
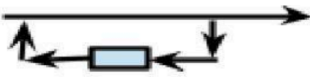

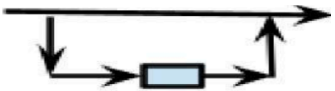


Práctica Nro. 2

Sintaxis

Objetivo: conocer como se define léxicamente un lenguaje de programación y cuales son las herramientas necesarias para hacerlo

Ejercicio 1: Complete el siguiente cuadro:

Meta símbolos utilizados por		Símbolo utilizado en Diagramas sintácticos	Significado
BNF	EBNF		
palabra terminal	palabra terminal		Definición de un elemento terminal
<>	<>	rectángulo 	Definición de un elemento no terminal
::=	::=	diagrama con rectángulos, óvalos y flechas	definición de una producción
	()	flecha que se divide en dos o más caminos	selección de una alternativa
< p > < p1 >	{ }	Repetición	Repetición
	*		Repetición de 0 o más veces
	+		Repetición de 1 o más veces
	[]		Opcional está presente o no lo está

Ejercicio 2: ¿Cuál es la importancia de la sintaxis para un lenguaje? ¿Cuáles son sus elementos?

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
- identificadores
- Operadores
- Palabra clave y palabra reservada
- Comentarios y uso de blancos

Ejercicio 3: ¿Explique a qué se denomina regla lexicográfica y regla sintáctica?

Reglas Lexicográficas: Conjunto de reglas para formar las “word”, a partir de los caracteres del alfabeto.

Reglas Sintácticas: Conjunto de reglas que definen cómo formar las “expresiones” y “sentencias”.

Ejercicio 4: ¿En la definición de un lenguaje, a qué se llama palabra reservadas? ¿A qué son equivalentes en la definición de una gramática? De un ejemplo de palabra reservada en el lenguaje que más conoce. (Ada,C,Ruby,Python,..)

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.
- Ventajas de su uso:
 - Permiten al compilador y al programador expresarse claramente
 - Hacen los programas más legibles y permiten una rápida traducción

Ejemplo: if , else

Conceptos y Paradigmas de Lenguajes de Programación 2025

Ejercicio 5: Dada la siguiente gramática escrita en BNF:

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

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

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

$S = \langle \text{número entero} \rangle$

$P = \{$

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

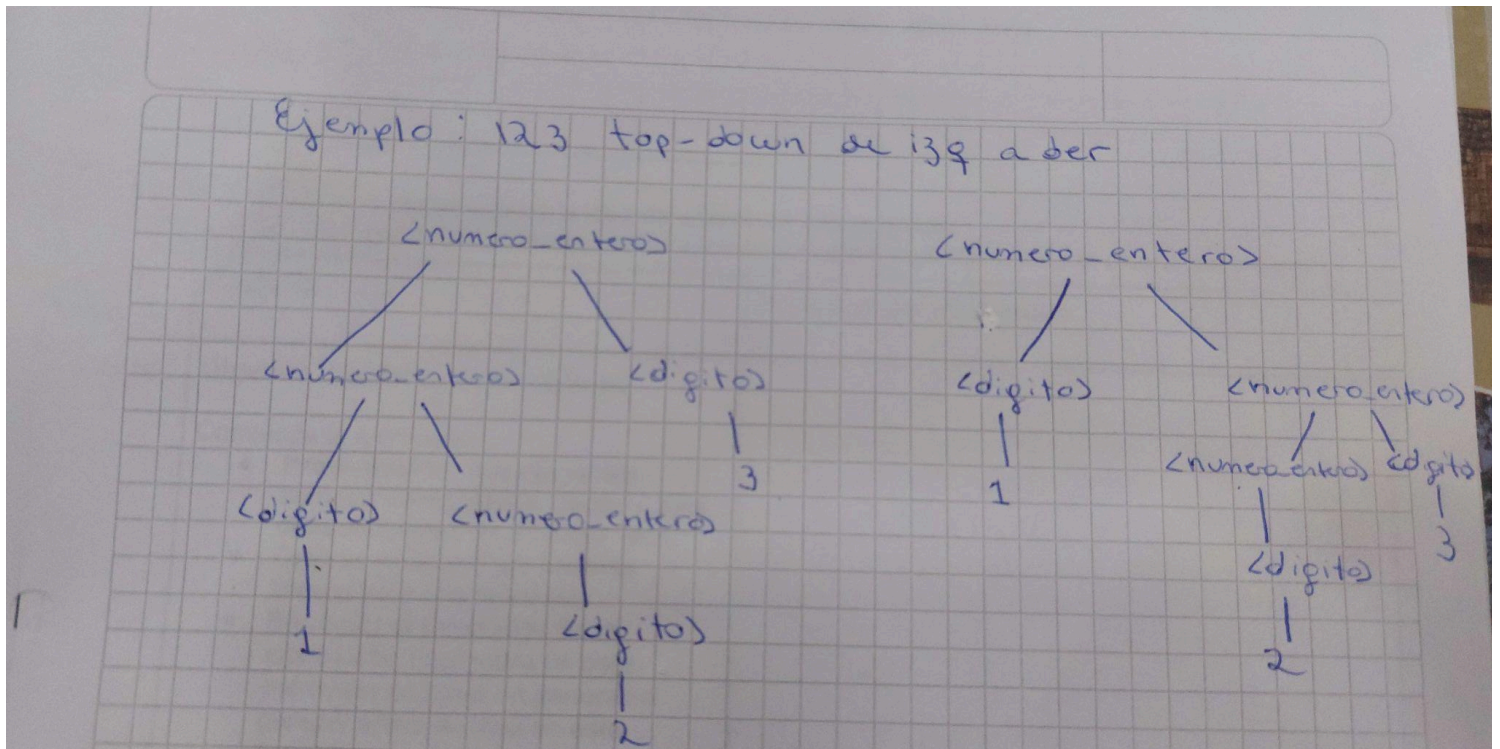
$\langle \text{digito} \rangle ::= 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$

