::=	Se define como
|	Opción alternativa
(a | b)	a ó b, pero no ambos
[algo]	0 ó 1 vez algo
{ algo }	0 ó más veces algo
'xyz'	símbolo terminal
.	final de la regla

Notación BNF (MR 110-112)

La notación BNF fue creada por J. Backus y P.Naur para describir la sintaxis del lenguaje ALGOL 60.

Utiliza reglas que se construyen con tres tipos de símbolos

metasímbolos : son los propios de la BNF.

símbolos terminales : son los que se usan en el texto del programa tal como aparecen en la regla (sin las comillas).

símbolos no terminales : son los demás. Se definen usando combinaciones de símbolos terminales, no terminales y metasímbolos

Metasímbolos

Una tabla de metasímbolos puede ser

metasímbolo	Significado
::=	Se define como
|	Opción alternativa
(a | b)	a ó b, pero no ambos
[algo]	0 ó 1 vez algo
{ algo }	0 ó más veces algo
'xyz'	símbolo terminal
.	final de la regla

Para *definir un dígito* hacen falta 10 reglas

```
digito ::= '0'.
digito ::= '1'.
---
digito ::= '9'.
```

También se puede abreviar con una regla con alternativas

```
digito ::= '0' | '1' | '2' | .. | '9'.
```

Esta última regla se leería en español como 'un digito se define como un cero o un uno o un dos ... o un nueve.

La *definición de un signo*

```
signo ::= '+' | '-' .
```

La *definición de un entero_sin_signo*

```
entero_sin_signo ::= digito | digito entero_sin_signo.
```

También se puede definir el entero_sin_signo usando las llaves de EBNF

```
entero_sin_signo ::= digito { digito }
```

La definición de un *entero*

```
entero ::= [ signo ] entero_sin_signo
```

La definición de un *identificador* es

```
identificador ::= letra { [ guión_bajo ] (letra | digito) }
```

Definición de sentencia

```
sentencia ::= [etiqueta ':'](sentencia simple | sentencia estructurada).
```

```
sentencia_vacía ::= .
```

La sentencia vacía NO es el espacio

```
sentencia_de_asignación ::= (variable | función) ':=' expresión.
```

```
sentencia_goto ::= 'goto' etiqueta.
```

```
sentencia_simple ::=
      sentencia_vacia          |
      sentencia_de_asignación  |
      sentencia_goto           |
      sentencia_de_procedimiento.
```

Una *secuencia de sentencias* es

```
secuencia_de_sentencias ::= sentencia {; sentencia}.
```

La definición para la *sentencia compuesta* es

```
sentencia_compuesta ::= 'begin' secuencia_de_sentencias 'end'.
```

La definición de la *sentencia if* es

```
sentencia_if ::=
    'if' exp_log
    'then' sentencia
    ['else' sentencia].
```

La definición de la *sentencia while* es

```
sentencia_while ::= 'while' exp_log 'do' sentencia.
```

La definición de la *sentencia repeat* es

```
sentencia_repeat ::=
    'repeat'
        secuencia_de_sentencias
    'until' exp_log.
```

La definición de la *sentencia for* es

```
sentencia_for ::=
   'for' identificador ':=' exp_ord
    ('to' | 'downto') exp_ord 'do' sentencia
```

La definición de *case* es

```
sentencia_case ::= 'case' selector 'of'
                      elemento_case
                      {';' elemento_case}
                      [ [';'] completador_case ]
                   'end'
```

donde

```
completador_case ::= 'otherwise' secuencia_de_sentencias.
```

```
elemento_de_case ::= lista_constantes_case ':' sentencia.
```

```
lista_constantes_case ::= grupo_case {',' grupo_case}.
```

```
grupo_case ::= constante_case | constante_case '..' constante_case.
```

Ejercicios

Consultando la BNF del manual de referencia impreso, responda a las siguientes preguntas. ¡Razone la respuesta!

1) Una secuencia de sentencias tiene como mínimo.

A. 0 sentencias.
B. 1 sentencia.
C. 2 sentencias.

2) Para el estándar, el número máximo de dígitos de un entero es

A. 8
B. 10
C. indefinido

3) Un identificador puede comenzar por

A. Un dígito
B. Una letra
C. El símbolo de guión bajo (_)

4) Un identificador NO puede terminar por

A. Dígito
B. Letra
C. El símbolo de guión bajo (_)

5) La parte entera de un número real debe constar de al menos

A. 0 dígitos (no es obligatoria).
B. 1 dígito.
C. 2 dígitos.