Departamento de
Informática e Ingeniería
de Sistemas
Universidad Zaragoza
1474

Examen de Teoría – Soluciones

21 de enero de 2021

Fundamentos de Informática
Grado en Tecnologías Industriales
Duración: 3 horas

Ejercicio 1 **[3.5 puntos]**

En los teléfonos móviles antiguos, en los móviles especiales para personas mayores o en los teclados de ciertos sistemas dedicados, la introducción de texto se hace con el mismo teclado numérico, mediante la pulsación repetida de las teclas numéricas. Por ejemplo, cuando estamos en modo texto y pulsamos repetidamente la tecla '2':

- Una pulsación representa el carácter 'A'.
- Dos pulsaciones representan el carácter 'B'.
- Tres pulsaciones representan el carácter 'C'.
- Cuatro pulsaciones representan el carácter '2'.

Una posible forma de implementar el sistema sería recogiendo la lista de pulsaciones de las teclas con sus códigos numéricos y traduciendo. Los grupos de pulsacione múltiples se separan mediante algún carácter especial, por ejemplo un punto `'.'`.

Asi, las pulsaciones detectadas como:

```
33.7777.8.666.0.33.7777.0.88.66.2.0.7.777.88.33.22.2.
```

se traducirían como

```
ESTO ES UNA PRUEBA
```

Implementa en Pascal las siguientes tareas:

- Define una tipo de datos denominado `tpTeclado` que permita representar los caracteres asociados a cada una de las teclas del teléfono. Nos centraremos sólo en las teclas numéricas del `'0'` al `'9'`.
- Define una estructura de datos `tpPulsacionMultiple` que almacene la información de una pulsación múltiple (obviamente): la tecla y el número de repeticiones.
- Define una función que a partir de una `tpPulsacionMultiple` y un `tpTeclado` te devuelva el carácter que representa.

```
function decodifica(???? pm: tpPulsacionMultiple;
                    ???? teclado: tpTeclado): char;
```

- Escribe un programa que lea del teclado una línea de pulsaciones (similar a la del ejemplo), y, usando la función anterior, escriba el texto correspondiente.

NOTA: Puedes utilizar el tipo de datos `string` si lo consideras necesario.

Propuesta de solución:

```
program telefono;
type
  tpTeclado = array['0'..'9'] of string[5];
  tpPulsacionMultiple = record
    tecla : char;
    reps  : integer;
  end;

function decodifica(const pm: tpPulsacionMultiple;
         const tec: tpTeclado): char;
begin
  if ('0'<=pm.tecla) and (pm.tecla<='9')
  then
    decodifica := tec[pm.tecla][pm.reps mod length(tec[pm.tecla])]
  else
    decodifica := pm.tecla;
end;

var
  teclado: tpTeclado = (
  ' 0','1','ABC2','DEF3','GHI4','JKL5','MNO6','PQRS7','TUV8','WXYZ9'
  );

var
  dummy,c: char;
  pm : tpPulsacionMultiple;
begin
  while not eoln do
  begin
    read(pm.tecla);
    pm.reps := 1;
    read(dummy);
    while dummy<>'.' do
    begin
      pm.reps := pm.reps + 1;
      read(dummy);
    end;
    c := decodifica(pm,teclado);
    write(c);
  end;
  readln;
  writeln;
end.
```

Ejercicio 2 **[3.0 puntos]**

Durante la primera etapa del proceso de vacunación en los centros de salud pertenecientes a un sector sanitario de Zaragoza, los centros van registrando el número de vacunas administradas cada día de vacunación. Lo hacen en un **fichero secuencial de registros**, de nombre `'vacunacionPrimeraEtapa.dat'`. Cada registro de dicho fichero contiene: el código numérico del centro (un número de 1 a 10, porque hay 10 centros en el sector), la fecha (en formato AAAAMMDD), y el número de vacunas administradas por el centro en esa fecha. Las estructuras de datos correspondientes son las siguientes:

```
1  tpVacUnDia = record
2      centro: integer;
3      fecha:  longint;
4      numVac: integer;
5  end;
6  tpFichVacunas = file of tpVacUnDia;
```