$\}$

a- Identifique las componentes de la misma

- G: 4-tupla que define un conjunto de reglas finitas que define un conjunto infinito de posibles sentencias válidas en el lenguaje.
- N: Conjunto de símbolos no terminales.
- T: Conjunto de símbolos terminales.
- S: Símbolo distinguido de la gramática que pertenece a N.
- P: Conjunto de producciones.

b- Indique porqué es ambigua y corríjala



Lo que hace ambigua a la gramática es esta sentencia: $\langle \text{numero_entero} \rangle ::= \langle \text{digito} \rangle \langle \text{numero_entero} \rangle \mid \langle \text{numero_entero} \rangle \langle \text{digito} \rangle$ ya que de dos formas distintas se obtiene la misma sentencia.

solución.

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

$N = \{ \}$

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

$S =$

$P = \{ ::= \mid ::= 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9 \}$

Ejercicio 6: Defina en BNF (Gramática de contexto libre desarrollada por Backus- Naur) la gramática para la definición de una palabra cualquiera.

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

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

$T = \{A, \dots, Z, a, \dots, z\}$

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

$P = \{$

$\langle \text{palabra} \rangle ::= \langle \text{mayuscula} \rangle \langle \text{palabra} \rangle \mid \langle \text{minúscula} \rangle \langle \text{palabra} \rangle \mid \langle \text{mayuscula} \rangle \mid \langle \text{minúscula} \rangle$

```
<mayuscula> ::= A | B | C | D | E | F | ... | X | Y | Z  
<minuscula> ::= a | b | c | d | e | f | ... | x | y | z  
}
```

Ejercicio 7: Defina en EBNF la gramática para la definición de números reales. Inténtelo desarrollar para BNF y explique las diferencias con la utilización de la gramática EBNF.

EBNF:

G = {N, T, S, P}

N = {<numero_real>, <digito>}

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

S = <numero_real>

P = {

<numero_real> ::= [(+|-) {digito}⁺ [, {<digito>⁺ }]

<digito> ::= (0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9)

