

## Tipos enumerados

1. Considere la siguiente declaración:

```
TYPE color = (rojo, blanco, azul, purpura);
VAR
```

```
    coloracion : color;
```

y los siguientes fragmentos de código :

```
Read (rojo);
WriteLn (rojo);
```

```
Read (coloracion);
Write (coloracion);
```

```
coloracion := blanco;
WriteLn (coloracion);
```

```
coloracion := blanco;
CASE coloracion OF

    rojo    : WriteLn ('rojo');
    blanco  : WriteLn ('blanco');
    azul    : WriteLn ('azul');
    purpura : WriteLn ('purpura')
END;
```

```
coloracion := azul;
CASE coloracion OF
    rojo    : WriteLn (rojo);

    blanco  : WriteLn (blanco);
    azul    : WriteLn (azul);
    purpura : WriteLn (purpura)
END;
```

```
IF coloracion = azul THEN
    WriteLn ('azul')
ELSE
    WriteLn ('no azul')
```

Determine cuáles son válidos.

2. Determine cuales de las siguientes declaraciones son válidas:

```
TYPE letra = ('X', 'Y', 'Z');
```

```
TYPE lenguaje = (Pascal, Fortran, Basic);
```

```
TYPE materias = (matematicas, historia, computacion, biologia);
    carrera = (matematicas, computacion);
```

```
TYPE estado = (residente, ciudadano, extranjero);
    nacionalidad = (americano, europeo, africano, asiatico, otra);
```

```
TYPE codigo = (1, 2, 3, 4, 5);
```

```

TYPE codigo = (c1, c2, c3, c4, c5);

TYPE estado = (soltera, casada, comprometida, divorciada);
VAR type : estado;

TYPE ciudad = (Paysandu, SanJose, Tacuarembu, Canelones);
VAR ciudad : type;

VAR ciudad : (Rivera, Salto, Soriano, Rocha);

PROCEDURE encontrar (VAR ciudad : (Minas, Florida, Flores));

TYPE trabajo = (obrero, oficinista, indefinido);
...
PROCEDURE buscar (VAR empleo : trabajo);

```

3. Estudie el siguiente programa:

```

PROGRAM bueno (input, output);
TYPE materia = (matematicas, historia, computacion, geografia, fisica);
VAR a, b : materia;
BEGIN
  a := matematicas;
  b := computacion;
  IF a > b THEN
    Write ('Magnifico')
  ELSE
    Write ('Excelente')
END.

```

Indique cuáles de las siguientes afirmaciones son ciertas:

- i. En la proposición IF, *a* y *b* no se pueden comparar porque no son números
  - ii. El programa exhibir á *Magnifico*
  - iii. El programa exhibir á *Excelente*
  - iv. La proposición IF provocará un error de ejecución
4. Definir un tipo enumerado que represente los meses del año. Escribir un procedimiento en PASCAL que reciba un parámetro de entrada con un valor del tipo definido anteriormente y exhiba el nombre completo del mes correspondiente

## Tipos subrango

5. Definir un tipo de subescala para cada una de las siguientes definiciones :

Los enteros no negativos.	Los caracteres de la primera mitad del alfabeto.
Los enteros mayores de 100.	Los caracteres de la segunda mitad del alfabeto.

6. Examine estas declaraciones:

```

TYPE tipodia = (Lunes, Martes, Miercoles, Jueves, Viernes, Sabado, Domingo);
VAR dia : tipodia;
  laborable : Lunes..Viernes;
  finsemana : Sabado..Domingo;

```

Indique cuáles de las siguientes afirmaciones son ciertas:

- i. Se producirá un error de ejecución si se asigna el valor *Martes* a *findesemana*

- ii. Se producirá un error de ejecución si se asigna el valor *Viernes a laborable*
7. Suponga que se hacen las siguientes declaraciones en un programa correcto en PASCAL:

```
VAR bajo : 1..5;
    medio : 1..10;
    alto : 6..20;

    grande : '1'..'10';
```

Si el valor de `alto` es 7, ¿cuál de estas asignaciones generará un error de compilación o de ejecución?

