 'A = ',A,' B = ',B,' C = ',C);
     UNO(B,A,C);
     write( 'ESCRIBE EL PROGRAMA PRINCIPAL : ');
     writeln( 'A = ',A,' B = ',B,' C = ',C);
end.
```

Ejercicio 2.- Suponiendo las siguientes declaraciones en un programa:

```
var
  X, Y, Z : REAL;
  M, N : INTEGER;
procedure MEZCLA (var A, B : REAL; X : INTEGER);
```

¿Cuáles de las siguientes llamadas al procedimiento están mal y por qué?

```
MEZCLA  (X,  Y,  Z)      MEZCLA (Y, X, N)            MEZCLA (X, Y, M+N)
                         MEZCLA (Y+Z, Y-Z, M)        MEZCLA (Z, Y, MAXINT)
MEZCLA (A, B, X)         MEZCLA (M, Y, N)
MEZCLA  (X,  Y,  8)      MEZCLA (25.0, 15.5, X)
                         MEZCLA (X, Y, M, 10)
```

Ejercicio 3.- Implementa las siguientes funciones booleanas:

EsMinuscula	indica si un carácter dado es una letra minúscula
EsMayuscula	indica si un carácter dado es una letra mayúscula
EsDigito	indica si un carácter dado es un dígito
EsLetra	indica si un carácter dado es una letra
EsAlfanumerico	indica si un carácter dado es un alfanumérico, ya sea letra o dígito

Implementa un programa en el que se haga uso de dichas funciones.

Ejercicio 4.-
a) Escribe el subprograma INVERSO que devuelve el resultado de invertir el número entero positivo que acepta. Se entiende por invertir dar la vuelta a los dígitos que componen el número (hallar su *imagen especular*); así, el inverso de 3952 es 2593.

b) Escribe el subprograma CAPICUA que, haciendo uso del subprograma INVERSO, devuelva un valor booleano que indique si el número entero positivo que recibe como argumento es o no capicúa.

c) Escribe un programa que solicite números enteros positivos e indique si son o no capicúas. El programa solicitará números hasta que se introduzca uno negativo y usará los subprogramas anteriores para determinar qué números de los introducidos son capicúa.

Ejercicio 5.- Escribe un subprograma que reciba dos números enteros positivos y devuelva su producto calculado usando sólo sumas.

Ejercicio 6.- Implementa un subprograma que reciba dos valores de tipo *Integer*. El subprograma pedirá la introducción de un número entero a través del teclado, tantas veces como sea necesario, hasta que el número suministrado se encuentre en el intervalo determinado por los datos de entrada. Una vez validado el número leído, el subprograma lo devolverá.

Ejercicio 7.- Escribe un subprograma que lea una secuencia de caracteres introducida por teclado y terminada en salto de línea, y cuente y devuelva el número de caracteres anteriores a la primera aparición de un carácter que se le pasa como argumento. Así mismo, el subprograma deberá devolver un valor que indique si en la secuencia de entrada se encontró o no el citado carácter.

Ejercicio 8.- Examen febrero 2000-01, 4,5 puntos. Para que los cálculos resultasen más fáciles, los romanos solían utilizar una notación aditiva pura. Un número romano en esta notación era una cadena de dígitos, comenzando por el dígito del valor más alto y terminando con el de valor más pequeño, y su valor se obtenía simplemente como la suma de sus dígitos (en esta notación, por ejemplo, el número arábigo 19 se representaba XVIIII).

a) Escribe el subprograma *romanoAEntero* que lea una cadena de caracteres introducida por teclado y acabada en fin de línea, y devuelva
   - un valor que indique si la cadena leída se puede interpretar como un número romano válido en notación aditiva pura, y
   - el número arábigo equivalente calculado en caso de ser válido el número romano.

   Así, por ejemplo, si se introduce por teclado *MDCCCCLXXXVIIII<eoln>* el subprograma devuelve un valor que indica que el número romano era válido y el número arábigo *1989*. En cambio, si se introduce por teclado *¿X<eoln>*, el subprograma devuelve un valor que indica que no es un número romano válido y un valor cualquiera como número arábigo equivalente.
   NOTA: la cadena no se podrá interpretar como número romano si contiene algún carácter que no sea un *dígito* romano. Recuerda que los dígitos romanos y sus valores correspondientes son: I = 1; V = 5; X = 10; L = 50; C = 100; D = 500; M = 1000.

b) Haciendo uso del subprograma *romanoAEntero*, implementa un programa que, de forma iterativa, haga lo siguiente:
   Muestre un menú con dos opciones:
   A. Operar con números romanos
   B. Salir
   - Si el usuario elige la opción A, el programa leerá de teclado dos cadenas representando números romanos en notación aditiva pura y un carácter que representa un operador aritmético, y mostrará por pantalla el resultado de la operación solicitada como número arábigo y como número romano. Para esto último supón implementado en el programa, y usa, el procedimiento de cabecera *procedure enteroARomano (num: integer);* que recibe un número arábigo y muestra por pantalla el

número romano equivalente en notación aditiva. Sitúa este procedimiento en el punto del programa que consideres apropiado.
Hay que tener en cuenta que los números romanos introducidos pueden no ser válidos y/o el carácter representando la operación puede ser incorrecto –los operadores considerados válidos son +, -, * o *:* representando, respectivamente, la suma, la resta, la multiplicación y la división entera. Si algún dato no es correcto, simplemente no hay que realizar cálculos.

- Si el usuario elige la opción B, finalizará la ejecución del programa.

Ejercicio 9.- Examen febrero 98-99, 3,5 puntos. En combinatoria, el número de variaciones de *x* elementos de orden *y* (x>0, 0<y≤ x), $V_{x,y}$, el número de permutaciones de *x* elementos (x>0), P_x, y el número de combinaciones de *x* elementos de orden *y* (x>0, 0<y≤ x), $C_{x,y}$, se obtienen mediante las siguientes fórmulas:

$$V_{x,y} = x \cdot (x-1) \cdot (x-2) \cdot \ldots \cdot (x-y+2) \cdot (x-y+1)$$

$$P_x = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot (x-2) \cdot (x-1) \cdot x$$

$$C_{x,y} = {V_{x,y}}/{P_y}$$

a) Implementa (sin usar recursividad en ningún caso):
   - la función *VARIACIONES*, que reciba dos números enteros *x* e *y* y calcule y devuelva el entero que representa $V_{x,y}$,
   - la función *PERMUTACIONES*, que reciba un número entero *x* y calcule y devuelva el entero que representa P_x, y
   - el procedimiento *COMBINACIONES*, que reciba dos números enteros *x* e *y* y calcule y devuelva (haciendo uso de las dos funciones anteriores) el entero que representa $C_{x,y}$.

   NOTA: se supone que el/los número(s) que reciben como entrada los tres subprogramas cumplen las condiciones necesarias para poder calcular los valores correspondientes es decir, dentro de los subprogramas no es necesario comprobar la validez de los datos de entrada.

b) Implementa un programa que contenga los subprogramas anteriores y, de modo iterativo, haga lo siguiente: solicite una pareja de números enteros positivos *x* e *y*, y, haciendo uso de dichos subprogramas, muestre por pantalla los valores $V_{x,y}$, P_x y $C_{x,y}$ si para cada caso se cumplen la(s) condición(es) indicada(s) en la definición correspondiente. En el momento en que se cumpla x=y=0 el programa deberá finalizar.

Ejercicio 10.- Examen junio 98-99, 1 punto. Dado el siguiente programa

