

Curso de Programación 1

Plan 97

Clase 8

(Tipos Escalares, Subrango, Array)

Tipos Escalares

Se dice que un tipo es escalar (u ordinal) si existe un orden entre sus valores individuales.

Ejemplos de tipos escalares son

- **Integer**
- **Char**
- **Boolean**
- Tipos enumerados

Tipos Escalares

Junto con el ordenamiento de los valores, los tipos escalares proveen las siguientes funciones estándar (notar la sobrecarga):

- **`succ: T -> T` (función sucesor)**
Dado v de tipo T , `succ(v)` retorna el valor inmediato superior (si lo hay).
- **`pred: T -> T` (función predecesor)**
Dado v de tipo T , `pred(v)` retorna el valor inmediato inferior (si lo hay).

Tipos Escalares

Ejemplos:

<code>succ(2)</code>	retorna	<code>3</code>
<code>succ('A')</code>	retorna	<code>'B'</code>
<code>succ(ene)</code>	retorna	<code>feb</code>
<code>pred('9')</code>	retorna	<code>'8'</code>
<code>pred(E)</code>	retorna	<code>A</code>
<code>pred(jueves)</code>	retorna	<code>miercoles</code>

Tipos Escalares

Otras funciones estándar:

- **ord** : **T** -> **Integer**

Dado v de tipo **T**, **ord**(v) retorna la posición de v en el ordenamiento de las constantes del tipo de dato **T**. (Las posiciones empiezan a contarse desde cero.)

- **chr** : **Integer** -> **Char**

Dado n de tipo **Integer**, **chr**(n) retorna el carácter cuya posición (en la tabla ASCII) es representada por el entero n .

Tipos Escalares

Ejemplo:

```
Type Vocal = (A,E,I,O,U);
```

```
ord(A)      retorna    0
```

```
ord(U)      retorna    4
```

```
ord('A')    retorna   65
```

```
chr(49)     retorna   '1'
```

```
chr(65)     retorna   'A'
```

Tipo Subrango

A menudo ocurre que una variable toma valores de un tipo escalar comprendidos en un cierto intervalo. Para reflejar esta situación, Pascal provee la posibilidad de declarar un tipo subrango.

Type T = *min*..*max*

siendo *min* y *max* los extremos del intervalo.

Tipo Subrango

Ejemplos:

Type

Escala = 1..10;

Mayuscula = 'A'..'Z';

Laborable = lunes..viernes;

Positivo = 1..maxint;

Tipo Subrango

Los valores que el subrango contiene son todos aquellos valores del *tipo escalar base* entre los extremos inferior (*min*) y superior (*max*) especificados.

Ejemplo:

```
Type Escala = 1..10;
```

declara los valores: **1,2,3,4,5,6,7,8,9 y 10.**

Tipo Subrango

La verificación de que los valores de tipo subrango caen en el intervalo especificado es realizada en tiempo de ejecución.

Si el valor asignado a una variable de tipo subrango cae fuera del intervalo especificado, un mensaje de error aclaratorio es exhibido.

Sentencia case

La sentencia **case** permite seleccionar entre varias alternativas.

Su formato es el siguiente:

```
case <selector> of
  etiqueta1 : <sentencia1> ;
  . . . . .
  etiquetan : <sentencian>
end
```

Sentencia case

- Las **etiquetas** de un case deben ser constantes, distintas entre sí y todas de un mismo tipo de dato escalar.
- El **selector** es una expresión de ese mismo tipo. El valor del selector determina la alternativa a escoger, y por lo tanto, la sentencia a ejecutar.
- Luego de ejecutar la sentencia en la alternativa seleccionada, el control continúa con la sentencia que sigue al case.

Sentencia case

Ejemplo:

```
Var v:(A,E,I,O,U);

case vocal of
  A : writeln('La vocal es A');
  E : writeln('La vocal es E');
  I : writeln('La vocal es I');
  O : writeln('La vocal es O');
  U : writeln('La vocal es U');
end;
```

Sentencia case

Si varias alternativas hacen referencia a una misma sentencia, entonces es posible agruparlas en una sola con las etiquetas separadas por comas.

