Departamento de
Informática e Ingeniería
de Sistemas
Universidad Zaragoza
1474

Examen de Teoría – Soluciones

17 de enero de 2023

Fundamentos de Informática
Grado en Tecnologías Industriales
Duración: 3h

Ejercicio 1 **[3.0 puntos]**

Se desea disponer de un programa que permita implementar distintos juegos de cartas con una baraja. En concreto, el programa utilizará la baraja española de 48 naipes; las cartas de esta baraja son de uno de los cuatro palos (oros, copas, espadas y bastos) y tienen números del 1 al 12 (esto es, incluye las cartas del 8 y del 9, no pasa del 7 al 10 como las barajas de 40 naipes). Para dicho programa, **se pide**, en lenguaje Pascal:

- Definir un tipo de dato **`tpCarta`** que nos permita representar una carta de la baraja, almacenando la información sobre su número (del 1 al 12) y su palo (oros, copas, espadas o bastos, con la representación que consideres adecuada).
- Definir un tipo de dato **`tpMazo`** que nos permita representar un mazo de cartas, que pueden estar en cualquier orden, y del que se podrán ir "extrayendo" cartas. La extracción de cartas se hará en el orden en que están en el mazo, de forma que el **`tpMazo`** debe permitir almacenar:
  - El conjunto completo de las 48 cartas.
  - El índice de la siguiente carta que toca extraer del mazo.

  Así, extraer una carta es simplemente tomar la siguiente carta que toca, en el orden en que estén guardadas (no se quita la carta del mazo).

A partir de esos tipos de dato, **se pide** implementar en lenguaje Pascal los siguientes procedimientos:

- Un procedimiento que inicialice el mazo de cartas:

  ```
  procedure inicializar(??? mazo: tpMazo);
  ```

  Para ello, deberá introducir en el mazo todas las cartas de la baraja de forma ordenada (los cuatro palos en orden: oros, copas, espadas y bastos, y dentro de cada palo, las cartas ordenadas del 1 al 12). La siguiente carta a extraer será la primera del mazo.

- Un procedimiento que baraje el mazo de cartas:

  ```
  procedure barajar(??? mazo: tpMazo; ??? nveces: integer);
  ```

  Para ello, se debe repetir `'nveces'` el algoritmo conocido como *shuffle*, algoritmo en el que cada carta i del mazo se intercambia con otra elegida aleatoriamente entre las que hay desde su posición i hasta el final del mazo. Para seleccionar una carta aleatoria puedes usar la función de Pascal estándar (ya existente, no tienes que implementarla):

  ```
  function random(maxval: longint): longint;
  ```

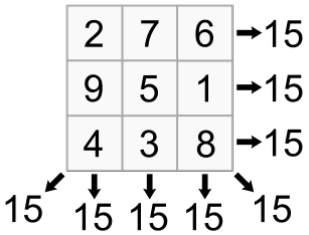
  Esta función toma como parámetro un entero (`maxval`, de tipo `longint`) y devuelve un entero aleatorio (`longint`) que está en el rango `[0..maxval`).

Nota: No hay que implementar un programa completo, sólo los tipos de dato y procedimientos que se piden.

Propuesta de solución:

```
program cartas;

type
  tpPalo = 1..4;
  // tpPalo = (Oros,Copas,Espadas,Bastos);
  tpCarta = record
    n: integer;
    p: tpPalo;
  end;
  tpMazo = record
    cartas : array[1..48] of tpCarta;
    sig    : integer;
  end;

procedure inicializar(var mazo: tpMazo);
var
  n,c : integer;  p : tpPalo;
begin
  n := 1;
  //for p:=Oros to Bastos do
  for p:=1 to 4 do
  begin
    for c:=1 to 12 do
    begin
      mazo.cartas[n].p := p;
      mazo.cartas[n].n := c;
      inc(n);
    end;
  end;
  mazo.sig := 1;
end;

procedure shuffle(var m: tpMazo);
var
  i,j : integer;  aux : tpCarta;
begin
  for i:=1 to 47 do
  begin
    j := i + (random(48-i)+1);
    aux := m.cartas[i];
    m.cartas[i] := m.cartas[j];
    m.cartas[j] := aux;
  end;
end;

procedure barajar(var m: tpMazo; nveces: integer);
var
  v : integer;
begin
  for v:= 1 to nveces do
    shuffle(m);
end;
```

Ejercicio 2 **[3.5 puntos]**



Dada una matriz cuadrada de enteros de tamaño $n \times n$, se dice que es un **cuadrado mágico** si contiene todos los números enteros del 1 al n^2, sin repeticiones, y el resultado de sumar cada fila, el de sumar cada columna, y el de sumar las diagonales principales, es siempre el mismo (ver ejemplo de la figura, en el que la matriz es un cuadrado mágico y dicho resultado es 15).