- i. `bajo := alto;`  
 ii. `alto := 3 * alto;`  
 iii. `grande := alto;`  
 iv. `medio := alto;`  
 v. `medio := alto - 7;`
8. Indique cuáles de las siguientes declaraciones de subescala son válidas:

- i. `TYPE etiq1 = '0'..9;`  
 ii. `TYPE etiq2 := -3..6;`  
 iii. `TYPE etiq3 = -1..-3;`  
 iv. `TYPE etiq4 = 0.0..9;`  
 v. `TYPE etiq5 = 1..1;`

9. Examine la siguiente definición :

```
TYPE raro = (gugol, nudol, brudol, cudol, zudol, budol);
```

Determine el valor de las siguientes expresiones:

<code>ord (gugol)</code>	<code>ord (zudol)</code>
<code>succ (brudol)</code>	<code>succ (budol)</code>
<code>pred (gugol)</code>	<code>ord (succ (zudol))</code>

10. Obtenga el valor de las siguientes expresiones:

<code>ord ('7') - ord ('0')</code>	<code>ord ('1') - ord ('1')</code>
<code>chr (3 + ord ('0'))</code>	<code>chr (0 + ord ('0'))</code>

11. Examine la siguiente declaración:

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

Determinar si se ejecutará sin error los siguientes códigos en PASCAL:

```
letra := a;
WHILE letra <= u DO
BEGIN
  Read (uncar);
  WriteLn ('El caracter capturado es ', uncar);
  letra := succ (letra)
END
```

```
letra := u;
REPEAT
  Read (uncar);
  WriteLn ('El caracter capturado es ', uncar);

  letra := pred (letra)
UNTIL letra = a
```

## Tipo set

12. Enumere los miembros de cada conjunto:

[0, 1, 2, 3]	[0..10, 13, 15]	['A'..'J', 'L'..'Z']
['a'..'m', 'A'..'M']	['0'..'5', '9']	

13. Determine el valor de las siguientes expresiones booleanas:

'A' IN ['B'..'S']	0 IN [-3..5]
7 IN [1..4, 6..8]	' ' IN []

## Arreglos

14. ¿Cuáles de las siguientes asignaciones son válidas, dada la declaración que se muestra ?

```
TYPE
  lista = ARRAY ['A'..'Z'] OF Integer;
  datos = ARRAY [-3..3] OF Char;
```

```
VAR
  lista1, lista2 : lista;
  datos1 : datos;
```

- i. lista1['A'] := lista2['B']
- ii. lista1 := lista2
- iii. lista1['Z'] := lista2['A'] + 1
- iv. datos1 := lista
- v. lista1['A'] := datos1[0]
- vi. datos1[lista1['A']] := 'A'
- vii. lista1[datos1[0]] := 0

15. Dadas las declaraciones:

```
CONST
  N = 500;
VAR
  puntos: ARRAY [1..N] OF Integer;
  prueba, menor, indice: Integer;
```

¿Cuál segmento de programa encontrará el valor más pequeño de este arreglo, y almacenará el subíndice del elemento donde está guardado este valor?

1. FOR prueba := 1 TO N DO
 

```
        IF (puntos[prueba] < menor) THEN
            menor := puntos [menor];
```
2. menor := puntos[1];
 FOR prueba := 2 TO N DO
 

```
        IF (puntos[prueba] < menor) THEN
            menor := puntos [prueba];
```
3. indice := 1;
 FOR prueba := 2 TO N DO
 

```
        IF (puntos[prueba] < puntos[indice]) THEN
            indice := prueba;
```
4. FOR prueba := 1 TO N DO
 

```
        IF (puntos[prueba] < menor) THEN
            menor := prueba;
```
5. indice := 1;
 FOR prueba := 2 TO N DO
 

```
        IF (puntos[prueba] < indice) THEN
            indice := prueba;
```

16. ¿Cuáles de los siguientes segmentos de programa invertirán un arreglo de caracteres llamado *cadena* que consta de *N* elementos?

```

1. FOR conmuta := 1 TO N DO BEGIN
    temp := cadena[conmuta];
    cadena[(N+1) - conmuta] := temp
  END

2. FOR conmuta := 1 TO N DO BEGIN
    temp := cadena[conmuta];
    cadena[conmuta] := cadena[(N+1) - conmuta];
    cadena[(N+1) - conmuta] := temp
  END

3. FOR conmuta := 1 TO (N DIV 2) DO BEGIN
    temp := cadena[conmuta];
    cadena[conmuta] := cadena[(N+1) - conmuta];
    cadena[(N+1) - conmuta] := temp
  END

4. FOR conmuta := 1 TO (N DIV 2) DO BEGIN
    temp := cadena[conmuta];

    cadena[conmuta] := cadena[(N+1) - conmuta];
    cadena[conmuta] := temp
  END

5. FOR conmuta := 1 TO (N DIV 2) DO Begin
    temp := cadena[conmuta];
    cadena[conmuta] := cadena[(N DIV 2) - conmuta];
    cadena [(N DIV 2) - conmuta] := temp
  END

