






Tema 2

Elementos Básicos de la Programación Imperativa

Notación BNF


La notación BNF permite realizar la definición formal de las reglas sintácticas de un lenguaje de programación. Se basa en la descripción de cada elemento gramatical en función de otros más sencillos, según determinados esquemas o construcciones. Cada uno de estos esquemas se define mediante una *regla de producción*.

Estas reglas sobre cómo ha de escribirse los elementos del lenguaje en forma de símbolos utilizan a su vez otros símbolos, que se denominan *metasímbolos*. Son los siguientes:

Meta símbolo	Uso
	Metasímbolo de definición. Indica que el elemento a su izquierda puede desarrollarse según el esquema de la derecha.
	Metasímbolo de alternativa. Indica que puede elegirse uno y sólo uno de los elementos separados por este metasímbolo.
	Metasímbolo de repetición. Indican que los elementos incluidos dentro de ellos se pueden repetir cero o más veces.
	Metasímbolo de opción. Indican que los elementos incluidos dentro de ellos pueden ser utilizados o no.
	Metasímbolo de agrupación. Agrupan los elementos incluidos en su interior.

También se emplearán distintos estilos de letra para distinguir los elementos simbólicos siguientes:

Elemento_no_terminal

Este estilo se emplea para escribir el nombre de un elemento gramatical que habrá de ser definido por alguna regla. Cualquier elemento a la izquierda del metasímbolo  será no terminal y aparecerá con este estilo.

Elemento_terminal

Este estilo se emplea para representar los elementos que forman parte del lenguaje Modula-2, es decir, que constituyen el texto de un programa. Si aparecen en una regla deberán escribirse exactamente como se indican.

Valores y tipos

- **Dato:** es un elemento de información que puede tomar un valor entre varios posibles.
- **Constante:** Cuando un dato tiene siempre necesariamente un valor fijo.
- **Tipos:** son las distintas clases de valores que puede tomar un dato.
- **Tipos abstractos de datos:** Identifican tanto el conjunto de valores que pueden tomar los datos de ese tipo como las operaciones significativas que pueden hacerse con dichos valores.

Representación de valores constantes

Uno de los objetivos de los lenguajes de programación es evitar las ambigüedades o imprecisiones que existen en los lenguajes humanos.

- **Valores numéricos enteros:** Los valores enteros representan un número exacto de unidades, y no pueden tener parte fraccionaria. Un valor entero se escribe mediante una secuencia de uno o más dígitos del 0 al 9 sin separadores de ninguna clase entre ellos y precedidos opcionalmente de los símbolos más (+) o menos (-).
- **Valores Numéricos Reales:** Los valores numéricos reales permiten expresar cualquier cantidad, incluyendo fracciones de unidad. Se pueden representar de dos maneras distintas:
 1. **En la notación decimal habitual:** Un valor real se escribe con una parte entera terminada siempre por un punto (.), y seguida opcionalmente por una secuencia de dígitos que constituyen la parte fraccionaria decimal.
 2. **En la notación científica:** Un número real se escribe como una mantisa, que es un número real en la notación decimal habitual, seguida de un factor de escala que se escribe como la letra E seguida del exponente entero de una potencia de 10 por la que se multiplica la mantisa.

- **Caracteres:** Los valores de caracter nos permiten representar los valores correspondientes a los caracteres de un texto. El valor de un caracter concreto se escribe poniendo dicho caracter entre apóstrofes (') o comillas(").

Observaciones:

- El espacio en blanco (' ') es un caracter válido como los demás.
- Hay que distinguir entre un valor entero de un dígito y el caracter correspondiente a dicho dígito.
- El caracter apóstrofo sólo se puede representar entre comillas, y viceversa.

La colección o *juego de caracteres* que pueden manipularse en un programa depende de la máquina que se esté usando.

- **Ristras de caracteres (Strings):** Una ristra o cadena de caracteres se escribe como una secuencia de caracteres incluidos entre apóstrofes (') o comillas(").

Observaciones:

- Si una ristra incluye el caracter apóstrofes en su interior sólo se podrá escribir entre comillas, y viceversa.
- Un valor de tipo caracter se representa igual que una ristra de un único carácter, aunque en realidad son valores de tipos diferentes.
- Es posible definir una ristra vacía que no contenga ningún caracter.

Tipos predefinidos

Dentro de una misma clase de valores pueden distinguirse varios tipos diferentes, tanto a nivel de tipos predefinidos en el lenguaje, como en forma de tipos definidos por el programador.