El valor de dicho resultado para un cuadrado mágico se denomina *constante mágica* (C). Puede deducirse matemáticamente a partir del tamaño n, y viene dado por la siguiente fórmula:

$$C = \frac{n(n^2+1)}{2}$$

Se pide, usando el lenguaje Pascal:

- Definir un tipo de dato **`tpMatCuad`** que permita almacenar una matriz cuadrada de enteros, de cualquier tamaño de lado n, hasta un máximo de 32.
- A partir del tipo de dato anterior, implementar la siguiente función:

  ```
  function esCuadradoMagico(??? mat: tpMatCuad): boolean;
  ```

  La función debe devolver un valor booleano indicando si el dato pasado como parámetro es un cuadrado mágico o no. Para serlo, debe cumplir las siguientes condiciones:
  - Las dos diagonales principales suman lo mismo (C).
  - Todas las filas suman lo mismo (C).
  - Todas las columnas suman lo mismo (C).

Se valorará el uso adecuado de procedimientos y funciones en la solución, así como que no exista código repetido innecesariamente. Se asume que la matriz es cuadrada, no hay que comprobar nada al efecto.

Propuesta de solución:

```
const
  SIZE = 32;

type
  tpMatriz = record
    nFil, nCol: 1..SIZE; { con uno solo es suficiente }
    dat: array[1..SIZE, 1..SIZE] of integer;
  end;

function sumaDiag1(const m: tpMatriz): integer;
var
  i: integer;
  suma: integer;
begin
  suma := 0;
  for i:=1 to m.nFil do
    suma := suma + m.dat[i,i];
  sumaDiag1 := suma;
end;

function sumaDiag2(const m: tpMatriz): integer;
var
  i: integer;
  suma: integer;
begin
  suma := 0;
  for i:=1 to m.nFil do
    suma := suma + m.dat[i,m.nCol+1-i];
  sumaDiag2 := suma;
end;

function sumaFila(const m: tpMatriz; f: integer): integer;
var
  i: integer;
  suma: integer;
begin
  suma := 0;
  for i:=1 to m.nCol do
    suma := suma + m.dat[f,i];
  sumaFila := suma;
end;

function sumaColumna(const m: tpMatriz; c: integer): integer;
var
  i: integer;
  suma: integer;
begin
  suma := 0;
  for i:=1 to m.nFil do
    suma := suma + m.dat[i,c];
  sumaColumna := suma;
end;

function esCuadradoMagico(const m: tpMatriz): boolean;
var
  fil, col: integer;
  esCuadrada, cond1, cond2, cond3, cond4: boolean;
  constanteMagica: integer;
begin
```

```
60    cond1 := false; cond2 := false; cond3 := false; cond4 := false;
61    { Cond0: cuadrada – No era necesario comprobarlo }
62    esCuadrada := m.nFil = m.nCol;
63    if esCuadrada then
64    begin
65      constanteMagica := ((m.nFil * m.nFil + 1)*m.nFil) div 2;
66      { Cond1: diag principal }
67      cond1 := sumaDiag1(m) = constanteMagica;
68      if cond1 then
69      begin
70        { Cond2: diag principal 2 }
71        cond2 := sumaDiag2(m) = constanteMagica;
72        if cond2 then
73        begin
74          { Cond3: Todas las filas han de sumar constanteMagica }
75          cond3 := true;
76          fil := 1;
77          while (fil <= m.nFil) and cond3 do
78          begin
79            if (sumaFila(m,fil) <> constanteMagica) then cond3 := false;
80            fil := fil + 1;
81          end;
82          if cond3 then
83          begin
84            { Cond4: Todas las columnas han de sumar constanteMagica }
85            cond4 := true;
86            col := 1;
87            while (col <= m.nCol) and cond4 do
88            begin
89              if (sumaColumna(m,col) <> constanteMagica) then cond4 := false;
90              col := col + 1;
91            end;
92          end;
93        end;
94      end;
95    end;
96    esCorrecto := cond1 and cond2 and cond3 and cond4;
97  end;
```

Ejercicio 3 **[3.5 puntos]**

El alfabeto Morse fue desarrollado por Alfred Vail y Samuel Morse en 1830 con la invención del telégrafo eléctrico. En esencia, es un sistema de representación de letras y números mediante códigos consistentes en rayas y puntos (es decir, señales telegráficas intermitentes que se diferencian en el tiempo de duración de la señal: larga o corta). La Figura 1 muestra la codificación en Morse de las letras mayúsculas.

A	·−	J	·−−−	S	···
B	−···	K	−·−	T	−
C	−·−·	L	·−··	U	··−
D	−··	M	−−	V	···−
E	·	N	−·	W	·−−
F	··−·	O	−−−	X	−··−
G	−−·	P	·−−·	Y	−·−−
H	····	Q	−−·−	Z	−−··
I	··	R	·−·		

Figura 1: Telégrafo y tabla de codificación Morse de las letras mayúsculas.

Para este ejercicio deberás crear un programa que reciba un fichero de texto con la información codificada en Morse y muestre por pantalla el mensaje decodificado. Para simplificar, se asumirá que el mensaje decodificado sólo contendrá letras mayúsculas.

