Ordinales, Enumerados y Conjuntos

Programación 1

InCo - FING

Section 1

Ordinales

¿Cuáles son los tipos ordinales?

- Los tipos ordinales (o escalares) son:
  - integer
  - char
  - boolean
  - subrangos
  - enumerados
- Para todos ellos están definidas las funciones `pred`, `succ` y `ord`.
- Notar que el tipo `real` **NO** es ordinal.
- En el libro Konvalina se los llama **escalares**.

Uso de tipos ordinales

Las siguientes construcciones de Pascal tienen que ver con tipos ordinales:

- ***for cont := .....***
  El tipo de la variable `cont` debe ser un ordinal.
- ***case expresion of ....***
  El tipo de `expresión` debe ser un ordinal.
- ***array [tipo_indice] of ...***
  El `tipo_indice` de un arreglo debe ser ordinal.
- ***set of tipo_base***
  El `tipo_base` de un **set** debe ser un ordinal.

Section 2

Enumerados

Motivación

- Un dato toma un valor dentro de un conjunto de valores posibles.
- Esos valores se pueden enumerar y opcionalmente ordenar.
- Podrían representarse como enteros, pero no nos interesa operar con ellos.
- Podrían representarse como caracteres, pero es poco mnemotécnico.

Ejemplos

- Días de la semana.
- Meses del año.
- Colores.
- Puntos cardinales.

```
type
    DiaSemana = (domingo,lunes,martes,
                 miercoles,jueves,viernes,
                 sabado);

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

    PuntoCardinal = (norte,sur,este,oeste);
```

En general

La declaración de un tipo enumerado tiene esta forma:

- **type** *identificador* = `(I1,...,In)`;

donde `I1,...,In` son *identificadores* cualesquiera.

Valores de un enumerado

Los valores de un tipo enumerado siempre son **identificadores**.

```
type
   (* incorrecto! *)
   DigitoPar = (0,2,4,6,8);

   (* incorrecto! *)
   Vocal = ('a','e','i','o', 'u');
```

Notar que el tipo `boolean` es un enumerado predefinido.

Operaciones con enumerados

De acuerdo al orden de enumeración quedan definidas las siguientes operaciones:

- Comparaciones: $=$, $<>$, $<$,$>$, $<=$, $>=$
- Contiguos: `succ`, `pred`

Conversión entre enumerados y enteros

Es fácil convertir de entero a enumerado y viceversa:

- **Enumerado a entero**: la función `ord` asigna un entero de acuerdo al orden de enumeración:
  - `ord(enero) --> 0`
  - `ord(sur)  --> 1`
- **Entero a enumerado** : se usa el *nombre* del tipo como función:
  - `Mes(0) -->  enero`
  - `PuntoCardinal(1) --> sur`

Entrada

No es posible hacer `read` con enumerados:

```
procedure LeeMes(var m: Mes);
var
   mes_aux:  0..12;
begin
   (* leer mes como entero *)
   Write('Ingrese mes (1-12): ');
   ReadLn(mes_aux);

   (* codificar *)
   m:= Mes(mes_aux-1);
end;
```

Salida

No es posible hacer `write` de un enumerado:

```
procedure MostrarMes(m: Mes);
begin
    case m of
      enero     : write('enero');
      febrero   : write('febrero');
      marzo     : write('marzo');
      abril     : write('abril');
      mayo      : write('mayo');
      junio     : write('junio');
      julio     : write('julio');
      agosto    : write('agosto');
      setiembre: write('setiembre');
      octubre   : write('octubre');
      noviembre: write('noviembre');
      diciembre: write('diciembre');
    end;
end;
```

Section 3

Conjuntos

Motivación

- Representación de conjuntos de elementos simples.
- Desde el punto de vista matemático representa el llamado *conjunto potencia* de un conjunto.
- Operaciones del álgebra de conjuntos: pertenencia, unión, intersección, diferencia.
- A diferencia de un arreglo no hay orden ni elementos repetidos.

Definición

Sintaxis de la definición:

```
type T  = set of tipo_elemento;
```

donde:

- *T* es un identificador, nombre de tipo conjunto definido
- `tipo_elemento` es un tipo *ordinal*. Se le denomina el *tipo base* del conjunto.
- Cada valor del tipo `T` es un conjunto de valores del tipo *tipo_elemento*.
- Free Pascal restricciones:
  - el cardinal del tipo base no puede superar 256
  - no se admiten elementos con ordinal negativo (ej: `set of -10..10`)

Ejemplo

```
type
    alfaset = set of 'A'..'Z';
    codigos = set of 1..50;
    charset = set of char;
    intset = set of integer' (* demasiado grande! *)
```

Representación de conjuntos

```
type digitos = set of 0..9;
```

Los siguientes son objetos de este tipo:

```
[0,2,4,6,8]
[]
[1,3]
[7]
[1..3,6..9]    (* free pascal *)
```

Operadores de conjuntos

Operaciones habituales del álgebra de conjuntos:

Símbolo	Operación
`+`	unión
`*`	intersección
`-`	diferencia
`in`	pertenencia
`=`	igualdad
`<>`	desigualdad
`<=`	inclusión

Ejemplo

Leer una línea de códigos:

```
type
    codigos = set of 1..50;
var
    setcod : codigos;
    codigo : 0..50;
    ...
begin
    ...
    setcod := [];
    while not eoln do
    begin
        read(codigo);
        setcod := setcod + [codigo]
    end;
    readln;
    ...
end
```

Ejemplo

Leer un texto y mostrar las letras que aparecen repetidas:

```
type
    CharSet = set of char;
var
    conj1,conj_rep: CharSet;
    car: char;
begin
    (* inicializacion *)
    conj1:= [];
    conj_rep:= [];
    read(car);
    while car <> CENTINELA do
    begin
        if car in conj1 then
            conj_rep:= conj_rep + [car]
        else
            conj1:= conj1 + [car];
        read(car);
    end;
    ...   {mostrar conj_rep}
```

Como mostrar un conjunto

Es necesario recorrer todo el universo:

```
for car:= 'A' to 'Z' do
   if car in conj then
      WriteLn(car);
```