Práctico 10 - Enumerados, Conjuntos y Registros

Programación 1
InCo - Facultad de Ingeniería, Udelar

1. Determine cuáles de las siguientes declaraciones son válidas y explique por qué las restantes no lo son.

☐
```
type Letra = ('X', 'Y', 'Z')
```
√
```
type Lenguaje = (Pascal, Fortran, Basic)
```
☐
```
type Codigo = (1, 2, 3, 4, 5)
```
√
```
type Codigo = (c1, c2, c3, c4, c5)
```
☐
```
procedure encontrar (var ciudad :
    (Minas, Florida, Flores))
```
√
```
type Trabajo = (obrero, oficinista);
...
procedure buscar (var empleo : Trabajo);
```

2. Dadas las siguientes declaraciones, determine cuáles de los siguientes fragmentos de código son válidos y explique por qué los restantes no lo son.

```
type color = (rojo, blanco, azul, purpura);
var coloracion : color;
```

☐
```
read (rojo);
    write (rojo)
```
√
```
coloracion := blanco;
case coloracion of
    rojo   : write ('rojo');
    blanco : write ('blanco');
    azul   : write ('azul');
    purpura: write ('purpura')
end
```
☐
```
read (coloracion);
write (coloracion)
```
☐
```
coloracion := blanco;
write (coloracion)
```
√
```
if coloracion = azul then
    write ('azul')
else
    write ('no azul')
```

3. Indique qué sucederá al ejecutar el siguiente programa.

```
program Ejercicio3;
type Asignatura = (matematica, historia, computacion, geografia, fisica);
var a, b: Asignatura;
begin
    a := matematica;
    b := computacion;
    if a > b then
        write ('Magnifico')
    else
        write ('Excelente')
end.
```

Excelente

4. Dadas las siguientes declaraciones:

```
type TipoDia = (lunes, martes, miercoles, jueves, viernes, sabado, domingo);
var dia : TipoDia;
    laborable : lunes..viernes;
    finsemana : sabado..domingo;
```

Indique cuáles de las siguientes asignaciones producirán error de rango y explique por qué:

☐ `laborable := pred(sabado)`

√ `finsemana := martes`

√ `dia := succ(domingo)`

☐ `dia := laborable`

5. Indique cuáles de las siguientes declaraciones de tipos subrango son válidas y explique por qué las restantes no lo son:

☐ `type Tipo1 = '0'..9`

√ `type Tipo2 = -3..6`

☐ `type Tipo3 = -1..-3`

☐ `type Tipo4 = 0.0..9.0`

√ `type Tipo5 = 1..1`

√ `type Tipo6 = false..false`

6. (a) Defina el tipo enumerado `TipoMes` que represente los meses del año.

```
type TipoMes = (enero, febrero, marzo, abril, mayo, junio, julio,
    agosto, setiembre, octubre, noviembre, diciembre);
```

(b) Escriba un procedimiento que reciba un parámetro de entrada del tipo `TipoMes` e imprima el nombre completo del mes correspondiente.

```
procedure mostrarMes (mes: TipoMes);
begin
  case mes of
    enero: writeln('Enero');
    febrero: writeln('Febrero');
    marzo: writeln('Marzo');
    abril: writeln('Abril');
    mayo: writeln('Mayo');
    junio: writeln('Junio');
    julio: writeln('Julio');
    agosto: writeln('Agosto');
    setiembre: writeln('Setiembre');
    octubre: writeln('Octubre');
    noviembre: writeln('Noviembre');
    diciembre: writeln('Diciembre')
  end
end;
```

(c) Defina un tipo subrango de los enteros `TipoMesRango` que represente los meses del año (entre 1 y 12).

```
type TipoMesRango = 1..12;
```

(d) Escriba un procedimiento que lea de la entrada un mes representado por el número (entre 1 y 12) y retorne en un parámetro de salida una variable de tipo `TipoMes`.

```
procedure leerMes (var mes: TipoMes);
var numeroMes: TipoMesRango;
begin
  write('Ingrese un mes entre 1 y 12: ');
  readln(numeroMes);
  mes := TipoMes(numeroMes - 1)
end;
```

7. Dadas las siguientes declaraciones:

```
type vocal = (a, e, i, o, u);
var letra : vocal;
    uncar : char
```

Determine si los siguientes fragmentos de programa producirán o no error en tiempo de ejecución. Explique.

