Departamento de Informática e Ingeniería de Sistemas
Universidad Zaragoza

Examen de Teoría

22 de enero de 2022

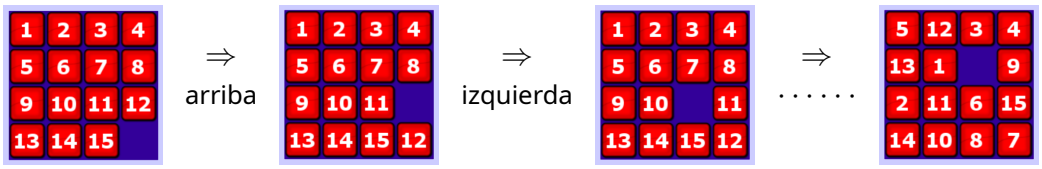
Fundamentos de Informática
Grado en Tecnologías Industriales
Duración: 2h 45m

Ejercicio 1 **[3.5 puntos]**

El puzle del 15 o *taken* es un pasatiempo formado por 15 piezas numeradas deslizables en un marco en el que caben 16 (4 por 4), por lo que dejan un hueco libre en el que se puede deslizar una de las piezas adyacentes.

El pasatiempo consiste en, partiendo del tablero inicial con las piezas en orden numérico, desordenarlas deslizando aleatoriamente una de las piezas hacia el hueco varias veces, y luego intentar volver a ordenarlo por el mismo procedimiento.

En este ejercicio vas a implementar un generador de puzles de este tipo, partiendo del tablero ordenado y realizando una serie de movimientos para desordenarlo. Para desordenar el puzle, en la realidad se mueven piezas dentro del hueco, pero podemos verlo de una forma más útil: es el hueco el que se mueve en una dirección determinada (arriba, derecha, abajo, izquierda) y se intercambia con la pieza correspondiente.



Define los siguientes tipos de datos:

- Un tipo `tpCasilla`, que represente una casilla del puzle (teniendo en cuenta que puede estar vacía).
- Un tipo `tpPuzle`, que represente el tablero completo.

A partir de esos tipos de datos, implementa las siguientes procedimientos:

- Un procedimiento que inicialice el puzle con las piezas ordenadas y el hueco en la esquina inferior derecha:

```
1  procedure Inicializa(??? pz: tpPuzle);
```

- Un procedimiento que encuentre la posicion del hueco (casilla vacía) y devuelva sus coordenadas:

```
1  procedure PosicionHueco(??? pz: tpPuzle; ??? hi,hj: integer);
```

- Un procedimiento que a partir de un código de dirección **d** representado con un entero (1 ⇒ arriba, 2 ⇒ abajo, 3 ⇒ izquierda, 4 ⇒ derecha), mueva el hueco en esa dirección:

```
1  procedure MueveHueco(??? pz: tpPuzle; d: integer);
```

  Para ello necesitarás encontrar la posición del hueco, e intercambiarlo con la pieza de la casilla correspondiente (si es posible, ten cuidado con los movimientos en los bordes...).

- Un procedimiento que genere el puzle, inicializándolo ordenado, y realizando 100 movimientos aleatorios del hueco:

```
1  procedure CreaPuzle(??? pz: tpPuzle);
```

  Para ello puedes utilizar la funcion de Pascal `random(n)`, que cada vez que se invoca devuelve un número entero aleatorio entre 0 y $n-1$.

Puedes definir estructuras de datos o procedimientos y funciones adicionales si lo consideras necesario.