BNF

G = {N, T, S, P}

N = {<numero_real>, <digito>}

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

S = <numero_real>

P = {

<numero_real> ::= <entero_sig> | <entero_sig><decimal>

<decimal> ::= , <entero>

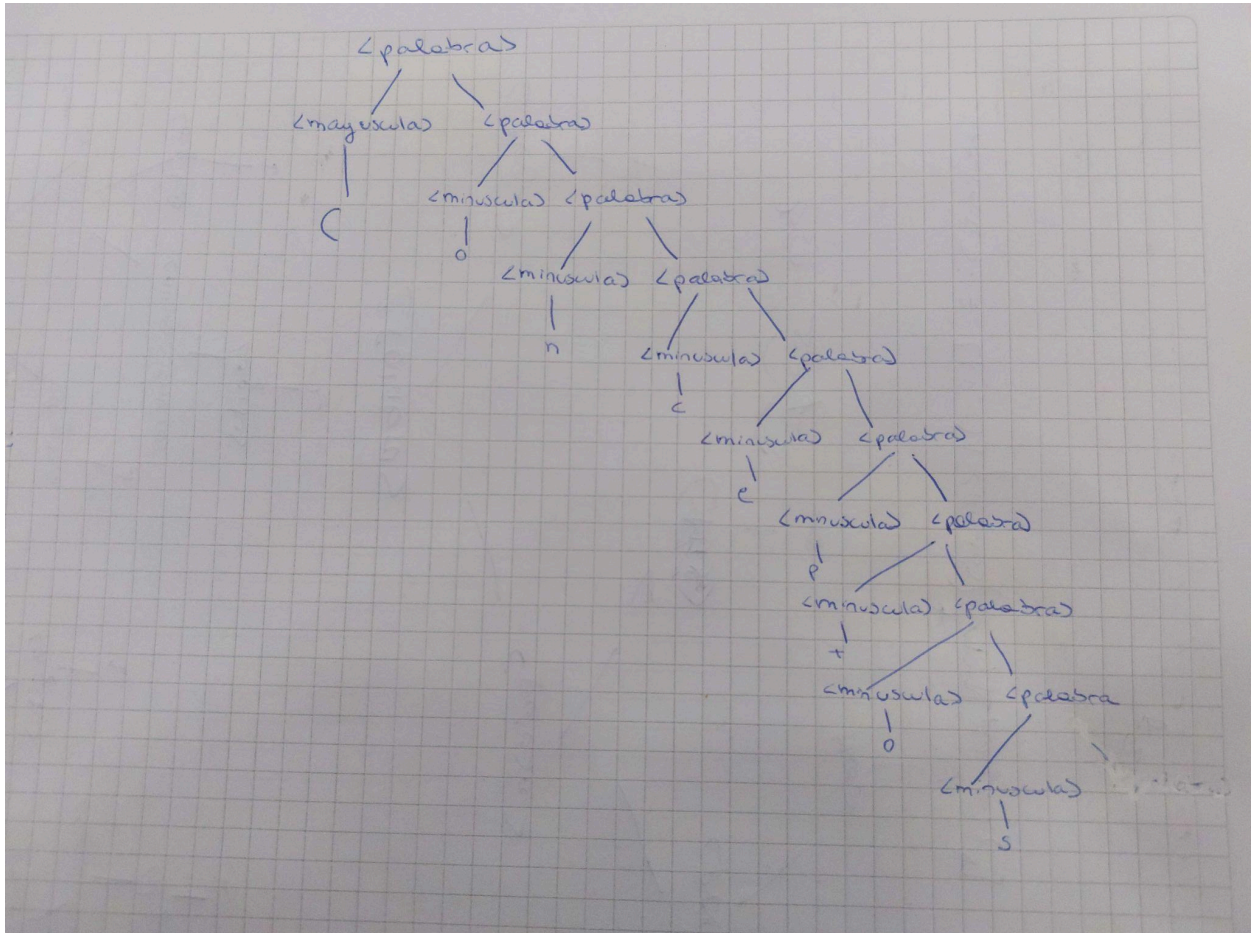
<entero_sig> ::= + <entero> | - <entero> | <entero>