I)
```
letra := a;
while letra <= u do
begin
    read (uncar);
    writeLn ('El caracter leido es: ',
        uncar);
    letra := succ (letra)
end
```

II)
```
letra := u;
repeat
    read (uncar);
    writeLn ('El caracter leido es: ',
        uncar);
    letra := pred (letra)
until letra = a;
```

I) Cuando `letra = u`, al invocar `succ` con el parámetro `letra` se produce un error de rango.

II) Cuando `letra = a` se termina la instrucción repeat y no produce ningún error.

8. (a) Enumere los elementos de cada uno de los siguientes conjuntos:

I) `[0, 1, 2, 3]`
II) `[0..10, 13, 15]`
III) `['a'..'m','A'..'M']`
IV) `['0'..'5','9']`

I) 0, 1, 2, 3
II) 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 13, 15
III) ‘A’, ‘B’, ‘C’, ‘D’, ‘E’, ‘F’, ‘G’, ‘H’, ‘I’, ‘J’, ‘K’, ‘L’, ‘M’, ‘a’, ‘b’, ‘c’, ‘d’, ‘e’, ‘f’, ‘g’, ‘h’, ‘i’, ‘j’, ‘k’, ‘l’, ‘m’
IV) ‘0’, ‘1’, ‘2’, ‘3’, ‘4’, ‘5’, ‘9’

(b) Determine el valor de cada una de las siguientes expresiones booleanas:

I) `'A' in ['B'..'S']`
II) `0 in [-3..5]`
III) `7 in [1..4,6..8]`
IV) `' ' in []`

I) `false`
II) `true`
III) `true`
IV) `false`

9. Determine cuáles de las siguientes definiciones de tipos registro son válidas y explique por qué las restantes no lo son:

√
```
type Estudiante = record
                    nombre, direccion :
                     array[1..30] of char;
                    prueba, examen : 0..100;
                    tipoest : (grado, posgrado)
                  end
```

□
```
type Persona = record
                 nombre : array[1..30] of char;
                 edad : 0..120;
                 sexo : (Fem, Masc)
```

□
```
type Calificacion = record
                      prueba : 0..100;
                      nota : 1..12;
                      prueba : 0..100
                    end
```

√
```
type Evaluacion = record
                    prueba : 0..100;
                    nota : 0..12;
                    orden : 1..100
                  end
```

10. Dadas las siguientes declaraciones:

```
type Fecha = record
               mes : 1..12;
               dia : 1..31;
               anio : 0..2100;
```

```
                bisiesto : (si, no)
            end;
var tiempo : Fecha;
```

Determine cuáles de las siguientes asignaciones son válidas y explique por qué las restantes no lo son:

- √ `tiempo.mes := 12`
- ☐ `tiempo.fecha := 3-15-1900`
- ☐ `tiempo.fecha := mes`
- √ `tiempo.anio := 2002`
- ☐ `tiempo.bisiesto := 1`
- √ `tiempo.dia := tiempo.mes`
- ☐ `tiempo.tiempo := tiempo`

11. Los números complejos tienen dos componentes, una parte real y una parte imaginaria. Cada una de las componentes se representa mediante un número real. La siguiente declaración permite representar un número complejo:

```
type
   Complejo = record
                  re, im : Real;
              end;
```

(a) Escriba el procedimiento `sumaComplejos` que almacena en `c3` la suma de los números complejos `c1` y `c2`.

`procedure sumaComplejos (c1, c2 : Complejo; VAR c3 : Complejo);`

Si `re1` e `im1` representan los componentes de `c1` y `re2` e `im2` representan los componentes de `c2`, entonces los componentes de `c3` están dados por:

```
re3 = re1 + re2
im3 = im1 + im2
```

(b) Escriba el procedimiento `multComplejos` que almacena en `c3` la multiplicación de los números complejos `c1` y `c2`.

