

# Tema 3

## Constantes Y Variables

### Identificadores

La manera de hacer referencia a los diferentes elementos que intervienen en un programa es darles un nombre particular a cada uno. Se denominan *identificadores* a los nombres usados para identificar cada elemento del programa.

### El vocabulario de Modula-2

Hay que distinguir ciertas *palabras reservadas* o *palabras clave* que tiene un significado invariable dentro del lenguaje y que no pueden ser nunca utilizadas como identificadores en un programa.

Palabras Clave	
AND	ARRAY
BEGIN	BY
CASE	CONST
DEFINITION	DIV
DO	ELSE
ELSIF	END
EXIT	EXPORT
FOR	FROM
IF	IMPLEMENTATION
IMPORT	IN
LOOP	MOD
MODULE	NOT
OF	OR
POINTER	PROCEDURE
QUALIFIED	RECORD
REPEAT	RETURN
SET	THEN
TO	TYPE
UNTIL	VAR
WHILE	WITH

El compilador también reconoce una serie de *identificadores predefinidos* para nombrar los tipos de valores básicos, algunas funciones de utilidad, etc.

Identificadores Predefinidos		
ABS	BITSET	BOOLEAN
CAP	CARDINAL	CHAR
CHR	DEC	EXCL
FALSE	FLOAT	HALT
HIGH	INC	INCL
INTEGER	LONGINT	LONGREAL
MAX	MIN	NIL
ODD	ORD	PROC
REAL	SIZE	TRUE
TRUNC	VAL	

### Constantes

- **Concepto de constante:** Una constante es un valor fijo que se utiliza en un programa. El valor debe ser siempre el mismo para cualquier ejecución del programa, es decir, el valor no puede cambiar de una ejecución a otra.
- **Declaración de constantes con nombre:** La declaración de un valor constante con nombre consiste en asociar un identificador a dicho valor constante. La declaración se inicia con la palabra clave `CONST`, y a continuación se escribe el nombre simbólico de la constante, seguido del signo igual y el valor asociado.  

```
CONST Nombre = ExpConstante;
```

Dicho valor puede ser sustituido por una expresión constante, para ello es necesario que todos los operandos que intervengan en la expresión sean valores constantes, y que las operaciones entre ellos sean operadores fijos del lenguaje o funciones predefinidas.

### Variables

- **Concepto de variable:** Una variable representa un valor almacenado, y que se puede conservar indefinidamente para ser usado tantas veces como se desee. Pero el valor de una variable se puede modificar en cualquier momento, y será el nuevo valor el que estará almacenado en ella a partir de entonces.
- **Declaración de variables:** Cada variable debe tener asociado un tipo de valores determinado, y han de ser declaradas en el programa antes de ser utilizadas. La declaración de una variable especifica su nombre y el tipo de valor asociado.  

```
VAR Nombre : Tipo;
```
- **Inicialización de variables:** Para usar una variable de manera correcta es necesario *inicializarla* antes de usar su valor en ningún cálculo. Inicializar una variable es simplemente darle un valor determinado por primera vez.
- **Uso de variables:** El valor almacenado en una variable puede utilizarse usando la variable como operando en una expresión aritmética.

## ***Sentencia de asignación***

Una forma de conseguir que una variable guarde un nuevo valor es mediante una sentencia de asignación. Esta sentencia es característica de la programación imperativa, y permite inicializar una variable o modificar el valor que tenía hasta el momento. Mediante asignaciones podremos dar valores iniciales determinados a las variables, o guardar en ellas los resultados intermedios o finales de cualquier programa.

- **Compatibilidad de tipos:** Para que la sentencia de asignación sea válida es necesario que el tipo de la variable y el del resultado de la expresión sean compatibles. La compatibilidad de tipos en la asignación es un poco menos restrictiva que entre los operandos de una misma operación.

## ***Operaciones de lectura simple***

- **Procedimientos ReadInt, ReadCard y ReadReal:** Los procedimientos ReadInt y ReadCard pertenecen al módulo *InOut*. El procedimiento ReadReal pertenece al módulo *RealInOut*. La sintaxis es la siguiente:

```
ReadInt( VariableEntera )
ReadCard( VariableCardinal )
ReadReal( VariableReal )
```

Estos leen un valor numérico del tipo concreto que corresponda: INTEGER, CARDINAL o REAL, respectivamente y asignan el valor leído a la variable que se indica.

- **Procedimiento Read:** El procedimiento Read pertenece al módulo *InOut*. La sintaxis es la siguiente:

```
Read( VariableCaracter )
```

Esta lee un carácter del tipo CHAR y asigna el valor leído a la variable que se indica.

## ***Estructura de un programa con constantes y variables***

La estructura de un programa es la siguiente:

```
MODULE NombreDePrograma;
  FROM Módulo IMPORT ListaDeNombres;
  [ CONST NombreCons = ExpConstante; ]
  [ VAR  NombreVar  : Tipo; ]
BEGIN
  Sentencias;
  ...
END NombreDePrograma.
```