4. EXPRESIONES Y OPERADORES

Conceptos: *Expresión, Operador, Operando, Asignación, Prioridad*

Resumen: En este tema se presentan los siguientes elementos de la programación: las expresiones y los operadores. Se define el concepto de expresión y se continúa con el estudio de los distintos tipos de operadores: aritmético, de relación, boléanos y de bit. En el apartado final se analizan las reglas de prioridad de los operadores que se siguen en la evaluación de expresiones de todo tipo.

Objetivos específicos. Al finalizar el tema, el alumno deberá ser capaz de:

a) Describir los operadores (asignación, aritméticos, de relación, lógicos y de bit) y los tipos de dato sobre los que actúan (*Conocimiento*)
b) Evaluar expresiones que empleen datos primitivos, operadores y paréntesis (*Comprensión*)
c) Construir expresiones que empleen combinaciones de datos simples, operadores y paréntesis (*Aplicación*)

4.1. INTRODUCCIÓN

Las expresiones son una parte fundamental de la programación ya que sirven para realizar una o varias operaciones sobre un dato o un conjunto de datos, obteniéndose otro dato como resultado. Los operadores definen algunas de las operaciones que pueden realizarse dentro de una expresión.

4.2. EXPRESIONES

Una **expresión** es una combinación de *operadores* y *operandos*. Los datos u operandos pueden ser constantes, variables y *llamadas a funciones*. Además, dentro de una expresión pueden encontrarse *subexpresiones* encerradas entre paréntesis. Por ejemplo, la siguiente expresión matemática:

$$x^2+(b-3)\cdot\cos(\alpha)$$

Cuando se ejecuta una sentencia de código que contiene una expresión, ésta se evalúa. Al evaluarse la expresión toma un valor que depende del valor asignado previamente a las variables, las constantes y los operadores y funciones utilizadas y la secuencia de la ejecución de las operaciones correspondientes. Este valor resultante de la evaluación de la expresión será de un determinado tipo de dato. Por ejemplo, de un tipo numérico entero (`integer`, `shortint`...), de un tipo real o de un tipo lógico o booleano.

Como en el capítulo anterior ya se trató de los datos simples u operandos que pueden emplearse en TurboPascal, este capítulo se centrará en los operadores. El capítulo acerca de las funciones se deja para más adelante.

4.3. OPERADORES

En el código fuente de un programa un **operador** es un carácter o una secuencia de caracteres. Por ejemplo: `+`, `*`, `div` o `shr`. Los operadores definen las operaciones que van a realizarse con los datos u operandos. En TurboPascal existen distintos tipos de operadores. Por un lado, pueden clasificarse, dependiendo del número de operandos, en unarios o unitarios (un operando) y binarios (dos operandos). Por otro lado, pueden clasificarse, dependiendo del tipo de operandos y de su resultado, en operadores aritméticos, de cadenas de caracteres, de relación, lógicos o booleanos, de bit y de conjuntos.

Algunos operadores están *sobrecargados*, lo que significa que la operación que representan depende del número o tipos de operandos sobre los que actúa. De esta forma, por ejemplo el operador + puede hacer referencia a la suma de valores numéricos, a la concatenación de caracteres o a la unión de conjuntos dependiendo del tipo de sus operandos.

4.4. EL OPERADOR DE ASIGNACIÓN

El operador de asignación se representa por la secuencia de caracteres `:=` . Permite asignar a una variable el valor de una expresión. Por ejemplo:

```
var x,y,z: real;
begin
x:=12.5;
y:=-5.7;
z:=2*x+3*y;
```

4.5. OPERADORES ARITMÉTICOS

Los operadores **aritméticos** operan sobre valores de tipo entero o real. Los operadores aritméticos se resumen en la Tabla 12. En el caso del operador unitario de cambio de signo, el resultado es del mismo tipo que el del operando; en el caso de los tres primeros operadores binarios (suma, resta y producto) si ambos operandos son enteros el resultado es entero, si alguno es real el resultado es real.