`procedure multComplejos (c1, c2 : Complejo; var c3 : Complejo);`

Si `re1` e `im1` representan los componentes de `c1` y `re2` e `im2` representan los componentes de `c2`, entonces los componentes de `c3` están dados por:

```
re3 = re1 * re2 - im1 * im2
im3 = im1 * re2 + im2 * re1
```

(c) Escriba un programa principal que lea dos números complejos y exhiba el resultado de su suma y multiplicación. Puede declarar subprogramas auxiliares que le permita cargar e imprimir un número complejo.

```
program complejos;

type
    Complejo = record
        re, im : real;
    end;
var
    c1, c2, resu : Complejo;

{Parte a}
procedure sumaComplejos (c1, c2 : Complejo; var c3 : Complejo);
begin
    c3.re:= c1.re + c2.re;
    c3.im:= c1.im + c2.im
end;
```

```
{Parte b}
procedure multComplejos (c1, c2 : Complejo; var c3 : Complejo);
begin
    c3.re := c1.re * c2.re - c1.im * c2.im;
    c3.im := c1.re * c2.im + c1.im * c2.re
end;

(* Procedimiento auxiliar que carga un numero complejo de la entrada *)
procedure cargar (var numero: Complejo);
begin
    write('Ingrese la parte real: ');
    readln(numero.re);
    write('Ingrese la parte imaginaria: ');
    readln(numero.im)
end;

(* Procedimiento auxiliar que muestra en pantalla un numero complejo *)
procedure mostrar (var numero: Complejo);
begin
    write (numero.re:8:2);
    if (numero.im > 0) then
        write (' + ', numero.im:8:2, 'i')
    else
        if (numero.im < 0) then
            write (' - ', abs(numero.im):8:2, 'i');
    writeLn
end;

begin
    writeLn('Se va a cargar un número complejo');
    cargar(c1);
    writeLn('Se va a cargar otro número complejo');
    cargar(c2);
    suma(c1,c2,resu);
    write('La suma es: ');
    mostrar(resu);
    writeLn;
    multiplicacion(c1,c2,resu);
    write('El producto es: ');
    mostrar(resu);
    writeLn
end.
```

12. Se considera una versión simplificada de la wikipedia que contiene datos de artículos. Se asume que la wikipedia posee `CANT_ARTICULOS` artículos, y que el nombre de los artículos tiene exactamente `CANT_LETRAS` letras. Los idiomas de los artículos pueden ser inglés, portugues y español.

    Para representar esta realidad se definen las siguientes declaraciones:

```
const
   CANT_LETRAS = ...; { valor entero mayor a 0 }
   CANT_ARTICULOS = ...; { valor entero mayor a 0 }

type
   TIdioma = (es, en, pt);

   TFecha = record
        dia : 1..31;
        mes : 1..12;
```

```
    anio : 2001..9999 (* La wikipedia comienza en 2001 *)
end;

TNombre = array [1..CANT_LETRAS] of char;

TArticulo = record
    nombre : TNombre;
    idioma : TIdioma;
    visitas : Integer;
    ultima_act : TFecha;
end;

Wikipedia = array [1..CANT_ARTICULOS] OF TArticulo;
```

(a) Implemente la función `esPosterior` tal que dadas dos fechas `f1` y `f2`, devuelve `TRUE` si la fecha `f1` es posterior que la fecha `f2` y `FALSE` en caso contrario.

`function esPosterior (f1, f2: TFecha) : boolean;`

```
function esPosterior (f1, f2: TFecha) : boolean;
begin
    EsPosterior := (f1.anio > f2.anio) or
        (f1.anio = f2.anio) and (f1.mes > f2.mes) or
        (f1.anio = f2.anio) and (f1.mes = f2.mes) and (f1.dia > f2.dia)
end;
```

(b) Teniendo en cuenta que cada artículo tiene la fecha de su última actualización, implemente el procedimiento `articuloMasReciente` tal que dados la wikipedia y un idioma, devuelve el artículo que tiene la fecha más reciente en el idioma especificado. Asuma que en la wikipedia hay al menos un artículo en el idioma especificado.

