






Practica 2 CyPLP

Conceptos y Paradigmas de Lenguajes de Programación

2025

Práctica Nro. 2

Ejercicio 1

Meta símbolos utilizados por		Símbolo utilizado en Diagramas sintacticos	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 >	{ }	flecha que vuelve a la condición	Repetición
	*		Repetición de 0 o más veces
	+		Repetición de 1 o más veces
	[]		Opcional esta presente o no lo esta

Ejercicio 2

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

Reglas léxicas: Conjunto de reglas para formar las "word", a partir de los caracteres del alfabeto.

Diferencias entre mayúsculas y minúsculas

Símbolo de distinto. En C != en Pascal <>

Reglas sintácticas: Conjunto de reglas que definen cómo formar a partir de esas palabras, las "expresiones" y "sentencias".

El If en C no lleva ""then"", en Pascal si

Ejercicio 4

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.

Las palabras reservadas son equivalentes a los símbolos terminales.

Ejemplos de palabras reservadas de Pascal ej.: absolute, and, array, begin, etc.

Ejercicio 5

1. **Ejercicio 5:** Dada la siguiente gramática escrita en BNF:

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

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

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

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

$P = \{$

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

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

$\}$

a- Identifique las componentes de la misma

b- Indique porqué es ambigua y corrija

a- Las

componentes de una gramática formal en BNF (Backus-Naur Form) se definen por cuatro conjuntos:

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

Donde:

- a. **N (Símbolos No Terminales)**: Son los nombres de estructuras sintácticas que pueden descomponerse en otras.
- b. **T (Símbolos Terminales)**: Son los elementos básicos del lenguaje que no pueden descomponerse más.
- c. **S (Símbolo Inicial)**: Es el símbolo desde el cual se inicia la construcción de expresiones válidas.
- d. **P (Producciones o Reglas de Producción)**: Son las reglas que describen cómo los no terminales pueden ser reemplazados por terminales u otros no terminales.

b- La gramática es **ambigua** porque una misma cadena puede generarse de múltiples maneras con diferentes árboles de derivación.

Para corregir la ambigüedad, debemos establecer un orden claro de construcción del número. Una forma correcta es usar

recursión por la izquierda, asegurando que los dígitos se agreguen de manera ordenada:

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

Ejercicio 6

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

$N = \{ \langle \text{palabra} \rangle \langle \text{caracter} \rangle \}$

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

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

$P = \{$

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

```
<character> ::= a | b | ... | z | A | B | ... | Z
}
```

Ejercicio 7

EBNF

G=(N,T,S,P)

N={<numReal>, <digito>, <coma>}

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

S={<numEntero>}

P={

<numReal>::= <numeroEntero> [<coma> <numEntero>]

<numEntero>::=<digito> {<digito>}

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

<coma> ::= ","

}

BNF

G=(N,T,S,P)

N={<numReal>, <digito>, <coma>}

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

S={<numEntero>}

P={

<numReal>::=<numeroEntero> | <numeroEntero> <coma> <numeroEntero>

<numEntero>::=<digito>|<digito> <numEntero>

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

<coma> ::= ","

}

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

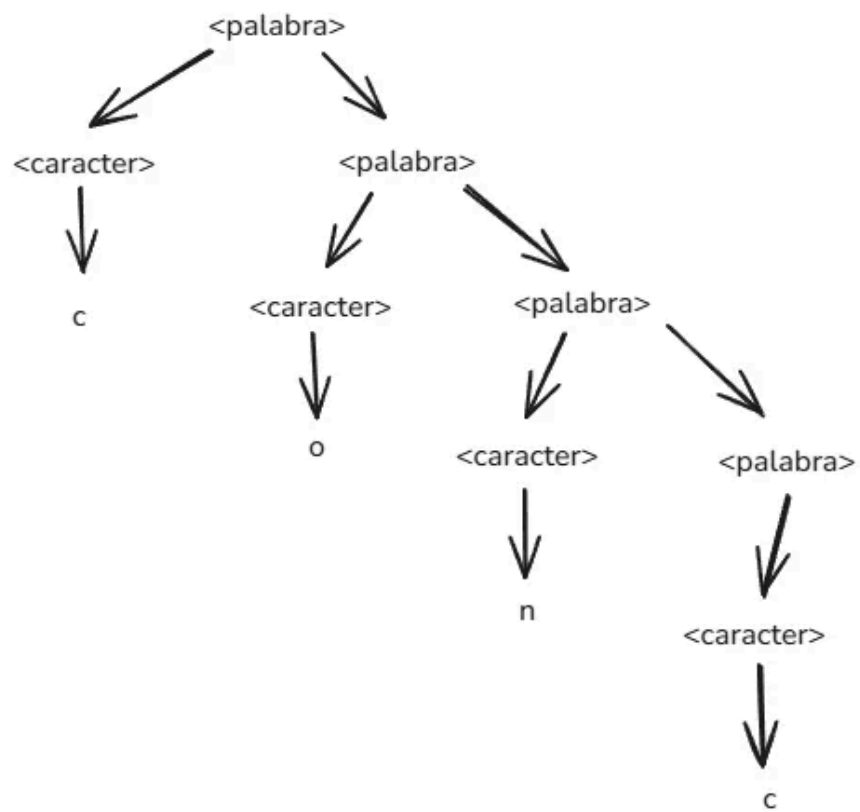
de:

a. Conceptos

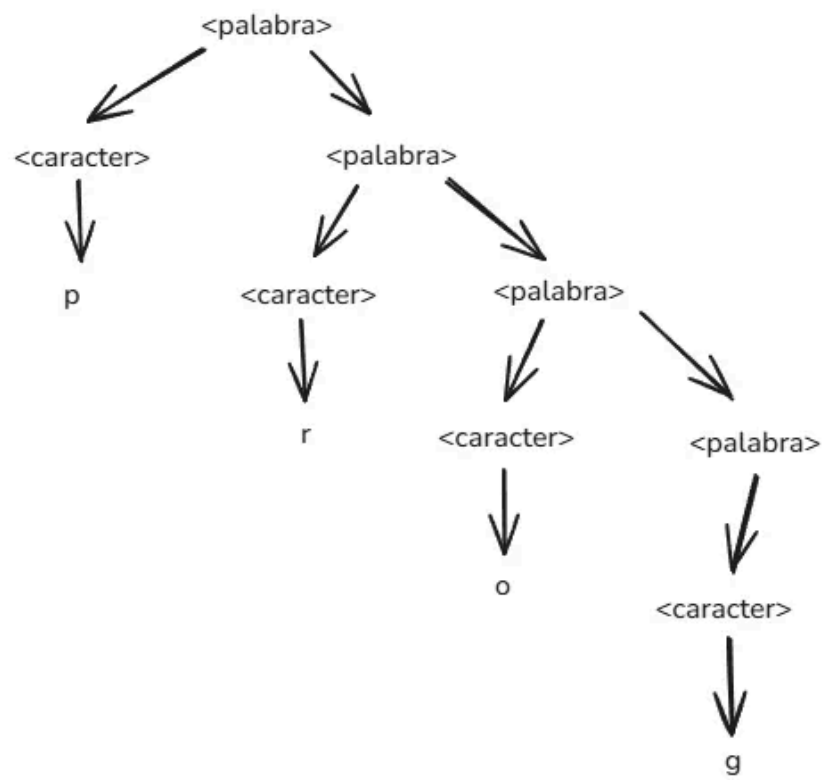
b. Programación

- c. 1255869
- d. 854,26
- e. Conceptos de lenguajes

a_

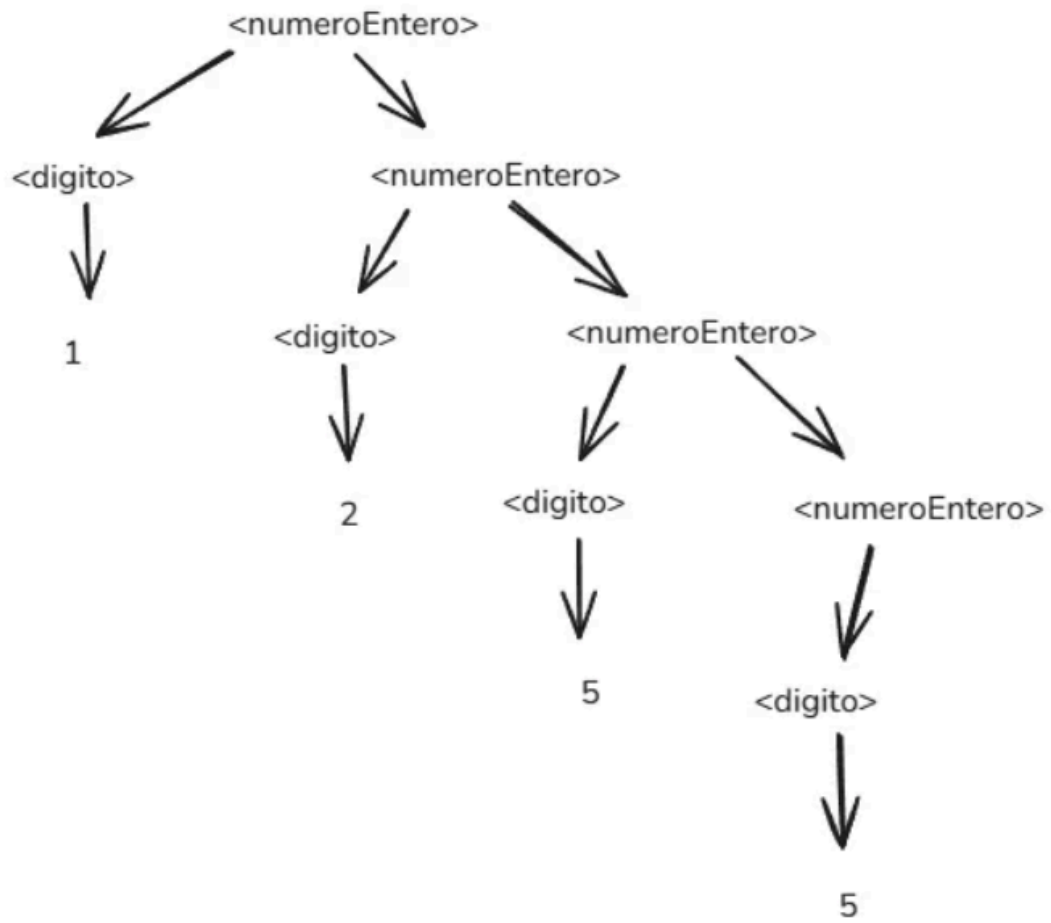


b_

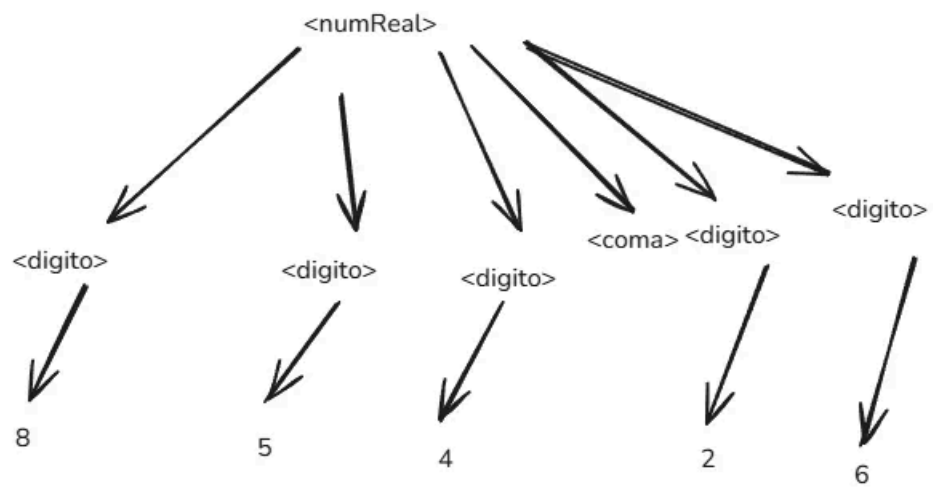


C_

⋮



d_



e_ Con la gramatica planteada solo se pueden representar palabras individuales

Ejercicio 9

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

$N = \{ \langle \text{identificador} \rangle, \langle \text{secuencia} \rangle, \langle \text{letra} \rangle, \langle \text{digito} \rangle \}$

$T = 0 \mid \dots \mid 9 \mid a \mid \dots \mid Z$

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

$P = \{$

$\langle \text{identificador} \rangle ::= \langle \text{letra} \rangle \mid \langle \text{letra} \rangle \langle \text{secuencia} \rangle$