<entero> ::= <digito> | <digito><entero>

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

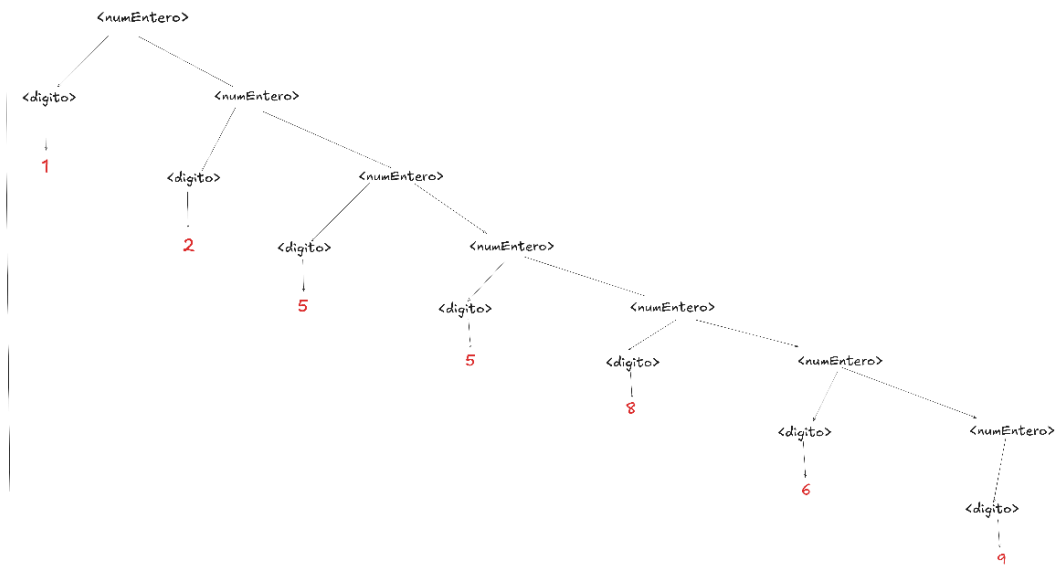
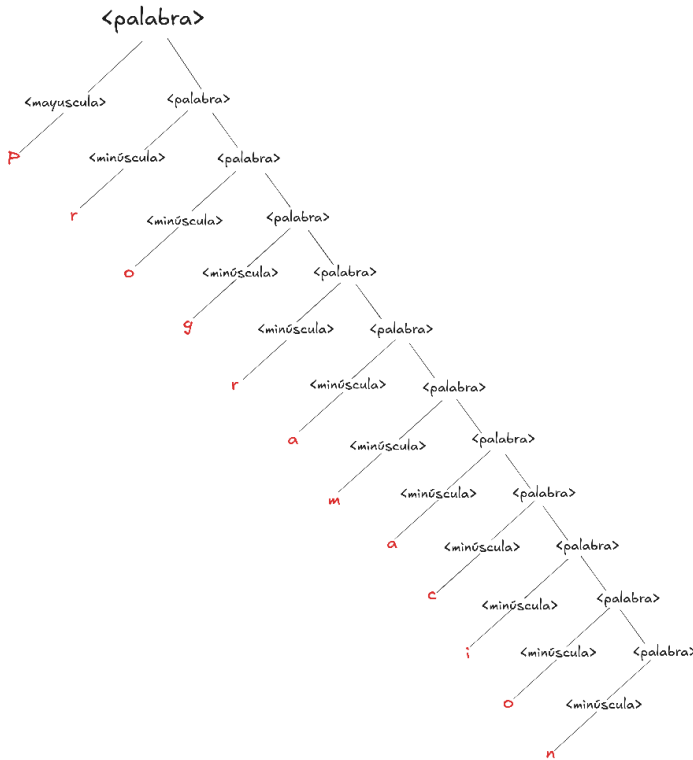
Ejercicio 8: Utilizando la gramática que desarrolló en los puntos 6 y 7, escriba el árbol sintáctico de:

a. Conceptos



b. Programación

c. 1255869



d. 854,26

e. Conceptos de lenguajes

Ejercicio 9: Defina utilizando diagramas sintácticos la gramática para la definición de un identificador de un lenguaje de programación. Tenga presente como regla que un identificador no puede comenzar con números.

