






## Conceptos y Paradigmas de Lenguajes de Programación - Práctica 2

### EJERCICIO 1

Meta símbolos utilizados por		Símbolo utilizado en Diagramas sintácticos	Significado
BNF	EBNF		
palabra terminal	palabra terminal	óvalo 	Definición de un elemento terminal
< >	< >	rectángulo 	Definición de un elemento no terminal
::=	::=	diagrama con rectángulos, óvalos y flechas	Meta-símbolo de definición que indica que el elemento a su izquierda se puede definir según el esquema de la derecha
	( )	flecha que se divide en dos o más caminos	Selección de una alternativa
< p > < p1 >	{ }		Repetición
	*		Repetición de 0 o más veces
	+		Repetición de 1 o más veces
	[ ]		Opcional, está presente o no lo está

### EJERCICIO 2

**Sintaxis:** Conjunto de reglas que definen como componer letras, dígitos y otros caracteres para formar los programas.

**Semántica:** Conjunto de reglas para dar significado a los programas sintácticamente válidos.

La definición de la **sintaxis** y la **semántica** de un lenguaje de programación benefician a los programadores y al implementador (compilador) y proporcionan mecanismos para que una persona o una computadora pueda decir:

- Si el programa es válido.
- Si lo es, qué significa.

## Características de la sintaxis

- La sintaxis debe ayudar al programador a escribir programas correctos sintácticamente.
- La sintaxis establece reglas que sirven para que el programador se comunique con el procesador.
- La sintaxis debe contemplar soluciones a características tales como:
  - Legibilidad
  - Verificabilidad
  - Traducción
  - Falta de ambigüedad

La sintaxis establece reglas que definen cómo deben combinarse las componentes básicas, llamadas **word**, para formar sentencias y programas.

## Elementos de la sintaxis

- **Alfabeto o conjunto de caracteres**
  - Conjunto finito de símbolos admitidos en el lenguaje
  - La secuencia de bits que compone cada carácter la determina la implementación
  - Importante: *Tener en cuenta qué conjunto de caracteres se trabaja sobre todo por el orden a la hora de comparaciones.*
- **Identificadores**
  - Elección más ampliamente utilizada: Cadenas de letras y dígitos, que deben comenzar con una letra
  - Si se restringe la longitud se pierde legibilidad
- **Operadores**
  - Con los operadores de suma, resta, etc. la mayoría de los lenguajes utilizan +, -. En los operadores no hay tanta uniformidad
- **Comentarios y uso de blancos**
  - Hacen los programas más legibles
- **Palabra clave y palabra reservada** (ej: *array, do, else, if*)
  - Palabra clave o keywords son palabras que tienen un significado dentro de un contexto
  - Palabra reservada son palabras claves que además no pueden ser usadas por el programador como identificador de otra entidad.

## EJERCICIO 3

### Estructura sintáctica

- **Vocabulario o words:**
  - Conjunto de caracteres y palabras necesarias para construir expresiones, sentencias y programas. Ej: identificadores, operadores, palabras claves, etc.
- **Expresiones:**
  - Son construcciones sintácticas compuestas de operadores y operandos, de cuya evaluación se obtiene un valor.
  - Son bloques sintácticos básicos a partir de los cuales se construyen las sentencias y los programas.
- **Sentencias:**
  - Componente sintáctico más importante
  - Tiene un fuerte impacto en la facilidad de escritura y legibilidad

- Hay sentencias simples, estructuradas y anidadas

### Reglas léxicas y sintácticas

- **Reglas léxicas:** Conjunto de reglas para formar las **word**, a partir de los caracteres del alfabeto. *Ej: Diferencias entre mayúsculas y minúsculas / Símbolo de distinto en C ! =, en Pascal <>.*
- **Reglas sintácticas:** Conjunto de reglas que definen cómo formar las **expresiones** y **sentencias** a partir de esas palabras. *Ej: El If en C no lleva then, en Pascal si.*

### EJERCICIO 4

#### Palabras reservadas

Son palabras claves que además no pueden ser usadas por el programador como identificador de otra entidad.

En el contexto de la **gramática formal**, las palabras reservadas son equivalentes a **símbolos terminales**. Los **símbolos terminales** son las unidades básicas que no se pueden descomponer más en el proceso de análisis sintáctico y representan las palabras específicas del lenguaje, como las palabras clave del lenguaje de programación.

- Ventajas de uso:
  - Permiten al compilador y al programador expresarse claramente
  - Hacen los programas más legibles y permiten una rápida traducción
- Ejemplos de lenguajes con uso de palabras reservadas:
  - **C:** auto, break, case, char, const, continue, default, do, double, else, enum, extern, float, for, goto, if, int, etc.
  - **Pascal:** absolute, and, array, begin, const, div, do, downto, else, if, label, mod, not, of, packed, procedure, record, set, shr, then, to, unit, uses, var, while, xor, etc.

### EJERCICIO 5

#### A) Componentes:

- Elementos No Terminales: <numero\_entero>, <digito>
- Elementos Terminales: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
- Símbolos: <numero\_entero>
- Producciones:
  - <numero\_entero> ::= <digito><numero\_entero> | <numero\_entero><digito> | <digito>
  - <digito> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

#### B) Ambigüedad y corrección

Es una gramática ambigua porque un número entero puede derivarse de más de una manera. Tiene recursión de ambos lados: por izquierda y por derecha.

Para eliminar la ambigüedad se corrige la recursión para que sea solamente por un lado.