Se pide desarrollar un programa Pascal que:

a) A partir de la información contenida en el fichero `'vacunacionPrimeraEtapa.dat'`, muestre por pantalla el número total de vacunas administradas en la primera etapa en ese sector sanitario ($nTot$), así como la media de vacunaciones por centro (esto es, el total de vacunas administradas entre el número de centros).

b) De cara a la segunda etapa de vacunación se dispone de $nDisp$ dosis. El reparto de estas dosis entre centros se va a hacer de forma proporcional a las vacunas ya administradas por cada centro. Si el centro i ha administrado un total de vac_i dosis en la primera etapa, en la segunda etapa le corresponderán $vac_i/nTot * nDisp$ dosis. Así, el programa pedirá al usuario el número de dosis disponibles para la segunda etapa ($nDisp$), y mostrará por pantalla cuántas dosis le corresponden a cada centro. Si se obtiene un número no entero de dosis a repartir para un centro, se mostrará el entero inmediatamente inferior.

Observaciones:

1. Los registros no están ordenados en el fichero, ni por fecha ni por centro ni de ningún otro modo, ya que cada centro tiene sus propios protocolos sobre cuándo incluir los datos de vacunación en el fichero.
2. Se pueden definir nuevas estructuras de datos si se considera apropiado.
3. El número total de vacunas administradas, así como el número de dosis disponibles para la segunda etapa, pueden almacenarse ("caben") en un dato de tipo entero.
4. Se valorará reducir el número de veces que se recorre el fichero al mínimo necesario.
5. Se muestra a continuación un ejemplo de fichero y un ejemplo de ejecución correspondiente a dicho fichero; en el ejemplo de ejecución, figura en negrita la información introducida por el usuario.

Ejemplo de fichero `'vacunacionPrimeraEtapa.dat'`:

centro = 1 fecha = 20210119 numVac = 200	centro = 2 fecha = 20210118 numVac = 250	centro = 1 fecha = 20210115 numVac = 300	centro = 5 fecha = 20210119 numVac = 500

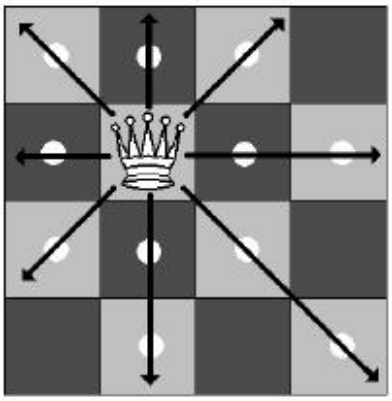
Ejemplo de ejecución:

Administradas 1250 vacunas en el sector.
Media por centro = 125.00 vacunas/centro.
Introduzca número vacunas disponibles para segunda etapa:
14000
Vacunas a distribuir por centro:
1 - 5600
2 - 2800
3 - 0
4 - 0
5 - 5600
6 - 0
7 - 0
8 - 0
9 - 0
10 - 0

Propuesta de solución:

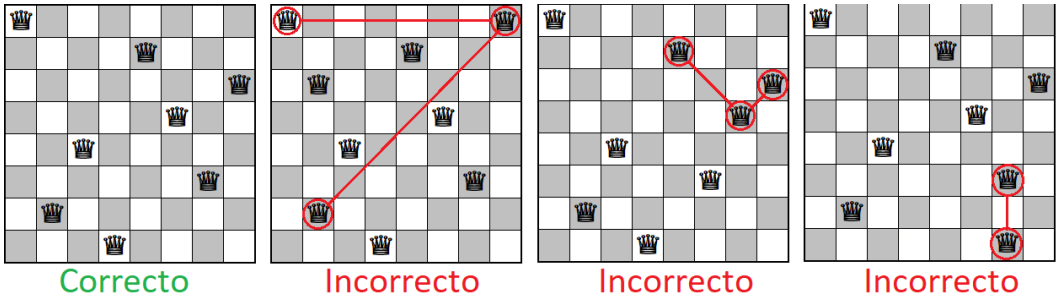
```
program analizaVacunas;
const
  NCENTROS = 10;
type
  tpVacUnDia = record
    centro: integer;
    fecha: longint;
    numVacunas: integer;
  end;
  tpFichVacunas = file of tpVacUnDia;
  tpCentros = array[1..NCENTROS] of integer;

var
  fich: tpFichVacunas;
  vac: tpVacUnDia;
  vec: tpCentros; { num vacunas admin por cada centro }
  nTot: integer;
  media: real;
  nDisp: integer;
  i: integer;
begin
  assign(fich, 'vacunacionPrimeraEtapa.dat');
  reset(fich);
  { Inicializar vector }
  for i := 1 to NCENTROS do
    vec[i] := 0;
  nTot := 0;

  while not eof(fich) do
  begin
    read(fich, vac);
    nTot := nTot + vac.numVacunas;
    vec[vac.centro] := vec[vac.centro] + vac.numVacunas;
  end;
  close(fich);
  media := nTot / NCENTROS;
  writeln('Se han administrado ', nTot, ' vacunas en este sector sanitario.');
  writeln('La media por centro es de ', media:1:2, ' vacunas/centro.');

  writeln('Introduce num. vacunas disponibles para la segunda etapa: ');
  readln(nDisp);
  writeln('Vacunas a distribuir por centro: ');
  for i := 1 to NCENTROS do
    writeln(i, ' - ', trunc(vec[i]/nTot*nDisp));

end.
```

