Fundamentos de la programación y la informática
Examen Parcial. 23 de Noviembre de 2023
Grados en ingeniería aeroespacial. Turno de tarde
Universidad Rey Juan Carlos

- Ejecuta en un terminal la orden
  `~mortuno/prepara`
- Comprueba que esto ha dejado en tu cuenta los ficheros
  `~/fpi.nov.23/logica.TULOGIN.txt`, donde deberás resolver el ejercicio 1 [1].
  `~/fpi.nov.23/parcial23.TULOGIN.txt`, donde deberás resolver el ejercicio 2.

Ejercicio 1 (2 puntos)

Sean las expresiones

```
e1 := not (a and not b) or not (c and not d) or (not e and f);
```

```
e2 := not (x >= 3) and (y = 20);
```

1. A partir de e1, aplica De Morgan y escribe una expresión lógica equivalente, intentando que sea más clara, de nombre s1
2. A partir de e2, escribe una expresión lógica equivalente, más legible, de nombre s2
3. Escribe una expresión n1, de la forma más clara posible, que sea la negación de s1
4. Escribe una expresión n2, de la forma más clara posible, que sea la negación de s2

Solución

```
s1: not a or b or not c or d or (not e and f)
```

```
s2: x<3 and y=20
```

```
n1: a and not b and c and not d and (e or not f)
```

```
n2: x>=3 or y<>20
```

Ejercicio 2 (8 puntos)

Completa el esqueleto de programa en Pascal que encontrarás en el fichero `~/fpi.nov.23/parcial23.TULOGIN.txt`, para que cumpla la siguiente especificación:

1. El objetivo del programa es comparar dos probabilidades e indicar si son prácticamente iguales, si la primera es mayor que la segunda o si la segunda es mayor que la primera.
2. Como sabes, una probabilidad ha de ser un número real mayor o igual que 0, y además, menor o igual que 1. El programa tendrá una función llamada *precond_prob* que compruebe si esto se cumple.

[1] TULOGIN será tu nombre de usuario en el laboratorio. La cuenta genérica alumno no es válida

3. El programa tendrá una función llamada *compara_probabilidades* que recibirá dos probabilidades y devolverá un código, que será un número entero:

   - El código 1 si la primera probabilidad es mayor que la segunda.
   - El código 2 si la primera probabilidad es menor que la segunda.
   - El código 0 si ambas probabilidades son prácticamente iguales. Consideraremos que lo son si la diferencia entre ambas es menor que la constante (local del cuerpo principal) Epsilon, que valdrá 0.001.

4. El programa tendrá un procedimiento llamado *escribe_resultado* que recibirá, al menos, dos probabilidades y el código (0,1 o 2) y mostrará en pantalla mensajes como los del ejemplo de ejecución mostrado al final del enunciado. Este código será un número entero.

5. El programa tendrá un procedimiento llamado *revisa_compara_escribe* que:

   a) Recibirá dos números reales y la constante Epsilon.

   b) Si ambos números están entre 0 y 1, comparará las probabilidades y escribirá en pantalla si la primera es mayor, si la primera es menor o si son prácticamente iguales.

   c) Si alguno de los números está fuera de este rango, o ambos lo están, mostrará un mensaje de error.

   d) El programa mostrará todos los números en notación tradicional (no científica), con 6 decimales.

   e) El programa no leerá ningún valor desde el teclado.

   f) Puedes dejar en el código fuente las llamadas de prueba que hagas a tus funciones.

   g) Cada uno de los subprogramas aquí indicados puede llamar a otros subprogramas si lo crees conveniente.

   Este es un ejemplo de ejecución del programa:

```
Error: es necesario que 2.000000 y 2.001000 estén dentro del rango [0,1]
1.000000 es mayor que 0.999000
1.000000 y 0.999900 son prácticamente iguales
0.500000 es menor que 0.999900
```

Esqueleto para completar:

```
{$mode objfpc}{$H-}{$R+}{$T+}{$Q+}{$V+}{$D+}{$X-}{$warnings on}


program parcial23tulogin;

function precond_prob(              // COMPLETA ESTO

function compara_probabilidades(    // COMPLETA ESTO

procedure escribe_resultado(        // COMPLETA ESTO

procedure revisa_compara_escribe(   // COMPLETA ESTO

const
    // COMPLETA ESTO
var
    // COMPLETA ESTO
begin
    p1 := 2;
    p2 := 2.001;
    revisa_compara_escribe(p1,p2,Epsilon);
```

```
    p1 := 1;
    p2 := 0.999;
    revisa_compara_escribe(p1,p2,Epsilon);

    p1 := 1;
    p2 := 0.9999;
    revisa_compara_escribe(p1,p2,Epsilon);

    p1 := 0.5;
    p2 := 0.9999;
    revisa_compara_escribe(p1,p2,Epsilon);
end.
```