Para poder decodificar el mensaje, se dispone de un fichero secuencial de registros de tipo **`tpCorresp`**, llamado `'morse.dat'`, en el que cada registro contiene la codificación Morse de una letra mayúscula, utilizando dos campos: el campo ch, que contiene la letra, y el campo cod, que contiene el código Morse correspondiente a dicha letra, representado con el tipo **`tpMorse`**. En Pascal:

```
tpMorse = array[1..5] of char;
tpCorresp = record
 ch:  char;
 cod: tpMorse;
end;
```

El código Morse de una letra, representado con el tipo **`tpMorse`**, tiene una longitud máxima de 5 caracteres; si el código de una letra requiere de menos de 5 caracteres, el resto serán espacios en blanco (`' '`).
A partir de dichos tipos de dato, **se pide**, usando el lenguaje Pascal:

- Definir el tipo de datos **`tpTablaMorse`**, que permita almacenar la traducción de todas las letras a su correspondiente código Morse.
- Desarrollar un procedimiento `cargarCodigo` que lea del **fichero secuencial de registros** `'morse.dat'` los códigos Morse correspondientes a cada letra, y los guarde en una estructura de datos de tipo `tpTablaMorse` suministrada como parámetro.

  ```
  procedure cargarCodigo(??? morse: tpTablaMorse);
  ```

  Se asumirá que el fichero es correcto y completo, esto es, almacena un único código Morse por cada letra mayúscula del alfabeto. Sin embargo, los datos del fichero `'morse.dat'` no tienen por qué seguir ningún orden (e.g., no están en orden alfabético).
- Desarrollar un programa que pida y lea de teclado el nombre de un **fichero de texto** que contiene un mensaje codificado en Morse, y muestre por pantalla el resultado de la decodificación. Notas:
  - El programa hará uso del procedimiento `cargarCodigo` (y el fichero secuencial `'morse.dat'`) para obtener la correspondencia entre las mayúsculas y su código Morse.
  - En el fichero de texto, cada letra codificada está separada de la siguiente por el carácter `'/'` (ver un ejemplo en la Figura 2).
  - El fichero de texto puede tener varias líneas, en cuyo caso se mostrará el mensaje codificado también separado en esas líneas.

- Para simplificar, en el mensaje codificado no habrá ningún carácter que no sean letras mayúsculas (no habrá espacios en blanco, ni signos de puntuación), y por tanto tampoco tiene que haberlos en el mensaje decodificado (ver ejemplo de ejecución en Figura 3).
- Puedes comparar dos datos de tipo **`tpMorse`** utilizando el operador de igualdad `'='`, que devuelve true si ambos datos son iguales, y false en caso contrario.

```
..../---/.-../.-/--/..-/-./-../---/
./.../-/---/./.../..-/-./.-/.--./.-./..-/./-.../.-/
..../.-/.../-/.-/.-../..-/./--./---/
```

Figura 2: Ejemplo de fichero de texto `'mensaje.txt'`, escrito en código Morse. El mensaje sólo contiene códigos Morse de letras mayúsculas, separados entre sí por el carácter `'/'`.

```
Introduce el nombre del fichero: mensaje.txt ⏎
Mensaje:
HOLAMUNDO
ESTOESUNAPRUEBA
HASTALUEGO
```

Figura 3: Ejemplo de ejecución. En negrita se muestra el texto introducido por el usuario.

Propuesta de solución:

```
program morse;

type
  tpMorse = array[1..5] of char;
  tpTrad = record
    ch:  char;
    cod: tpMorse;
  end;
  tpTablaMorse = array['A'..'Z'] of tpMorse;
  tpFichCodigos = file of tpTrad;

procedure cargarCodigo(var morse: tpTablaMorse);
var
  f: tpFichCodigos;
  d: tpTrad;
begin
  assign(f, 'morse.dat');
  reset(f);
  while not eof(f) do
  begin
    read(f, d);
    morse[d.ch] := d.cod;
  end;
  close(f);
end;

var
  tablaMorse: tpTablaMorse;
  nomFich: string;
  t: text;
  c, cTrad, car: char;
  m: tpMorse;
  i, j: integer;
begin
  cargarCodigo(tablaMorse);
  write('Introduce el nombre del fichero: ');
  readln(nomFich);
  writeln('Mensaje: ');
  assign(t, nomFich);
  reset(t);

  while not eof(t) do
  begin
    while not eoln(t) do
    begin
      { Lee bloque codigo morse }
      for j:=1 to 5 do m[j] := ' ';
      i := 1;
      read(t, c);
      while (c <> '/') do
      begin
        m[i] := c;
```

```
53          i := i + 1;
54          read(t,c);
55        end;
56
57        { Buscar codigo Morse en tabla }
58        for car:='A' to 'Z' do
59          if (tablaMorse[car] = m) then
60            cTrad := car;
61        write(cTrad);
62      end;
63      readln(t);
64      writeln();
65    end;
66    close(t);
67  end.
```