Un tipo de datos define:

1. Una colección de valores posibles.
 2. Las operaciones significativas sobre ellos.
- **El tipo INTEGER:** Los valores de este tipo son los valores numéricos enteros positivos y negativos. Los rangos más comúnmente usados son los siguientes:

-32.768 ... 0 ... 32.767

-2.147.483.648 ... 0 ... 2.147.483.647

Se puede hacer referencia al rango de valores mediante las expresiones:

MIN(INTEGER) ... 0 ... MAX(INTEGER)

Se pueden utilizar los siguientes operadores:

Operador	Uso
+	Suma de enteros / Positivo
-	Resta de enteros / Negativo
*	Multiplicación de enteros
DIV	División de enteros
MOD	Resto de la división entera

- **El tipo CARDINAL:** Los valores de este tipo son los valores numéricos enteros positivos. Los rangos más comúnmente usados son los siguientes:

0 ... 65.535

0 ... 4.294.967.295

Se puede hacer referencia al rango de valores mediante las expresiones:

0 ... MAX(CARDINAL)

Se pueden utilizar los siguientes operadores:

Operador	Uso
+	Suma de cardinales / Positivo
-	Resta de cardinales
*	Multiplicación de cardinales
DIV	División de cardinales
MOD	Resto de la división cardinal

- **El tipo REAL:** Con este tipo se trata de representar en el computador los valores numéricos reales positivos y negativos. Sin embargo esta representación puede no ser exacta, y además, dado que la capacidad de los computadores es limitada, la representación sólo se puede considerar válida dentro de un rango. Los rangos más comúnmente usados son los siguientes:

-3.4E+38 ... -1.2E-38

0

+1.2E-38 ... +3.4E+38

(32 bits precisión: 6 cifras decimales)

-1.7E+308 ... -2.3E-308

0

+2.3E-308 ... +1.7E+308

(64 bits precisión: 15 cifras decimales)

Se pueden utilizar los siguientes operadores:

Operador	Uso
+	Suma de reales / Positivo
-	Resta de reales / Negativo
*	Multiplicación de reales
/	División de reales

- **El tipo CHAR:** Este tipo incluye como valores todos los caracteres disponibles en el computador. La tabla de caracteres que habitualmente se adopta en la mayoría de los computadores se conoce como tabla ASCII. Se pueden usar las siguientes funciones:

Función	Uso
CHR(X)	Devuelve el caracter correspondiente a la posición X de la tabla de caracteres.
ORD(C)	Devuelve la posición que ocupa el caracter C en la tabla de caracteres.

Expresiones aritméticas

Una expresión aritmética representa un cálculo a realizar con valores numéricos o de otro tipo. Una expresión aritmética es una combinación de operandos y operadores. Para indicar el orden en que se quieren realizar las operaciones parciales se pueden utilizar paréntesis.

Si no se utilizan paréntesis el orden de las operaciones depende de una jerarquía entre los operandos empleados, que es la siguiente:

Operador	Uso
1º Multiplicativos	- / DIV MOD
2º Aditivos	+ -

Operaciones de escritura simples

Las acciones que envían resultados al exterior se llaman, en general, *operaciones de escritura*, con independencia de que se trate o no de una impresión en papel, o la simple visualización, o la grabación de los datos en un soporte donde queden registrados.

- **Procedimientos WriteInt y WriteCard:** Los procedimientos WriteInt y WriteCard pertenecen al módulo *InOut*. La sintaxis es la siguiente:

```
WriteInt(ValEntero, NumEspacios)
WriteCard(ValCardinal, NumEspacios)
```

Estas escriben el valor numérico del tipo concreto que corresponde: INTEGER o CARDINAL, respectivamente ocupando el número de espacios indicado.

- **Procedimiento WriteReal:** El procedimiento WriteReal pertenece al módulo *RealInOut*. La sintaxis es la siguiente:

```
WriteReal( VarReal, NumEspacios )
```

Esta escribe en valor numérico del tipo REAL ocupando el número de espacios indicado.

- **Procedimientos Write y WriteString:** Los procedimientos Write y WriteString pertenecen al módulo *InOut*. La sintaxis es la siguiente:

```
Write( ValCaracter )
WriteString( RistraCaracteres )
```

Estas escriben un carácter del tipo CHAR o una ristra de caracteres.

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

```
WriteLn
```

Provoca que los siguientes resultados se escriban desde el comienzo de la siguiente línea de escritura.

Estructura de un programa completo

- **Uso de comentarios:** Los comentarios son información adicional que se incluye dentro del programa fuente para documentarlo y facilitar su comprensión. El compilador los ignora. Estos se escriben entre los símbolos:

```
(* ... *)
```

Los comentarios se pueden anidar uno dentro de otro.

- **Descripción formal de la estructura de un programa:** La estructura de un programa es la siguiente:

```
MODULE NombreDePrograma;
  FROM Módulo IMPORT ListaDeNombres;
BEGIN
  Sentencias;
  ...
END NombreDePrograma.
```