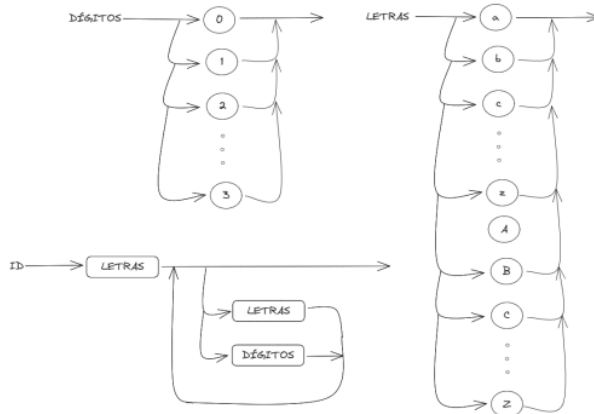
Sea la gramática para identificadores $GI = (N, T, S, P)$

$N = \{ \langle id \rangle, \langle letra \rangle, \langle digito \rangle, \langle otro \rangle \}$

$T = \{ A, \dots, Z, 0, \dots, 1 \}$

$S = \langle id \rangle$

$P = \{ \langle id \rangle ::= \langle letra \rangle \mid \langle letra \rangle \langle otro \rangle, \langle otro \rangle ::= \langle letra \rangle \mid \langle digito \rangle \mid \langle letra \rangle \langle otro \rangle \mid \langle digito \rangle \langle otro \rangle, \langle letra \rangle ::= A \mid B \mid C \mid \dots \mid Z, \langle digito \rangle ::= 0 \dots 9 \}$



Ejercicio 10:

a) Defina con EBNF la gramática para una expresión numérica, donde intervienen variables y números. Considerar los operadores +, -, * y / sin orden de prioridad. No considerar el uso de paréntesis.

EBNF:

$$G = \{N, T, S, P\}$$
$$N = \{ \langle \text{numero} \rangle, \langle \text{digito} \rangle \}$$
$$T = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, "+", "-", "/", A, \dots, Z\}$$

S= <numero>

$$P = \{$$

```
<exp> ::= <numero> <operador> <numero> | <numero> <operador> <variable> |
         <variable> <operador> <numero> | <variable> <operador> <variable>
```

$$\langle \text{numero} \rangle ::= [(+|-)] \{ \langle \text{digito} \rangle \}^+$$
$$\langle \text{variable} \rangle ::= \langle \text{letra} \rangle \{ (\langle \text{letra} \rangle \mid \langle \text{digito} \rangle) \}^*$$
$$\langle \text{operator} \rangle ::= (+ \mid - \mid * \mid /)$$
$$\langle \text{digito} \rangle ::= (0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9)$$
$$\langle \text{letra} \rangle ::= (A, \dots, Z)$$

}

b) A la gramática definida en el ejercicio anterior agregarle prioridad de operadores.

$$G = \{N, T, S, P\}$$
$$N = \{ \langle \text{expresionConPrioridad} \rangle, \langle \text{expresionSinPrioridad} \rangle, \langle \text{exp} \rangle, \langle \text{variable} \rangle, \dots \}$$

<operadorConPrioridad>, <número>, <dígito>, <letra>}

$$T = \{a, \dots, z, A, \dots, Z, 0, \dots, 9, "+", "-", "*", "/" \}$$

S = <expr>

$$P = \{$$
$$\langle \text{expr} \rangle ::= \langle \text{expr con p} \rangle \{ \langle \text{expresionConPrioridad} \rangle \langle \text{expresionConPrioridad} \rangle \}^*$$
$$\langle \text{expresionConPrioridad} \rangle ::= (\langle \text{variable} \rangle \mid \langle \text{número} \rangle) \{ \langle \text{operadorConPrioridad} \rangle (\langle \text{variable} \rangle \mid \langle \text{número} \rangle) \}^*$$
$$\langle \text{número} \rangle ::= [(+|-)] \{ \langle \text{dígito} \rangle \}^+$$
$$\langle \text{variable} \rangle ::= \langle \text{letra} \rangle \{ (\langle \text{letra} \rangle \mid \langle \text{digito} \rangle) \}^*$$
$$\langle \text{dígito} \rangle ::= (0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9)$$
$$\langle \text{operadorConPrioridad} \rangle ::= (* \mid /)$$
$$\langle \text{operadorSinPrioridad} \rangle ::= (+ \mid -)$$

$\langle \text{letra} \rangle ::= (a \mid \dots \mid z \mid A \mid \dots \mid Z)$

c) Describa con sus palabras los pasos y decisiones que tomó para agregarle prioridad de operadores al ejercicio anterior.