Con el fin de mantener la coherencia durante la operación, para un operador binario, operandos con distinto tipo se convierten a un mismo tipo común antes de la operación. El tipo común es el tipo de dato predefinido de TurboPascal con el menor intervalo de representación que incluye los valores de ambos operandos. Un concepto equivalente sería el de mínimo común múltiplo.

Por ejemplo, el tipo común de dos tipos `byte` e `integer` es el tipo `integer`. El tipo común de dos tipos `integer` y `word` es un `longint`. El tipo común de dos tipos `integer` y `real` es un `real`. La operación se lleva a cabo dentro del intervalo de representación y con la precisión de este tipo común y el resultado es también de este tipo común.

Tabla 12. Operadores aritméticos

Operador	Descripción	Ejemplo de expresión	Resultado del ejemplo
-	operador unario de cambio de signo	`-4`	`-4`
+	Suma	`2.5+7.1`	`9.6`
-	Resta	`235.6-103.5`	`132.1`
*****	Producto	`1.2*1.1`	`1.32`
/	división real (independientemente de si los operandos son enteros o reales)	`6/8` `0.050/0.2`	`0.75` `0.25`
div	cociente entero (sólo permite operandos enteros)	`20 div 7`	`2`
mod	resto de la división entera (sólo permite operandos enteros)	`20 mod 7`	`6`

Los operadores aritméticos en TurboPascal realizan operaciones aritméticas muy simples. Por ejemplo, ni siquiera existe un operador que permita elevar un valor a una potencia determinada.

Otras operaciones numéricas más complejas (como por ejemplo, logaritmos o funciones trigonométricas) pueden llevarse a cabo con las funciones y procedimientos estándar que incorpora TurboPascal y que se verán más adelante en el capítulo de *Procedimientos y Funciones*.

Las siguientes sentencias incluyen expresiones que contienen algunos operadores aritméticos:

```
var x,y,r: real; n:integer;
begin
x:=-2.3;
y:=x+5;
n:=-6;
r:=4.3*x+(y/2)*n;
writeln('El cuadrado de r es: ',r*r)
end.
```

En una sección posterior de este capítulo (Niveles de prioridad) se explica lo que ocurre cuando se encuentran dos o más operadores y paréntesis en una misma expresión.

4.6. OPERADORES DE RELACIÓN

Los operadores de relación son operadores binarios en los que los operandos son ordinales, reales o de cadena. Los dos primeros operadores sirven también para operandos de tipo `record` y punteros. Todos ellos dan lugar a resultados de tipo booleano. Los operadores de relación se resumen en la Tabla 13.

Tabla 13. Operadores de relación

Operador	Descripción	Ejemplo de expresión	Resultado del ejemplo
=	igual que	`7 = 38`	`false`
<>	distinto que	`'a' <> 'k'`	`true`
<	menor que	`'G' < 'B'`	`false`
>	mayor que	`'beso' > 'alamo'`	`true`
<=	menor o igual que	`7.5 <= 7.38`	`false`
>=	mayor o igual que	`38 >= 7`	`true`

No hay que confundir el operador lógico igualdad **=**, con el operador de asignación **:=**, que asigna valores a variables o funciones. La expresión `a=b` compara los valores almacenados en la variables `a` y `b` y devuelve `true` o `false` según el resultado, mientras que la sentencia `a:=b;` asigna a la variable `a` el valor almacenado en la variable `b`.

4.7. OPERADORES LÓGICOS O BOOLEANOS

Los operadores lógicos o boolenos realizan operaciones con operandos de tipo lógico o booleano y tiene como resultado un dato también del mismo tipo. Los operadores booleanos definidos en TurboPascal se resumen en la Tabla 14[10]:

Tabla 14. Operadores lógicos o booleanos

Operador	Descripción	Ejemplo de expresión	Resultado del ejemplo
not	Negación (unario)	`not false`	`true`
or	Suma lógica (binario)	`true or false`	`true`
and	Producto lógico (binario)	`true and false`	`false`
xor	Suma lógica exclusiva (binario)	`true xor false` `true xor true`	`true` `false`

[10] La tabla resumen completa de los operadores lógicos es la mostrada a continuación:

