

# Tema 10

## Ampliación De Estructuras De Control

### Estructuras complementarias de iteración

- **Sentencia REPEAT:** Se encarga de realizar un bucle comprobando la condición al final de cada iteración, no al principio como el WHILE.

```
REPEAT
    acciones;
UNTIL condición;
```

- **Sentencias LOOP y EXIT:** Utilizan dos sentencias para crear una estructura iterativa. Se crea lo que se denomina un *bucle indefinido*. Puede haber varios EXIT dentro del bucle.

```
LOOP
    acciones;
    IF condición THEN EXIT END;
    acciones;
END;
```

### Estructuras complementarias de selección

- **Sentencia CASE:** Se utiliza cuando la selección entre varios casos alternativos depende del valor que toma una determinada expresión, siendo el resultado de está de un tipo ordinal: INTEGER, CARDINAL, CHAR, enumerado o subrango.

```
CASE expresión OF
    valores1: acción1 |
    valoresN: acciónN |
    ...
ELSE
    acción por defecto
END;
```

### Equivalencia entre estructuras

Las estructuras básicas estrictamente necesarias para la programación estructurada son la selección (IF-THEN-ELSE-END) y la iteración (WHILE-DO-END) y se consideran estructuras primarias. El resto, que consideraremos estructuras secundarias, son en general más complejas y tienen como objetivo lograr programas más sencillos en situaciones particulares.

- **Selección general:** Se consigue mediante selecciones primarias anidadas.

```
IF condición1 THEN
    acción1;
ELSIF condiciónN THEN
    acciónN;
...
ELSE
    acción por defecto
END;
```

Se convierte en:

```
IF condición1 THEN
    acción1;
ELSE
    IF condiciónN THEN
        acciónN;
    ELSE
        ...
        acción por defecto
    END;
END;
```

- **Selección por casos:** Se consigue mediante selecciones primarias anidadas.

```
CASE expresión OF
    valores1: acción1 |
    valoresN: acciónN |
    ...
ELSE
    acción por defecto
END;
```

Se convierte en:

```
valor := expresión;
IF valor = valores1 THEN
    acción1;
ELSIF valor = valoresN THEN
    acciónN;
...
ELSE
    acción por defecto
END;
```

- **Bucle con contador:** Se puede hacer mediante un control explícito del contador del bucle.

```
FOR var:=ini TO fin BY incre DO
    acciones;
END;
```

Se convierte en:

```
var := ini;
WHILE var <= fin DO
    acciones;
    INC(var, incre);
END;
```

- **Repetición:** La sentencia `REPEAT` se puede transformar en `WHILE` complementando la condición de repetición y ordenando la ejecución incondicional de la primera iteración.

```
REPEAT  
  acciones;  
UNTIL condición;
```

Se convierte en:

```
acciones;  
WHILE NOT condición DO  
  acciones;  
END;
```

- **Bucle indefinido:** La transformación de esta estructura puede resultar bastante compleja dependiendo del nivel de anidamiento en el que se produzcan las salidas y el número de salidas existentes.

```
LOOP  
  accionesA;  
  IF condición THEN EXIT END;  
  accionesB;  
END;
```

Se convierte en:

```
accionesA;  
WHILE NOT condición DO  
  accionesB;  
  accionesA;  
END;
```