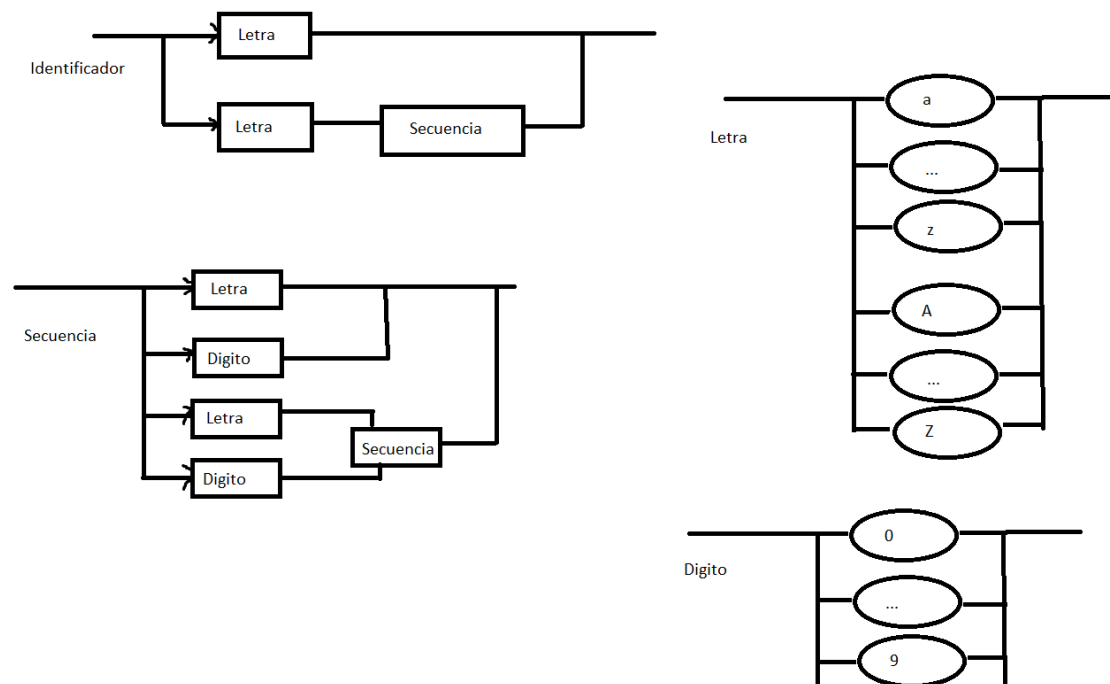
$\langle \text{secuencia} \rangle ::= \langle \text{letra} \rangle \mid \langle \text{digito} \rangle \mid \langle \text{letra} \rangle \langle \text{secuencia} \rangle \mid \langle \text{digito} \rangle$

$\langle \text{secuencia} \rangle$

$\langle \text{letra} \rangle ::= a \mid \dots \mid Z$

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

$\}$



Ejercicio 10

a_

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

$N = (<expresion>, <elemento>, <identificador>, <secuencia>, <letra>, <num>, <digito>)$

$T = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, \dots, z, A, \dots, Z\}$

$S = \{<expresion>\}$

$P = \{ <expresion> ::= <elemento>\{ (+ \mid - \mid * \mid /) <elemento> \}^*$

$<elemento> ::= (<identificador> \mid <num>)$

$<identificador> ::= <letra>\{ (<letra> \mid <digito>) \}^*$

$<letra> ::= (a \mid \dots \mid Z)$

$<num> ::= <digito>\{ <digito> \}^*$

$<digito> ::= (0 \mid \dots \mid 9)$

$\}$

b_

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

$N = (<expresion>, <termino>, <elemento>, <identificador>, <secuencia>, <letra>, <num>, <digito>)$

$T = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, \dots, z, A, \dots, Z\}$

$S = \{<expresion>\}$

$P = \{ <expresion> ::= <termino>\{ (+ \mid -) <termino> \}^*$

$<termino> ::= <elemento> \{ * \mid / \} <elemento> \}^*$

$<elemento> ::= (<identificador> \mid <num>)$

$<identificador> ::= <letra>\{ (<letra> \mid <digito>) \}^*$

$<letra> ::= (a \mid \dots \mid Z)$

$<num> ::= <digito>\{ <digito> \}^*$

$<digito> ::= (0 \mid \dots \mid 9)$

$\}$

c_ Lo que hicimos fue agregar una produccion mas llamada termino, que multiplicaba y dividia dos elementos. Luego usamos este termino en expresion para sumar y restarlos. De esta manera, primero, se hacen las operaciones de multiplicacion y division, y despues las de suma y resta.