`procedure articuloMasReciente (wiki: Wikipedia; idioma: TIdioma; VAR art: TArticulo);`

```
procedure articuloMasReciente (wiki: Wikipedia; idioma: TIdioma;
                                          var art: TArticulo);
var i, j : integer;
begin
    i := 1;
    (* busco el primer articulo del idioma dado. Por el enunciado, se que siempre
       hay al menos uno *)
    while (wiki[i].idioma <> idioma) do
        i := i + 1;
    art := wiki[i];

    (* determino el articulo del idioma dado con mayor fecha de actualizacion *)
    for j := i + 1 to CANT_ARTICULOS do
        if (wiki[j].idioma = idioma) and esPosterior(wiki[j].ultima_act,
                    art.ultima_act) then
            art := wiki[j];
end;
```

(c) Implemente el procedimiento `imprimirArticulosMasRecientes` tal que para cada idioma imprime el nombre del artículo más reciente en dicho idioma, junto con su cantidad de visitas y la fecha de su última actualización. Asuma que en la wikipedia hay al menos un articulo en cada uno de los tres idiomas.

`procedure imprimirArticulosMasRecientes (wiki: Wikipedia);`

Todas las soluciones usarán el procedimiento auxiliar `imprimirArticulo` definido a continuación:

```
        (* Procedimiento auxiliar que imprime un articulo *)
procedure imprimirArticulo (a: TArticulo);
var i : integer;
begin
    (* Imprimo nombre *)
    write ('Nombre: ');
    for i := 1 to CANT_LETRAS do
        Write (a.nombre[i]);
    writeLn;
    (* Imprimo idioma *)
    write ('Idioma: ');
    case a.idioma of
        pt: writeLn('Portugues');
        en: writeLn('Ingles');
        es: writeLn('Espa~nol');
    end;
    (* Imprimo visitas *)
    writeLn ('Visitas: ', a.visitas);
    (* Imprimo ultima_act *)
    writeLn ('Ultima actualizacion: ', a.ultima_act.dia:2, '/',
    a.ultima_act.mes:2,'/', a.ultima_act.anio:4);
end;
```

Una solución para el procedimiento `imprimirArticulosMasRecientes` puede ser utilizando el procedimiento de la parte b:

```
procedure imprimirArticulosMasRecientes (wiki: Wikipedia);
const CANT_IDIOMAS = 3;
var art : TArticulo;
    id : TIdioma;
begin
  for id := TIdioma(0) to TIdioma(CANT_IDIOMAS - 1) do
    begin
        articuloMasReciente(wiki, id, art);
        imprimirArticulo(art)
    end;
end;
```

Esta solución resuelve el problema pero al llamar tres veces a articuloMasReciente, se recorre tres veces toda la wikipedia, una vez para cada idioma, sin embargo este problema se puede resolver de una forma más eficiente.

A continuación se presenta una segunda solución:

```
procedure imprimirArticulosMasRecientes (wiki: Wikipedia);
const CANT_IDIOMAS = 3;
var arts : Array [TIdioma] of TArticulo;
    encontre : Array [TIdioma] of boolean;
    i, j, k : integer;
    id : TIdioma;
begin
    j := 0;
    i := 1;

    for id := TIdioma(0) to TIdioma(CANT_IDIOMAS - 1) do
        encontre[id] := false;

    {*Encontrar el primero para cada idioma*}
```

```
    while (j <> CANT_IDIOMAS) do
    begin
        if not encontre[wiki[i].idioma] then
        begin
            j := j + 1;
            encontre[wiki[i].idioma] := true;
            arts[wiki[i].idioma] := wiki[i];
        end;
        i:= i + 1;
    end;

    {*Recorrer la wikipedia comparando}
    for k := 1 to CANT_ARTICULOS do
        if esPosterior(wiki[k].ultima_act, arts[wiki[k].idioma].ultima_act) then
            arts[wiki[k].idioma] := wiki[k];

    for id := TIdioma(0) to TIdioma(CANT_IDIOMAS - 1) do
    begin
        articuloMasReciente(wiki, id, art);
        imprimirArticulo(art)
    end;
end;
```

Esta solución recorre dos veces la wikipedia, una primera vez para buscar el primer artículo de cada idioma, almacenándolos en el arreglo *arts* y luego una segunda vez para buscar si hay artículos más recientes para cada idioma. Esta solución es más eficiente ya que se recorre únicamente dos veces la wikipedia.

Una solución aún más eficiente recorre una única vez la wikipedia, realizando una búsqueda de los primeros artículos de cada idioma, pero, al mismo tiempo, si se encuentra un artículo de un idioma ya procesado, se comparan las fechas para quedarse con el más reciente. Una vez que se tiene un artículo para cada idioma, se retoma la recorrida desde el punto anterior hasta el final de la wikipedia, recorriéndola una única vez en todo el procedimiento.