Propuesta de solución:

```
program quince;
const
  SIZE = 4;

type
  tpCasilla = record
    ocupada : boolean;
    numero : integer;
  end;
  tpPuzle = array[1..SIZE,1..SIZE] of tpCasilla;

procedure Inicializa(var p: tpPuzle);
var
  i,j : integer;
begin
  for i:=1 to SIZE do
  begin
    for j:=1 to SIZE do
    begin
      p[i,j].ocupada := true;
      p[i,j].numero  := (i-1)*SIZE + (j);
    end;
  end;
  p[SIZE,SIZE].ocupada := false;
end;

procedure PosicionHueco(const p: tpPuzle; var hi,hj: integer);
var
  i,j : integer;
begin
  hi := 0; hj := 0;
  for i:=1 to SIZE do
  begin
    for j:=1 to SIZE do
    begin
      if not p[i,j].ocupada then
      begin
        hi := i; hj := j;
      end;
    end;
  end;
end;









procedure Intercambia(var a,b: tpCasilla);
var
  aux : tpCasilla;
```

```
52 begin
53   aux := a;
54   a   := b;
55   b   := aux;
56 end;
57
58 procedure MueveHueco(var p:tpPuzle; d: integer);
59 var
60   hi,hj : integer;
61 begin
62   PosicionHueco(p,hi,hj);
63   if (d=1) and (hi>1)    then Intercambia(p[hi,hj],p[hi-1,hj  ]);
64   if (d=2) and (hi<SIZE) then Intercambia(p[hi,hj],p[hi+1,hj  ]);
65   if (d=3) and (hj>1)    then Intercambia(p[hi,hj],p[hi  ,hj-1]);
66   if (d=4) and (hj<SIZE) then Intercambia(p[hi,hj],p[hi  ,hj+1]);
67 end;
68
69 procedure CreaPuzle(var p: tpPuzle);
70 var
71   i,dir : integer;
72 begin
73   Inicializa(p);
74
75   for i:=1 to 100 do
76   begin
77     dir := 1+random(4);
78     MueveHueco(p,dir);
79   end;
80 end;
```

Ejercicio 2 **[3.5 puntos]**

La Confederación Hidrográfica del Ebro monitoriza constantemente el caudal del río a su paso por la ciudad de Zaragoza, en m^3/s. Cada día, durante las 24 horas del mismo, se toma una medición del caudal **cada minuto**. Los valores medidos se guardan en un **fichero secuencial** de **números reales**, generándose uno de estos ficheros cada día.
Como la medida de caudal por minuto podría ser ruidosa (esto es, podría fluctuar mucho), se desea calcular el caudal medio por hora a partir de las medidas de caudal por minuto, obteniéndose 24 medidas de caudal en un día.
Además, es de interés monitorizar los picos en el caudal. Un pico en el caudal es un valor de caudal que es mayor que el valor anterior y que el valor siguiente de la serie temporal (esto es, un máximo local en el caudal).
Dado el siguiente tipo de datos:

```
type
  tpFichero = file of real;
```

se pide desarrollar en Pascal:
a) Un procedimiento que cumpla la siguiente especificación:

```
procedure caudalesPorHora(??? fPorMinuto: tpFichero; ??? fPorHora: tpFichero)
{ El fichero fPorMinuto contiene un valor de caudal por minuto a lo largo de un día.
A partir de la información ya contenida en el fichero fPorMinuto, el procedimiento debe
obtener los valores de caudal medio por hora del día y almacenarlos en fPorHora. }
```

b) Un procedimiento que cumpla la siguiente especificación:

```
procedure mostrarPicosCaudal(??? fPorHora: tpFichero)
{ El fichero fPorHora contiene un valor de caudal por hora a lo largo de un día.
A partir de la información ya contenida en fPorHora, el procedimiento debe
calcular y mostrar por pantalla los picos de caudal. Se mostará un pico por línea,
cada uno con el siguiente formato:
Pico detectado: 503.25 m3/s }
```

Observaciones importantes:

1. No se puede utilizar vectores.
2. Puedes considerar que en los ficheros no falta ningún dato, ni ningún dato es incorrecto.
3. Se valorará reducir el número de veces que se recorren los ficheros al mínimo necesario.
4. A la hora de calcular los picos, se supondrá que los valores extremos de la serie temporal (el primero y el último) no constituyen picos en el caudal (aunque su valor sea mayor que el del valor anterior o posterior de la serie).