11_

- No declara el conjunto de los terminales ni el conjunto distinguido.
- Las producciones de <otro>, <operacion>, <llamada_a_funcion>, <numero>, <sentencia_asignacion>, <sentencia_if>, <sentencia_while>, <sentencia_switch> no las define.
- La forma de definicion de la produccion <sentencia_for> esta mal. No utiliza bien la gramatica EBNF.
- En la produccion <variable>, <bloque> y <sentencia> no utiliza los parentecis para utilizar el operador "|".
- La definicion de la produccion <bloque> se podria simplificar haciendo uso correcto de la gramatica EBNF.
Ej: <bloque> ::= <sentencia> {<sentencia>}*.

CONSULTAR SENTENCIA_FOR

12_

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

$N = \{ \langle \text{tag.div} \rangle \langle \text{atributo} \rangle \langle \text{letra} \rangle \langle \text{digito} \rangle \langle \text{style} \rangle \langle \text{class} \rangle \langle \text{id} \rangle \langle \text{spam} \rangle \langle \text{a} \rangle \}$

$T =$

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

$P = \{$

$\langle \text{tag.div} \rangle ::= ' \langle \text{div} \{ \langle \text{atributo} \rangle \}^* ' >'$

$\{ \langle \text{tags} \rangle \}^*$

$' / \text{div} '$

$\langle \text{tags} \rangle ::= (\langle \text{spam} \rangle \mid \langle \text{a} \rangle \mid \langle \text{class} \rangle \mid \langle \text{id} \rangle \mid \dots)$

$\langle \text{atributo} \rangle ::= (\langle \text{style} \rangle \mid \langle \text{class} \rangle \mid \langle \text{id} \rangle \mid \dots)$

$\langle \text{style} \rangle ::= ' \text{style} ' :'$

$\langle \text{id} \rangle ::= \langle \text{letra} \rangle \{ (\langle \text{letra} \rangle \mid \langle \text{digito} \rangle) \}^*$

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

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

$\}$

13_ No se puede realizar sintacticamente, tiene solo una representacion semantica. Si lo representara sintacticamente no puedo asegurar que sean todos numeros primos.

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)

Usaremos el siguiente lenguaje: Pascal

Codigo ejemplo:

```
program ParOlmpar;  
function sumar(x, i: Integer): Integer;  
begin  
    sumar := x + i;  
end;  
begin  
end;
```

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

$N = \{ \langle \text{function} \rangle \langle \text{type} \rangle \langle \text{body} \rangle \langle \text{id} \rangle \langle \text{result} \rangle \langle \text{letra} \rangle \langle \text{digito} \rangle \}$

$T = \{ \text{'begin'}, \text{'('}, \text{' '}, \text{'.'}, \text{' := '}, \text{' ; '}, \text{'end'} \}$

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

$P = \{$

$\langle \text{function} \rangle ::= \text{'function'} \langle \text{id} \rangle \text{'('} \{ \{ \langle \text{id} \rangle [,] \}^* \text{'.'} \langle \text{type} \rangle [;] \}^* \text{' '}' \text{'.'} \langle \text{type} \rangle \text{' ; '}$

'begin'

$[\langle \text{body} \rangle \text{' ; '}]$

$\langle \text{id} \rangle \text{' := ' } \langle \text{result} \rangle \text{' ; '}$

'end ; '

$\langle \text{id} \rangle ::= \langle \text{letra} \rangle \{ (\langle \text{letra} \rangle \mid \langle \text{digito} \rangle) \}^*$

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

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

$\text{type} ::= (\text{'Integer'} \mid \text{'Real'} \mid \text{'Char'} \mid \text{'Boolean'} \mid \dots)$

```
<body> ::= (<function> | ... | <id>)  
<result>  
}
```