Fundamentos de la programación y la informática
Examen de ejercicios de laboratorios
19 de Enero 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.enero.23/cuenta.TULOGIN.pas`
  `~/fpi.enero.23/huevos.TULOGIN.pas` [1].
- Para entregar el ejercicio, ejecuta
  `~mortuno/entrega`
  Conviene que relices varias entregas durante el examen, como medida de seguridad. La última elimina todas las anteriores.
- Recuerda que si alguno de los programas no compila, su nota será nula.

Ejercicio 1 (4 puntos)

Edita el fichero `~/fpi.enero.23/cuenta.TULOGIN.pas` para escribir un programa en Pascal que contenga

- Un subprograma que reciba una cadena y devuelva el número de dígitos que contiene. Ejemplos:

```
[cadena vacía]
digitos:0

hola
digitos:0

2  j 44
digitos:3

1111-2
digitos:5
```

- Varias llamadas al subprograma anterior, probando, al menos, estos 4 ejemplos o ejemplos similares.
- Y por supuesto, todos los subprogramas adicionales que consideres oportunos.

Solución

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

```
1  program cuenta;
2  function cuenta_digitos(cadena:string):integer;
3  var
4      i,contador:integer;
5  begin
6      contador:=0;
7      for i:=1 to length(cadena) do
8          if (cadena[i] >= '0') and (cadena[i]<='9') then
9              contador:=contador+1;
10     result := contador;
11 end;
```

[1] TULOGIN será tu nombre de usuario en el laboratorio

```
12
13 procedure cuenta_y_escribe(cadena:string);
14 begin
15     writeln(cadena);
16     writeln('digitos:',cuenta_digitos(cadena));
17     writeln();
18 end;
19 var
20     ejemplo:string;
21 begin
22     ejemplo:='';
23     cuenta_y_escribe(ejemplo);
24
25     ejemplo:='hola';
26     cuenta_y_escribe(ejemplo);
27
28     ejemplo:='2  j 44';
29     cuenta_y_escribe(ejemplo);
30
31     ejemplo:='1111-2';
32     cuenta_y_escribe(ejemplo);
33 end.
34
```

Observaciones:

- Otra solución igualmente válida sería, en la línea 8, detectar si el caracter es un dígito o no usando el procedimiento *val*: intentaríamos convertir `cadena[i]` en entero, y cuando sea posible, esto es, cuando el código devuelto sea 0, incrementaríamos el contador.

Ejercicio 2 (6 puntos)

La Unión Europea clasifica los huevos de gallina por su peso según la siguiente tabla

```
S   Pequeños        peso < 53g
M   Medianos        53g <= peso < 63g
L   Grandes         63g <= peso <= 73g
XL  Super Grandes   peso > 73g
```

Adicionalmente, nosotros vamos a considerar que un huevo pequeño ha de pesar 43 gramos o más. Y que un huevo super grande ha de pesar como mucho 83 gramos, pero no más. Si un huevo está fuera de estos límites, será un error (medición incorrecta, huevo inadecuado o cualquier otro factor)

Completa el programa en Pascal que encontrarás en el fichero `~/fpi.enero.23/huevos.TULOGIN.txt`, según la siguiente especificación.

1. El fichero contiene el esqueleto de un programa con una función que simula el pesaje de un huevo y devuelve un peso en gramos. Esta función devuelve un peso aleatorio, pero no con distribución uniforme: habrá muchos huevos de peso medio y pocos de pesos extremos. Esta función ya está resuelta y no es necesario que te preocupes de su contenido: te basta saber que proporciona una medición, en gramos.

   El esqueleto es el siguiente:

   ```
   program huevos;
   uses math;

   function pesaje():real;
   const
   ```

```
    Media = 63;
    DesvTip = 15;
begin
    result := randg(Media, DesvTip);
end;

begin
end.
```

2. Añade un subprograma que tenga un parámetro de entrada llamado *peso* y que devuelva la cadena *S*, *M*, *L*, *XL*, o *ERROR* (todo siempre en mayúsculas), según lo todo lo indicado anteriormente.