Propuesta de solución:

```
type
  tpFichero = file of real;

procedure caudalesPorHora(var fPorMinuto: tpFichero; var fPorHora: tpFichero);
var
  dato, mediaPorHora: real;
  i: integer;
begin
  reset(fPorMinuto);
  rewrite(fPorHora);
  while (not eof(fPorMinuto)) do begin
    mediaPorHora := 0;
    for i:=1 to 60 do begin
      read(fPorMinuto, dato);
      mediaPorHora := mediaPorHora + dato;
    end;
    mediaPorHora := mediaPorHora / 60;
    write(fPorHora, mediaPorHora);
  end;
  close(fPorMinuto); close(fPorHora);
  writeln('Fichero por hora creado con exito.');
end;

procedure mostrarPicosCaudal(var fPorHora: tpFichero);
var
  c1, c2, c3: real;
  pico: real;
begin
  reset(fPorHora);
  read(fPorHora, c1, c2);
  while (not eof(fPorHora)) do begin
    read(fPorHora, c3);
    if (c2 > c1) and (c2 > c3) then begin
      pico := c2;
      writeln('Pico detectado: ', pico:1:2, ' m3/s');
    end;
    c1 := c2;
    c2 := c3;
  end;
  close(fPorHora);
end;
```

Ejercicio 3 **[3.0 puntos]**

En 2009, la UEFA estableció el denominado *Fair Play* Financiero (FFP, del inglés *Financial Fair Play*), como medida para evitar que los clubes profesionales de fútbol gastaran más dinero del que oficialmente tienen. La implementación del FFP es sencilla: los clubes deben tener un balance neto de gastos menos ingresos anuales cercano a cero. Por ello, se permite gastar hasta 5 millones de euros (M €) más de lo que se ingresa, esto es, se tiene que cumplir:

$$\texttt{gastos} - \texttt{ingresos} < 5\texttt{M €} \tag{1}$$

Supondremos que los ingresos anuales de un club (ventas, premios, taquilla, merchandising, patrocinios, derechos de televisión...), vendrán dados por una única cifra agregada. Los gastos anuales de un club son los derivados de los salarios anuales de los jugadores, y los costes de los fichajes realizados ese año.

Pongamos por ejemplo el club hipotético mostrado a la derecha. El club del ejemplo tendrá 220M € en ingresos, y 210M € en gastos en la presente temporada (150M € en salarios, y 60M € en costes de fichaje). De acuerdo a la Inecuación 1 el club cumple el FFP, ya que la diferencia de gastos menos ingresos es menor de 5M € (de hecho, es de −10M €, el club tiene beneficios ese año).

PSG

Jugadores:

Nombre	**Salario**	**Fichaje**	
Leo Messi	50M	50M	
Kylian Mbappé	25M	0M	
Neymar Jr	40M	0M	
Sergio Ramos	20M	10M	
Ander Herrera	15M	0M	
Total Gastos:	150M	60M	**210M**
Total Ingresos:			**220M**

Se pide implementar un conjunto de módulos en Pascal necesarios para evaluar el cumplimiento del FFP, en concreto:

- Un tipo `tpJugador` que represente la información relevante de un jugador; debe incluir: el nombre completo del jugador (una cadena de hasta 255 caracteres), su salario anual y el coste de fichaje. El coste de fichaje de un jugador sólo será mayor que 0 el año en que ha sido fichado, y será 0 en todos los demás años (en el ejemplo, "Leo Messi" y "Sergio Ramos" han sido fichados en ese año, mientras que el resto de jugadores ya estaban fichados y tienen por tanto coste de fichaje 0).

- Un tipo `tpClub` que permita almacenar la situación financiera de un club en un cierto año, incluyendo: los ingresos del club (una única cifra, en el ejemplo arriba eran 220M €), y la lista completa de jugadores, con la información relevante a efectos de gastos. El número máximo de jugadores por equipo es `N=50`.