Ejemplo:

```
Var m:(ene,feb,...,dic);

case m of
  ene,mar,may,
  jul,ago,oct,dic: writeln('31 dias');
  abr,jun,set,nov: writeln('30 dias');
  feb              : writeln('28 dias')
end;
```

Tipos de datos estructurados

Enteros, reales, caracteres, booleanos, enumerados y subrangos conforman los llamados **tipos de datos simples**.

A partir de estos tipos elementales otros más complejos pueden ser contruidos aplicando diversas formas de estructuración de datos. Los tipos así obtenidos son llamados **tipos estructurados**.

En este curso vamos a estudiar dos formas básicas de estructurar datos: **arrays** y **registros**.

La estructura Array

El **array** es una estructura de datos común a muchos lenguajes de programación.

Un **array** es una estructura **homogénea**, en el sentido que está compuesta por elementos todos de un mismo tipo.

Es también una estructura de **acceso aleatorio**; todos sus elementos son igualmente accesibles.

Declaración de arrays

Un array se declara de esta forma:

```
Type T = array [I] of T0
```

T_0 es llamado el **tipo base**. De este tipo son todos los elementos del array.

I es el **tipo del índice** (debe ser un tipo escalar o subrango). El valor del **índice** es lo que identifica a cada elemento del array.

Ejemplo

```
Type fila = array [1..5] of integer;
```

Si f : **fila** y suponemos que cada lugar del array satisface

$$f_i = 2 * i$$

entonces en forma gráfica podemos ver lo siguiente:

f_1	f_2	f_3	f_4	f_5
2	4	6	8	10

Otros ejemplos

Type

```
nombre      = array [1..20] of char;  
cuentacar  = array [char] of integer;  
tablames   = array [Mes] of 28..31;
```

donde

Type

```
Mes = (ene, feb, ..., dic);
```

Acceso a elementos de un array

Conceptualmente, un array

```
A : array [I] of T0
```

puede ser entendido como una función con dominio I y codominio T_0 .

```
A : I -> T0
```

Usando esta interpretación, vamos a denotar por $A[x]$ la forma de *seleccionar* el valor de un elemento del array, por analogía con la aplicación de funciones $A(x)$.

Acceso a elementos de un array

Para imprimir el valor de cada uno de los elementos individuales del array `f : fila` visto antes hacemos:

```
for i := 1 to 5 do
  writeln(f[i]);
```

y para sumar los elementos del mismo:

```
Var suma : integer;
suma := 0;
For i := 1 to 5 do
  suma := suma + f[i];
```

Actualización de arrays

La *modificación* de un array se hace a través de la *actualización* de sus elementos.

Esto se realiza mediante una asignación en la que el elemento a modificar aparece a la izquierda de la misma:

```
A[x] := e;
```

Ejemplo: Para inicializar el array `f : fila` hacemos

```
for i := 1 to 5 do
  f[i] := 2 * i;
```

Ejemplo

Sea

```
tmes : tablames
```

una tabla con el número de días de cada mes.

Para inicializar esta tabla hacemos:

```
for m := ene to dic do
  case m of
    ene,mar,may,
    jul,ago,oct,dic: tmes[m] := 31;
    abr,jun,set,nov: tmes[m] := 30;
    feb              : tmes[m] := 28;
end;
```

Programación 1. Plan 97. InCo. Fac. de Ingeniería

23

Copia de arrays

Dependiendo del compilador Pascal, la **copia** del contenido de un array a otro del mismo tipo se puede realizar de dos maneras distintas.

Sean,

```
Var A,B : array [1..10] of char;
```

El contenido de **A** se puede copiar a **B** haciendo:

```
for i := 1 to 10 do
  B[i] := A[i];
```

o mediante la asignación:

```
B := A;
```

Programación 1. Plan 97. InCo. Fac. de Ingeniería

24

Arrays multidimensionales

Los elementos de un array pueden ser a su vez estructurados.

O sea, estructuras como arrays o registros (que veremos mas adelante) pueden ser elementos de arrays.

Por ejemplo, un array cuyos elementos son también arrays se denomina una **matriz**.

Ejemplo

Por ejemplo, dado el tipo

```
Type fila = array [1..5] of integer;
```

visto antes, podemos declarar una matriz:

```
Var M : array [1..10] of fila;
```

compuesta por 10 filas, cada una de 5 elementos.

La forma de seleccionar el elemento *j* de la fila *i* es usando dos selectores:

```
M[i][j]      que se escribe      M[i,j]
```