A	B	not A	A or B	A and B	A xor B
`false`	`false`	`true`	`false`	`false`	`false`
`false`	`true`	`true`	`true`	`false`	`true`
`true`	`false`	`false`	`true`	`false`	`true`
`true`	`true`	`false`	`true`	`true`	`false`

4.8. OPERADORES DE BIT

Los operadores de bit tienen operandos y resultados de tipo entero. Realizan sus operaciones con los ceros y los unos de las representaciones binarias correspondientes a los operandos. Los operadores de bit definidos en TurboPascal se resumen en la Tabla 15. Salvo el operador `Not` los demás son operadores binarios.

Tabla 15. Operadores de bit

Operador	Descripción	Ejemplo de expresión	Resultado del ejemplo
`not`	Cambia el valor de cada bit de la representación binario del dato entero (unario)	`not 2`	`-3`
`and`	Multiplica lógicamente los respectivos bits de las representaciones binarias de cada entero	`12 and 10`	`8`
`or`	Suma lógicamente los bits respectivos.	`12 or 10`	`14`
`xor`	Suma exclusiva de los bits correspondientes	`12 xor 10`	`6`
`shr`	Contracción de la expresión inglesa *shift right*: Desplaza los bits del primer operando hacia la derecha tantas veces como indique el segundo operando (cada vez que lo hace introduce un cero como primer bit a la derecha del resultado)	`32 shr 3`	`4`
`shl`	Contracción de la expresión inglesa *shift left*: Desplaza los bits a la izquierda (introduce un cero a la izquierda del resultado)	`32 shl 2`	`128`

4.9. NIVELES DE PRIORIDAD DE LOS OPERADORES

Una expresión puede contener distintos tipo de operadores mientras cada operador trabaje con operandos del tipo adecuado. La pregunta que surge a continuación es: ¿qué operación de las que se pueda encontrar en una expresión se realiza antes que las demás? Los niveles de prioridad entre operadores en una misma expresión se resumen en la Tabla 16.

Tabla 16. Orden de prioridades entre operadores

Nivel de prioridad	Operadores
1.	Paréntesis (que, en realidad, no es un operador, pero sirve para dar prioridades a operaciones determinadas dentro de una expresión que contenga varias operaciones)
2.	El operador cambio de signo
3.	`@ Not`
4.	`* / div mod and shl shr` (Operadores multiplicativos)
5.	`+ - or xor` (Operadores aditivos)
6.	`= <> > < >= <= in` (Operadores de relación)

Las secuencias de operadores de igual prioridad normalmente se evalúan de izquierda a derecha dentro de una expresión, aunque, en algunos casos, el compilador puede reordenar los operandos durante el proceso de compilación para generar código objeto óptimo para su posterior ejecución. En muchas ocasiones se recomienda el uso de los paréntesis para hacer que las expresiones sean más claras y fáciles de entender. En la Tabla 17 se muestran algunos ejemplos de expresiones y de los correspondientes resultados al ser evaluadas.

Tabla 17. Expresiones y resultados correspondientes

Expresión	Resultado de la evaluación
`2*3+4`	`10`
`2+3*4`	`14`
`(2+3)*4`	`20`
`17 div 2 –1`	`7`

Las reglas de evaluación de expresiones pueden resumirse en las siguientes:

a) Un operando situado entre dos operadores de diferente prioridad se liga al operador de mayor prioridad.
b) Un operando situado entre dos operadores de igual prioridad se liga al operador de la izquierda.
c) Las expresiones entre paréntesis se evalúan primeramente para ser tratadas como operandos simples.

Bibliografía básica

- **García-Beltrán, A., Martínez, R.** y **Jaén, J.A.** *Métodos Informáticos en TurboPascal*, Ed. Bellisco, 2ª edición, Madrid, 2002
- **Joyanes, L.** *Fundamentos de programación, Algoritmos y Estructuras de Datos*, McGraw-Hill, Segunda edición, 1996
- **Duntemann, J.** *La Biblia de TurboPascal*, Anaya Multimedia, Madrid, 1991