Esta solución se presenta a continuación:

```
procedure imprimirArticulosMasRecientes (wiki: Wikipedia);
const PRIMER_IDIOMA = es;
        ULTIMO_IDIOMA = pt;
        CANT_IDIOMAS = 3;
var arts : Array [TIdioma] of TArticulo;
    encontre : set of TIdioma;
    i, j : integer;
    id : TIdioma;
begin
    i := 1;
    j := 0;
    encontre := [];
    repeat
    if wiki[i].idioma in encontre then
    begin
        if esPosterior(wiki[i].ultima_act, arts[wiki[i].idioma].ultima_act) then
        arts[wiki[i].idioma] := wiki[i]
    end else
    begin
        encontre := encontre + [wiki[i].idioma];
        arts[wiki[i].idioma] := wiki[i];
        j := j + 1;
    end;
```

```
    i := i + 1;
    until j = CANT_IDIOMAS;
    for j := i to CANT_ARTICULOS do
    if esPosterior(wiki[j].ultima_act, arts[wiki[j].idioma].ultima_act) then
        arts[wiki[j].idioma] := wiki[j];
    for id := TIdioma(0) to TIdioma(CANT_IDIOMAS - 1) do
    imprimirArticulo(arts[id])
end;
```

13. Se considera una versión simplificada para la gestión de la información de los salones que integran una facultad. Se asume que la facultad posee `CANT_SALONES` salones, y que cada salón tiene un máximo de `MAX_PIZARRONES` pizarrones.

    Para representar esta realidad se definen las siguientes declaraciones:

```
const
   CANT_SALONES = ...; { valor entero mayor a 0 }
   MAX_PIZARRONES = ...; { valor entero mayor a 0 }

type
   TSalon = record
               asientos : Integer;
               pizarrones : 1..MAX_PIZARRONES;
               hayProyector : Boolean;
            end;

    TFacultad = array [1..CANT_SALONES] of TSalon;
```

    (a) Implemente el procedimiento `informeSalones` tal que imprima un informe de todos los salones de la facultad, incluyendo, por cada uno de ellos, cantidad de asientos, pizarrones, y si tiene proyector o no.

    ```
    procedure informeSalones (facu: TFacultad);
    ```

```
    procedure informeSalones (facu: TFacultad);
    var i :  1 .. CANT_SALONES;
    begin
       for i := 1 to CANT_SALONES do
       begin
          write('El salón ',i:1,' tiene ',facu[i].asientos:1,' asientos,',
            facu[i].pizarrones:1,' pizarrones y ');
          if facu[i].hayProyector then
             writeLn('tiene proyector.')
          else
             writeLn('no tiene proyector.')
       end
    end;
```

    (b) Implemente el procedimiento `salonMasAsientos` tal que devuelve el índice de la celda correspondiente al salón con la mayor cantidad de asientos de la facultad junto con la cantidad de asientos correspondiente. En caso de haber dos o más salones con la mayor cantidad de asientos, devuelve el primero de ellos.

    ```
    procedure salonMasAsientos (facu: TFacultad; var indSalon: Integer;
                                                        var maxAsientos: Integer);
    ```

```
    procedure salonMasAsientos (facu: TFacultad; var indSalon: Integer;
                                                          var maxAsientos: Integer);
    var i :  1 .. CANT_SALONES;
```

```
begin
    indSalon := 1;
    for i := 2 to CANT_SALONES do
        if facu[i].asientos > facu[indSalon].asientos then
            indSalon := i;
    maxAsientos := facu[indSalon].asientos
end;
```

(c) Implemente la función `primerSalonSinAsientos` tal que devuelve el índice de la celda correspondiente al primer salón de la facultad que no tiene ningún asiento. En caso de que no haya ningún salón sin asientos, devuelve cero. Su encabezado es el siguiente:

`function primerSalonSinAsientos (facu: TFacultad) : Integer;`

```
function primerSalonSinAsientos (facu: TFacultad) : Integer;
var i :  integer;
begin
    i := 1;
    while (i <= CANT_SALONES) and (facu[i].asientos <> 0) do
        i := i + 1;
    if i <= CANT_SALONES then
        primerSalonSinAsientos := i
    else
        primerSalonSinAsientos := 0
end;
```