Ejercicio 3 **[3.5 puntos]**



El **problema de las ocho reinas** es un pasatiempo que consiste en poner ocho reinas en el tablero de ajedrez sin que se amenacen. Fue propuesto por el ajedrecista alemán Max Bezzel en 1848. En el juego del ajedrez, el tablero es una cuadrícula de 8×8, y la reina amenaza a aquellas piezas que se encuentren en su misma fila, columna o diagonal. La figura de la derecha muestra todos los movimientos posibles de una reina en un tablero de 4×4.

Se considera una respuesta correcta al problema de las ocho reinas **si ninguna de ellas se amenaza entre sí**. En la figura inmediatamente inferior se muestra cuatro potenciales soluciones al problema. El tablero de la izquierda sería **una solución correcta**, ya que ninguna de las reinas se amenaza entre sí. Por contra, los otros tres ejemplos serían **respuestas incorrectas**, ya que al menos un par de reinas (marcadas en el tablero) están en posición de amenaza.



Se pide, usando el lenguaje Pascal, implementar los siguientes puntos:

- Define un tipo de datos para representar un tablero con una serie de reinas colocadas en él:

```
tpTablero = ........
```

- Implementa una función en Pascal que evalúe si un tablero es una solución correcta o no al problema de las 8 reinas:

```
function esSolucionCorrecta(????? tb: tpTablero): ????;
```

Se pueden definir tantas estructuras de datos y funciones/procedimientos como se estime necesario.
Se valorará el uso de procedimientos y funciones adecuados.
Se valorará reducir al mínimo el número de veces que se recorre el tablero.

Propuesta de solución:

```
program OchoReinas;

const
    N = 8;
type
    tpTablero = array[1..N, 1..N] of boolean;

function amenazaEnDireccion( const tablero : tpTablero;
               i : integer; j : integer;
               di : integer; dj: integer): boolean;
var
  ii, jj: integer;
  amenaza : boolean;
begin
  ii := i+di;
  jj := j+dj;
  amenaza := false;
  while (ii > 0) and (ii <= N) and (jj > 0) and (jj<=N) and (not amenaza) do
  begin
    amenaza := tablero[ii,jj];
    ii := ii+di;
    jj := jj+dj;
  end;

  amenazaEnDireccion := amenaza;
end;

function amenaza( const tablero : tpTablero; i : integer; j : integer): boolean;
var
  di, dj : integer;
begin
  amenaza := false;
  for di:=-1 to 1 do
    for dj :=-1 to 1 do
      if (di <> 0) or (dj <> 0) then
        if amenazaEnDireccion( tablero, i, j, di, dj) then
          amenaza := true;
end;

function esSolucionCorrecta(const tablero : tpTablero): boolean;
var
  i,j: integer;
begin
  esSolucionCorrecta := true;
  for i:=1 to N do
    for j:=1 to N do
      if tablero[i,j] then
        if amenaza( tablero, i, j ) then
          esSolucionCorrecta := false;
end;

begin
end.
```