```
PROGRAM ejemplo;

CONST
      pi = 3.1416;
      diez = 10;

VAR
      r1, r2: REAL;

FUNCTION redondeo (x: REAL) : INTEGER;
(* redondea el número real x *)
VAR
      fraccion: REAL;
      parteEntera: INTEGER;

      PROCEDURE ruptura (num: REAL; var fraccion: REAL;
                         var entero: INTEGER);
      (* separa el número real num en sus partes entera y decimal *)

      BEGIN (* ruptura *)
            ... (* código del procedimiento *)

      END; (* ruptura *)
```

```
BEGIN (* redondeo *)
      ...  (* código de la función *)
END; (* redondeo *)

BEGIN (* p.p. *)
      ...
END. (* p.p. *)
```

Identifica y numera los bloques. Dibuja una tabla de alcance y visibilidad de los identificadores marcados en negrita : *r1* (variable del programa principal) y *fraccion* (variable local de la función *redondeo*).

Ejercicio 11.- Examen febrero 97-98, 1,5 puntos. Dadas las siguientes declaraciones de constantes y variables

```
CONST
      pi = 3.1415;
      e = 2.7182;
      centinela = '*';

VAR
      quizas: boolean;
      longTotal, longParcial, x: real;
      caracLeido: char;
      i, j: integer;
```

y las siguientes cabeceras de subprogramas

```
PROCEDURE P1 (var Param1, Param2: real; Param3: integer);
PROCEDURE P2 (var Param1: boolean; var Param2: char; var Param3: in-
teger);
FUNCTION F1 (x, y, z: real; a: boolean; h: char) : integer;
```

indica cuáles de las siguientes instrucciones (o grupos de instrucciones) son válidas. En los que no lo sean, indica todos los errores existentes.

a)
```
if not quizas then
      P1(i, longTotal, MAXINT);
```

b)
```
x := F1(longParcial, 7.0, longTotal, True, Pred(centinela)) / 6
mod 2;
i := round(pi);
writeln(i:4:2);
```

c)
```
P2(not quizas, caracLeido, j);
```

d)
```
case ord(quizas) of
      0: quizas := F1(pi * e, pi, e, quizas, caracLeido) <
trunc(pi);
      1: while caracLeido <> centinela do
            read(caracLeido)
end;
```

e)
```
F1(longTotal, longParcial, pi, False, Succ(centinela)) :=
Ord(caracLeido);
```