3. Añade los subprogramas necesarios para que el programa simule un pesaje, lo clasifique y escriba en pantalla peso y clasificación. Esto se debe hacer N veces, donde N será una constante en el cuerpo del programa principal con el valor 30. Algo parecido a lo siguiente (aunque aquí N vale 6)

```
 43.9 S
 52.2 S
 59.6 M
101.3 ERROR
 71.2 L
 65.0 L
```

Observaciones

- Puedes considerar que los pesos que marcan todos los límites están fijados de forma universal e inmutable, y por tanto puedes escribirlos dentro del subprograma como constantes literales (como números *tal cual*), no como constantes.
- El ejercicio del pasado noviembre era similar a este, pero con alguna diferencia importante:
  - Como seguramente recordarás, aquel enunciado solicitaba distinguir en un parámetro de salida si los parámetros de entrada eran correctos o no. En caso de ser correctos, se clasificaba el coche.
  - Observa que, por motivos de tiempo, este enunciado es más sencillo. No solicita un parámetro de éxito/error (lo cual sería mejor diseño). Basta con que consideres el error como una categoría más.

Solución

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

```
1  program huevos;
2  uses math;
3
4  function pesaje():real;
5  const
6      Media = 63;
7      DesvTip = 15;
8  begin
9      result := randg(Media, DesvTip);
10 end;
11
12 function clasifica_huevo(peso:real):string;
13 begin
14     if (43 <= peso) and (peso < 53) then
```

```
15          result := 'S'
16      else if (53 <= peso)  and (peso<63) then
17          result := 'M'
18      else if (63 <= peso ) and (peso<=73) then
19          result := 'L'
20      else if (73 < peso) and (peso <= 83) then
21          result := 'XL'
22      else
23          result := 'ERROR'
24  end;
25
26  procedure pesa_y_clasifica_huevos(N: integer);
27  var
28      i: integer;
29      peso: real;
30  begin
31      for i:=1 to N do begin
32          peso:= pesaje();
33          write(peso:6:1,' ');
34          writeln(clasifica_huevo(peso));
35      end;
36  end;
37
38  const
39      N= 30;
40
41  begin
42      randomize;
43      pesa_y_clasifica_huevos(N);
44  end.
```

Observaciones

- Linea 33. Escribimos `peso:6:1` para que el número ocupe siempre 6 caracteres (y tenga 1 dígito decimal). Esto es: 3 caracteres para el peso, 1 caracter para el punto, otro para el decimal y un espacio adicional. De esa forma, la categoría siempre queda perfectamente alineada en una columna.
- El subprograma *clasifica_huevo()* no escribe nada en pantalla, solo devuelve un resultado. Aunque fuera un procedimiento, tampoco debería escribir nada, solo devolver una cadena. Esto es un principio básico de diseño: unos subprogramas calculan y otros hacen cosas con lo calculado. Además, el enunciado lo pide explícitamente.
- Línea 32. Llamamos a la función *pesaje()*, guardamos el resultado en una variable y en lo sucesivo usamos siempre la variable para considerar siempre el mismo peso, y no otro nuevo. Un error frecuente entre principiantes es escribir cosas como

  ```
  write(pesaje():0:1,' ');
  writeln(clasifica_huevo(pesaje()));
  ```

  Esto es, llamar a *pesaje()* para escribir en pantalla un peso aleatorio, y luego volver a llamar a la función para obtener un segundo peso aleatorio, distinto del anterior, que es el que se clasifica. Por tanto, el resultado es incoherente.

  Otro error similar, aún peor, es escribir cosas como

  ```
  if (pesaje() >= 43) and (pesaje() < 53) then
      result := 'S'
  else if (pesaje()> 53 ) and (pesaje()<63) then
  ...
  ```