Solución:

```
{$mode objfpc}{$H-}{$R+}{$T+}{$Q+}{$V+}{$D+}{$X-}{$warnings on}
```

```
1  program parcial23;
2
3  function precond_prob(p:real):boolean;
4  begin
5      result := (p>=0) and (p<=1);
6  end;
7
8  function compara_probabilidades(p1,p2, epsilon: real): integer;
9  var
10     practicamente_iguales: boolean;
11 begin
12     practicamente_iguales :=  abs(p1-p2) < epsilon;
13
14     if not practicamente_iguales and (p1 > p2) then
15         result := 1;  // El primero es mayor
16
17     if not practicamente_iguales and (p2 > p1) then
18         result := 2;  // El segundo es mayor
19
20     if practicamente_iguales then
21         result := 0;
22 end;
23
24 procedure escribe_resultado(p1,p2: real;codigo_comparacion: integer);
25 begin
26     if (codigo_comparacion = 0) then begin
27         write(p1:0:6, ' y ');
28         write(p2:0:6);
29         writeln(' son prácticamente iguales')
30     end;
31
32     if (codigo_comparacion = 1) then begin
33         write(p1:0:6, ' es mayor que ');
34         writeln(p2:0:6);
35     end;
36
37     if (codigo_comparacion = 2) then begin
38         write(p1:0:6, ' es menor que ');
39         writeln(p2:0:6);
40     end;
41
42     if (codigo_comparacion <0) or (codigo_comparacion>2) then begin
43         write('Error: el código de comparación ',codigo_comparacion);
44         writeln(' es incorrecto. Debe ser 0,1 o 2');
45     end;
46 end;
47
48 procedure revisa_compara_escribe(p1,p2, epsilon:real);
49 begin
50     if (precond_prob(p1) and precond_prob(p2)) then begin
51         escribe_resultado(p1, p2, compara_probabilidades(p1,p2,epsilon));
52     end
53     else begin
54         write('Error: es necesario que ',p1:0:6);
55         write(' y ',p2:0:6);
56         writeln(' estén dentro del rango [0,1]');
57     end;
58 end;
```

```
59
60  const
61      Epsilon = 0.001;
62  var
63      p1,p2: real;
64  begin
65      p1 := 2;
66      p2 := 2.001;
67      revisa_compara_escribe(p1,p2,Epsilon);
68
69      p1 := 1;
70      p2 := 0.999;
71      revisa_compara_escribe(p1,p2,Epsilon);
72
73      p1 := 1;
74      p2 := 0.9999;
75      revisa_compara_escribe(p1,p2,Epsilon);
76
77      p1 := 0.5;
78      p2 := 0.9999;
79      revisa_compara_escribe(p1,p2,Epsilon);
80  end.
```

Observaciones

1. En este ejemplo, los subprogramas que *calculan* siempre son funciones y lo que *hacen* siempre son procedimientos. Por tanto, no es necesario ningún procedimiento con parámetros de salida (parámetros por referencia).

   Solo sería necesario si un subprograma tuviera que devolver más de un valor, o si estuviéramos manejado algún dato que ocupe mucha memoria. (O si el enunciado lo hubiera pedido explícitamente por algún otro motivo, por ejemplo, demostrar saber manejar parámetros de salida)

2. Las funciones *calculan* valores y los devuelven. No escriben nada. Los procedimientos, escriben los valores calculados por las funciones.

3. Un principiante es fácil que cometa alguna variante del siguiente error

   ```
   function compara_probabilidades(p1,p2, epsilon: real): integer;
   begin     // ¡¡MAL!!
       if abs(p1-p2) < epsilon then
           result := 0;

       if (p1 > p2) then
           result := 1;  // El primero es mayor

       if (p2 > p1) then
           result := 2;  // El segundo es mayor
   end;    // ¡¡MAL!!
   ```

   En caso de que los probabilidades sean similares, *result* valdría inicialmente 0. Esto es correcto. Pero después, pasaría a valer, erróneamente, 1 o 2 si alguno de los dos valores fuera mayor que el otro, aunque la diferencia fuera despreciable.

4. En las líneas 61, 67, 71, 75 y 79, esto es, en el cuerpo del programa principal, *Epsilon* es una constante y por tanto se escribe con mayúscula. En todas las demás, es un parámetro y se escribe con minúscula.

5. La función mostrada a continuación devuelve lo que debe. Pero incumple la especificación, porque *epsilon* es una variable de la función *compara_probabilidades*. Y por tanto, no se podría cambiar fácilmente. Tal y como indica el enunciado, debería ser una constante local del cuerpo del programa principal, que luego se pase como parámetro a los subprogramas que lo necesiten.

```
function compara_probabilidades(p1,p2: real): integer;
var
    epsilon: real = 0.001;
    practicamente_iguales: boolean;
begin
    practicamente_iguales :=  abs(p1-p2) < epsilon;

    if not practicamente_iguales and (p1 > p2) then
        result := 1;  // El primero es mayor

    if not practicamente_iguales and (p2 > p1) then
        result := 2;  // El segundo es mayor

    if practicamente_iguales then
        result := 0;
end;
```

6. En esta versión de la solución, el procedimiento *revisa_compara_escribe*, no distingue si una probabilidad incumple la precondición de estar entre 0 y 1, si lo incumple la otra o ambas incumplen. Es lo que el enunciado pedía, para que resultara especialmente sencillo. Normalmente distinguiríamos estos casos para dar un error lo más detallado posible.