- Una función que, dado como entrada un club de tipo `tpClub`, devuelva el balance (gastos − ingresos) del mismo, con la siguiente cabecera:

```
function balance(??? club: tpClub): ????
```

- Una función `puedeFichar` que evalúe si un club (de tipo `tpClub`) puede o no fichar a un jugador (de tipo `tpJugador`) cumpliendo el FFP, con cabecera:

```
function puedeFichar(??? club : tpClub; ??? jugador: tpJugador): ????
```

  Para ello, primero comprobará si el jugador ya está en plantilla. Si lo está, no podrá ser fichado. Si no lo está, la función tendrá que evaluar si el total del balance actual del club más los gastos asociados al jugador (su salario anual y coste de fichaje), cumple la Inecuación 1. Puedes asumir que el nombre de cada jugador le identifica de forma inequívoca.

NOTA: Para todos los apartados, se recomienda almacenar y trabajar con los importes en millones de euros (M €). Recordad que se pueden utilizar los operadores de comparación = y <> con el tipo `String`.

Propuesta de solución:

```
program ffp;

const
  N = 50;

type
  tpJugador = record
    nombre  : string;
    ficha   : real; {En M EUR}
    salario : real; {Anual, en M EUR}
  end;

  tpJugadores = array [1..N] of tpJugador;

  tpClub = record
    nombre   : string;
    ingresos : real; {En M EUR}

    jugadores  : tpJugadores;
    njugadores : integer;
  end;


function balance(const club : tpClub): real;
var
  i : integer;
  gastos : real;
begin
  gastos := 0;
  for i:=1 to club.njugadores do
    gastos := gastos + club.jugadores[i].ficha + club.jugadores[i].salario;

  balance := gastos - club.ingresos;
end;


function puedeFichar ( const club : tpClub; const jugador : tpJugador ) : boolean;
var
  i : integer;
  encontrado : boolean;
begin

  encontrado := false;
  for i:=1 to club.njugadores do
    if club.jugadores[i].nombre = jugador.nombre then
      encontrado := true;

  puedeFichar := (not encontrado) and ((balance(club) + jugador.salario + jugador.ficha) < 5);
end;


var
{-  PSG: tpClub = (
    nombre : 'PSG';
    ingresos: 200;
    jugadores : (
      (nombre: 'Leo Messi'; ficha : 50; salario : 50)
      );
```

```
    njugadores : 0;
  ); -}

  PSG : tpClub;
  js : tpJugadores;
  j : tpJugador;
begin
  PSG.nombre := 'PSG'; PSG.ingresos := 220;
  PSG.jugadores[1].nombre := 'Leo Messi'; PSG.jugadores[1].ficha := 50; PSG.jugadores[1].salario := 50;
  PSG.jugadores[2].nombre := 'Mbappe'; PSG.jugadores[2].ficha := 25; PSG.jugadores[2].salario := 0;
  PSG.jugadores[3].nombre := 'Neymar'; PSG.jugadores[3].ficha := 40; PSG.jugadores[3].salario := 0;
  PSG.jugadores[4].nombre := 'Sergio Ramos'; PSG.jugadores[4].ficha := 20; PSG.jugadores[4].salario := 10;
  PSG.jugadores[5].nombre := 'Ander Herrera'; PSG.jugadores[5].ficha := 15; PSG.jugadores[5].salario := 0;
  PSG.njugadores := 5;


  writeln('Balance: ', balance(PSG):0:2);

  j.nombre := 'Leo Messi'; j.ficha := 0; j.salario := 1;
  writeln('Puede fichar a ', j.nombre, '?:', puedeFichar(PSG, j));
  j.nombre := 'Nayim'; j.ficha := 5; j.salario := 6;
  writeln('Puede fichar a ', j.nombre, '?:', puedeFichar(PSG, j));
  j.nombre := 'Nayim'; j.ficha := 50; j.salario := 6;
  writeln('Puede fichar a ', j.nombre, '?:', puedeFichar(PSG, j));
end.
```