#### Posibles soluciones:

Corrección con recursión por la **derecha**:

- <numero\_entero> ::= <digito> | <digito><numero\_entero>
- <digito> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Otra opción de recursión por la **derecha**:

- <numero\_entero> ::= <digito><numero\_entero> | <digito>
- <digito> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Corrección con recursión por la **izquierda**:

- $\langle \text{numero\_entero} \rangle ::= \langle \text{digito} \rangle \mid \langle \text{numero\_entero} \rangle \langle \text{digito} \rangle$
- $\langle \text{digito} \rangle ::= 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$

Otra opción de recursión por la **izquierda**:

- $\langle \text{numero\_entero} \rangle ::= \langle \text{numero\_entero} \rangle \langle \text{digito} \rangle \mid \langle \text{digito} \rangle$
- $\langle \text{digito} \rangle ::= 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$

## EJERCICIO 6

**PREGUNTAR: debería identificar entre mayúsculas y minúsculas con símbolos no terminales ( $\langle \text{minúscula} \rangle \langle \text{mayúscula} \rangle$ )? y las tildes?**

$G = (N, T, S, P)$

$N = \{ \langle \text{palabra} \rangle, \langle \text{letra} \rangle \}$

$T = \{ a, b, c, d, e, f, g, h, i, j, k, l, m, n, \tilde{n}, o, p, q, r, s, t, u, v, w, x, y, z, A, B, C, D, E, F, G, H, I, J, K, L, M, N, \tilde{N}, O, P, Q, R, S, T, U, V, W, X, Y, Z \}$

$S = \{ \langle \text{palabra} \rangle \}$

$P = \{$

$\langle \text{palabra} \rangle ::= \langle \text{letra} \rangle \mid \langle \text{letra} \rangle \langle \text{palabra} \rangle$

$\langle \text{letra} \rangle ::= a \mid b \mid c \mid d \mid e \mid f \mid g \mid h \mid i \mid j \mid k \mid l \mid m \mid n \mid \tilde{n} \mid o \mid p \mid q \mid r \mid s \mid t \mid u \mid v \mid w \mid x \mid y \mid z$   
 $\mid A \mid B \mid C \mid D \mid E \mid F \mid G \mid H \mid I \mid J \mid K \mid L \mid M \mid N \mid \tilde{N} \mid O \mid P \mid Q \mid R \mid S \mid T \mid U \mid V \mid W \mid X \mid Y \mid Z$   
 $\}$

### **Componentes:**

- Elementos No Terminales:  $\langle \text{palabra} \rangle, \langle \text{letra} \rangle$
- Elementos Terminales:  $a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z, A, B, C, D, E, F, G, H, I, J, K, L, M, N, \tilde{N}, O, P, Q, R, S, T, U, V, W, X, Y, Z$
- Símbolos:  $\langle \text{palabra} \rangle$
- Producciones:

-  $\langle \text{palabra} \rangle ::= \langle \text{letra} \rangle \mid \langle \text{letra} \rangle \langle \text{palabra} \rangle$

-  $\langle \text{letra} \rangle ::= a \mid b \mid c \mid d \mid e \mid f \mid g \mid h \mid i \mid j \mid k \mid l \mid m \mid n \mid \tilde{n} \mid o \mid p \mid q \mid r \mid s \mid t \mid u \mid v \mid w \mid x \mid y \mid z$   
 $\mid A \mid B \mid C \mid D \mid E \mid F \mid G \mid H \mid I \mid J \mid K \mid L \mid M \mid N \mid \tilde{N} \mid O \mid P \mid Q \mid R \mid S \mid T \mid U \mid V \mid W \mid X \mid Y \mid Z$

**Nota: puedo poner  $\{ a, b, \dots, z, A, B, \dots, Z \}$  y  $\{ a \mid b \mid \dots \mid z \mid A \mid B \mid \dots \mid Z \}$**

## EJERCICIO 7

### **EBNF**

$G = (N, T, S, P)$

$N = \{ \langle \text{real} \rangle, \langle \text{digito} \rangle \}$

$T = \{ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, "+", "-", ",", " " \}$

$S = \{ \langle \text{real} \rangle \}$

$P = \{$

$\langle \text{real} \rangle ::= [(+|-)] \{ \langle \text{digito} \rangle \}^+ , \{ \langle \text{digito} \rangle \}^+ \text{ otra opción } // \langle \text{real} \rangle ::= [-] \{ \langle \text{digito} \rangle \}^+ [ , \{ \langle \text{digito} \rangle \}^+ ]$

$\langle \text{digito} \rangle ::= 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$

$\}$

## BNF

$G = (N, T, S, P)$

$N = \{ \langle \text{real} \rangle, \langle \text{entero\_sig} \rangle, \langle \text{entero} \rangle, \langle \text{decimal} \rangle, \langle \text{digito} \rangle \}$

$T = \{ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, "+", "-", ",", " " \}$

$S = \{ \langle \text{real} \rangle \}$

$P = \{$

$\langle \text{real} \rangle ::= \langle \text{entero\_sig} \rangle \mid \langle \text{entero\_sig} \rangle \langle \text{decimal} \rangle$

$\langle \text{entero\_sig} \rangle ::= + \langle \text{entero} \rangle \mid - \langle \text{entero} \rangle \mid \langle \text{entero} \rangle$

$\langle \text{entero} \rangle ::= \langle \text{digito} \rangle \mid \langle \text{digito} \rangle \langle \text{entero} \rangle$

$\langle \text{decimal} \rangle ::= , \langle \text{entero} \rangle$

$\langle \text{digito} \rangle ::= 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$

$\}$

## Diferencias

La diferencia más clara entre las dos formas es que se necesitan más símbolos no terminales para definir en BNF lo mismo que definimos en EBNF, lo que hace que sea menos legible.

Además, en la forma BNF se precisa el uso de recursión, lo que en la forma EBNF se resuelve con repetición.

13) No se puede resolver porque tendrías que tener un condicional y en la gramática no se puede poner una condición.