```

17. Dada la siguientes declaraciones:

```

CONST N = ...;
...
TYPE
  RangoN = 1..N;
  Tabla = Array[RangoN] OF Integer;

```

- a. Escriba una función *HallaMax* que encuentre el valor más grande de un arreglo.

```

Function HallaMax (t : Tabla) : Integer;
(* Hallar el elemento más grande en tabla[1..N] *)

```

- b. Escriba una función *HallaIndMax* que encuentre el índice del valor más grande de un arreglo.

```

Function HallaIndMax (t : Tabla) : RangoN;
(* Hallar el índice del elemento más grande en tabla[1..N] *)

```

18. Examine la siguiente declaración :

```

TYPE
  Tabla = Array ['0'..'9', 1..9] OF Boolean;
VAR
  matriz : Tabla;

```

- a. ¿ Cuántos componentes contiene esta tabla ?  
 b. ¿ Cuáles de las siguientes proposiciones de asignación son válidas ?  
 i. matriz [0,1] := true;  
 ii. matriz ['9',9] := 0;  
 iii. matriz ['0',1] := matriz ['9',0] AND matriz ['7',7];  
 iv. matriz ['1',1] := '1' < 1;

19. Escriba un procedimiento en Pascal llamado *Cambio* que tenga los parámetros *matriz* (arreglo bidimensional de enteros de diez renglones y diez columnas) y dos variables enteras *m* y *n*. El

procedimiento *Cambio* intercambia los renglones  $m$  y  $n$  de *matriz*. Incluya las definiciones de tipo necesarias.

20. Escriba un procedimiento *flipflop* en Pascal que dada la matriz de booleanos de tipo:

```
CONST
    M = ...;
    N = ...;

    P = ...;

TYPE
    RangoM = 1..M;
    RangoN = 1..N;
    RangoP = 1..P;
    MatrizBool = ARRAY[RangoM,RangoN,RangoP] OF Boolean;
```

cambie todos los *true* a *false* y viceversa, en cada uno de los componentes de la matriz *lógica*.

21. La transpuesta de un arreglo bidimensional cuadrado  $a$  es un arreglo  $b$  del mismo tipo cuyos componentes satisfacen la relación  $b[i,j] = a[j,i]$  para todos los valores posibles de  $i$  y  $j$ . Escriba un procedimiento que calcule la transpuesta de un arreglo de números reales con cinco renglones y cinco columnas.

Obs: Intente hacerlo sobre la misma matriz.

22. Una matriz simétrica  $a$  es un arreglo cuadrado bidimensional con la propiedad de que  $a[i,j] = a[j,i]$ . Una matriz así se puede almacenar en forma compacta (en el arreglo  $ac$ ) ya que basta con almacenar una sola vez los elementos duplicados del arreglo.

- En primer término, diseñe un método para almacenar un arreglo de éstos sin repetir los elementos duplicados.
- Después escriba una función *ObtSim*( $ac, i, j$ ) que produzca el valor de  $a[i,j]$
- Finalmente, un procedimiento *AlmaSim*( $ac, i, j, val$ ) que almacena  $val$  en la localidad apropiada del arreglo compacto  $ac$   
Sugerencia : basta con almacenar  $n$  elementos de la columna  $n$ ; piense en un arreglo unidimensional que contiene los elementos de la columna uno, después de la columna dos y así sucesivamente.

23. Dada la siguientes declaraciones:

```
CONST
    M = ...;
    N = ...;

TYPE
    RangoM = 1..M;
    RangoN = 1..N;

    MatrizMxN = ARRAY[RangoM,RangoN] OF Integer;

    MatrizMxM = ARRAY[RangoM,RangoM] OF Integer;
```

Escribir un procedimiento *ProdMatriz* que implemente el producto de una matriz  $a$  con  $b$ :

```
PROCEDURE ProdMatriz(a : MatrizNxM; b : MatrizMxM; VAR c : MatrizNxM);
```