Para agregarle prioridad al ejercicio anterior pensamos en dividir la expresión en dos tipos de expresiones: con prioridad que usen operadores con prioridad (* o /) y sin prioridad que usen operadores sin prioridad (+ o -). En base a esto creamos una expresión general en la cual se pueden realizar 3 acciones:

- Trabajar únicamente con expresiones con prioridad entre 1 o n veces
- Trabajar únicamente con expresiones sin prioridad entre 1 o n veces
- Trabajar con ambas expresiones entre 1 y n veces, siempre calculando primero las expresiones con prioridad.

Ejercicio 11: La siguiente gramática intenta describir sintácticamente la sentencia for de ADA, indique cuál/cuáles son los errores justificando la respuesta.

$N = \{ \langle \text{sentencia_for} \rangle, \langle \text{bloque} \rangle, \langle \text{variable} \rangle, \langle \text{letra} \rangle, \langle \text{cadena} \rangle, \langle \text{digito} \rangle, \langle \text{otro} \rangle, \langle \text{operacion} \rangle, \langle \text{llamada_a_funcion} \rangle, \langle \text{numero} \rangle, \langle \text{sentencia} \rangle \}$

$P = \{ \langle \text{sentencia_for} \rangle ::= \text{for } (i = \text{IN } 1..10) \text{ loop } \langle \text{bloque} \rangle \text{ end loop};$
 $\langle \text{variable} \rangle ::= \langle \text{letra} \rangle \mid \langle \text{cadena} \rangle$
 $\langle \text{cadena} \rangle ::= \{ (\langle \text{letra} \rangle \mid \langle \text{digito} \rangle \mid \langle \text{otro} \rangle) \}^+$
 $\langle \text{letra} \rangle ::= (a \mid \dots \mid z \mid A \mid \dots \mid Z)$
 $\langle \text{digito} \rangle ::= (1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9 \mid 0)$
 $\langle \text{bloque} \rangle ::= \langle \text{sentencia} \rangle \mid \langle \text{sentencia} \rangle \langle \text{bloque} \rangle \mid \langle \text{bloque} \rangle \langle \text{sentencia} \rangle ;$
 $\langle \text{sentencia} \rangle ::= \langle \text{sentencia_asignacion} \rangle \mid \langle \text{llamada_a_funcion} \rangle \mid \langle \text{sentencia_if} \rangle \mid$
 $\langle \text{sentencia_for} \rangle \mid \langle \text{sentencia_while} \rangle \mid \langle \text{sentencia_switch} \rangle$
}

- No hay elementos terminales T
- Sintaxis incorrecta en for (i= IN 1..10), $\langle \text{sentencia_for} \rangle ::= \text{for } \langle \text{variable} \rangle \text{ in } \langle \text{rango} \rangle \text{ loop } \langle \text{bloque} \rangle \text{ end loop} ;$
- $\langle \text{sentencia_asignacion} \rangle$ $\langle \text{llamada_a_funcion} \rangle$ $\langle \text{sentencia_if} \rangle$ $\langle \text{sentencia_while} \rangle$ $\langle \text{sentencia_switch} \rangle$ $\langle \text{otro} \rangle$ no se encuentran definidos
- No está definido S
- $\langle \text{sentencia} \rangle \langle \text{bloque} \rangle \mid \langle \text{bloque} \rangle \langle \text{sentencia} \rangle$: es una gramática ambigua porque una sentencia puede derivarse de más de una forma

Conceptos y Paradigmas de Lenguajes de Programación 2025

Ejercicio 12: Realice en EBNF la gramática para la definición de un tag div en html 5.

(Puede ayudarse con el siguiente enlace

(<https://developer.mozilla.org/es/docs/Web/HTML/Elemento/div>)

```
<div class="warning">
  
  <p>Beware of the leopard</p>
</div>
```

G (N,T,S,P)

N= {

T= {

S= {

P= {

<div> ::= <tag_a> div <tag_c> { ({ [<clase>] } * | { [] } * | { [<p>] } *) } *
<tag_a> /div <tag_c>

<clase> ::= class = " { (<letras> [<especiales>] | <digitos> [<especiales>]) } + "

 ::= <tag_a> <ruta> [<alt>] /img <tag_c>

<ruta> ::= src = " { <separadores> <caracteres> } + "

<alt> ::= alt = " { (<letras> | <digitos> | <especiales>) } + "

<p> ::= <tag_a>p <tag_c> { (<letras> | <digitos> | <especiales> | <separadores>) } * <tag_a> /p <tag_c>

tag_a ::= <

tag_c ::= >

separadores ::= (/ | \)

caracteres ::= (<letras> | <digitos> | <especiales>) + <extension> |
 <letras> + <extension> | <digitos> + <extension>

letras ::= (a | .. | z | A | .. | Z)

digitos ::= (1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0)

especiales ::= (_ | - | = | . | ,)

extension ::= . <letras> +

Ejercicio 13: Defina en EBNF una gramática para la construcción de números primos. ¿Qué debería agregar a la gramática para completar el ejercicio?

Es igual que los números enteros solo que tengo que verificar que sea primo.

Ejercicio 14: Sobre un lenguaje de su preferencia escriba en EBNF la gramática para la definición de funciones o métodos o procedimientos (considere los parámetros en caso de ser necesario)

G (N,T,S,P)

N=

T=

S= <funcion>

P= {

funcion= <def> <nombre><p_ap> [argumento] <p_cierre> :
 <sentencia> [<return>]

<def> ::= def

<nombre>::= {(<palabra> | {<digitos>}* | {<especiales>}*) }*

<palabra>::={<letra>}*

<especiales>::= (_ | - | / | \)

<p_ap>::=(

<p_ci>::=)

<sentencia>::= <sentencia_for> | <sentencia_for_range> <sentencia_while> |
<asignacion> | <sentencia_if> | <print>

<argumento>::= <variable> {,<argumento>}*

<variable>::= {(<palabra> | {<digito>}* | {<especiales>}*) }*

<sentencia_for>::= for <palabra> in <variable> : {<sentencia>}*

<sentencia_for_range> ::= for <palabra> in range <p_ap> <{digito}*><p_ci> :
{<sentencia>}*

<asignacion>::= <variable> = <variable> ;

<sentencia_while>::=while <condicion> : {<sentencia>}*

<sentencia_if>::= if <condicion> : {<sentencia>}*

<print>::=print <p_ap> (<variable> | "<palabra>")<p_ci>

<return>::=return <variable>

Conceptos y Paradigmas de Lenguajes de Programación 2025

$\langle \text{condicion} \rangle ::= (\langle \text{variable} \rangle (\langle \text{operador} \rangle \mid \langle \text{op_esp} \rangle) \langle \text{variable} \rangle) \mid \langle \text{variable} \rangle$
 $(\langle \text{operador} \rangle \mid \langle \text{op_esp} \rangle) \{ \langle \text{digito} \rangle \}^* \mid \langle \text{variable} \rangle \langle \text{operador} \rangle \langle \text{None} \rangle$

$\langle \text{operador} \rangle ::= (== \mid !=)$

$\langle \text{op_esp} \rangle ::= (> \mid < \mid >= \mid <=)$

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

}

NOTAS CLASE 2:

- Errores semánticos son los que hacen que mi código no se llegue a ejecutar
- Errores lógicos: loop infinito
- semántica estática y semántica dinamica: la diferencia es muy gris?
- semántica estática ocurre en el tiempo de compilación (antes que se ejecute), segun la catedra. Hay ciertos lenguajes interpretados que hace ciertas validaciones antes de la ejecución del programa (SEGUN LA CATEDRA). Lenguaje compilado seguro tiene semantica dinamoca,
- Ejemplo: referencia a memoria null: semantica dinámica

Diferencia entre intérprete y traducción: interprete linea a linea y traducción es el paso a lenguaje máquina.

Diferencias entre sintaxis y semántica:

- la sintaxis es el paso previo en el analisis.
- La semántica da significado a las palabras que se definieron anteriormente

-La sintaxis (gramática) puede ser distinta pero la semántica puede ser igual

-SEMANTICA ES MAS DEBIL EN LENGUAJES F/B TIPADOS

-SEMANTICA DE JAVA ES MUCHO MAS ROBUSTA

-Para el parcial estudiar todos los lenguajes