




# Práctica 2 CPLP

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	Indicar una definición formal de una producción de una gramática
	( )	flecha que se divide en dos o más caminos	Se utiliza para la selección de dos alternativas
< p > < p1 >	{ }		Repetición
	*		Repetición de 0 o más veces
	+		Repetición de 1 o más veces
	[]		Se utiliza para indicar que un elemento es opcional

2) La sintaxis de un lenguaje de programación es fundamental ya que establece las reglas y convenciones que deben seguirse para escribir código válido en ese lenguaje. La sintaxis define la estructura y la forma que deben tener las instrucciones y expresiones que se escriben en el código, permitiendo al compilador o intérprete entender correctamente lo que se quiere expresar y generar el programa ejecutable.

Los elementos fundamentales de la sintaxis de un lenguaje de programación son:

1. Tokens o símbolos: son los elementos básicos que conforman el lenguaje, como los operadores, los identificadores, las palabras clave, los caracteres especiales, entre otros.
2. Reglas gramaticales: son las reglas que establecen cómo deben combinarse los tokens para crear expresiones y estructuras válidas en el lenguaje.
3. Estructuras de control: son las formas en que se organizan y controlan las instrucciones en un programa, como las estructuras de selección (if-else, switch), las estructuras de iteración (for, while, do-while), entre otras.

4. Comentarios y uso de blancos: son fragmentos de texto que se incluyen en el código para explicar su funcionamiento o aclarar su significado, pero que son ignorados por el compilador o intérprete.
5. Palabra clave: tienen un significado dentro de un contexto.
6. Palabra reservada: son palabras claves que además no pueden ser usadas por el programados como identificador de otra identidad.
7. Alfabeto o conjunto de caracteres.
8. Identificadores.

La importancia de la sintaxis radica en que permite a los programadores escribir código que sea fácil de entender, mantener y depurar, al mismo tiempo que garantiza que el compilador o intérprete pueda interpretar el código correctamente y generar el programa ejecutable.

***“El código es leído muchas más veces de lo que se escribió”***

3) La regla lexicográfica se refiere a las reglas que definen la estructura de las palabras o símbolos que se utilizan en un lenguaje, incluyendo el conjunto de caracteres permitidos, la forma en que se combinan para formar palabras y la forma en que se clasifican las palabras según su función. En otras palabras, la regla lexicográfica define la gramática de los elementos básicos que componen un lenguaje. Se utiliza para formar las **“word”**.

La regla sintáctica se refiere a las reglas que definen cómo se combinan las palabras y símbolos permitidos en un lenguaje para formar oraciones o expresiones válidas. Estas reglas determinan el orden en que se deben presentar las palabras y símbolos, así como las estructuras gramaticales permitidas en el lenguaje. En resumen, la regla sintáctica define la gramática del lenguaje en términos de la combinación de elementos léxicos.

4) En la definición de un lenguaje, las palabras reservadas son aquellas que tienen un significado especial y predefinido en el lenguaje y no pueden ser utilizadas como identificadores o nombres de variables. Estas palabras son reservadas para ser utilizadas en estructuras de control, declaraciones de variables, funciones, entre otros.

En la definición de una gramática, las palabras reservadas son equivalentes a los símbolos terminales, que son los símbolos que no pueden ser reemplazados por otros símbolos o terminales en la producción de una cadena en el lenguaje generado por la gramática.

Un ejemplo de palabras reservadas en el lenguaje Python son: **if, else, while, for, def, class, import, from, and, or, not, True, False, None**, entre otras.

5) (a) Componentes de la gramática:

- N: conjunto de no terminales = {<numero\_entero>, <digito>}
- T: conjunto de terminales = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
- S: símbolo inicial = <numero\_entero>
- P: conjunto de producciones = { <numero\_entero> ::= <digito><numero\_entero> | <numero\_entero><digito> | <digito> <digito> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 }

(b) La gramática es ambigua porque la producción <numero\_entero> ::= <digito><numero\_entero> | <numero\_entero><digito> puede generar la misma cadena de varias formas distintas. Por ejemplo, la cadena "123" puede ser generada por <numero\_entero> ::= <digito><numero\_entero> ::= 1<digito><numero\_entero> ::= 1 2<digito><numero\_entero> ::= 1 2 3<digito>. Pero también puede ser generada por <numero\_entero> ::= <numero\_entero><digito> ::= <digito><digito><digito> ::= 1 2 3.

Para corregir la ambigüedad se puede modificar la producción <numero\_entero> ::= <digito><numero\_entero> | <numero\_entero><digito> para que indique un orden específico en la concatenación de dígitos. Por ejemplo, se puede cambiar a <numero\_entero> ::= <digito> | <numero\_entero><digito>, lo que garantiza que siempre se agregue un dígito al final de la cadena. De esta manera, la gramática queda sin ambigüedad:

N = {<numero\_entero>, <digito> } T = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9} S = <numero\_entero> P = { <numero\_entero>::=<digito> | <numero\_entero><digito> <digito> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 }

6) Palabras

G= (N,T,S,P)

N = {<letra>, <palabra>}

T= {a,b,c...,z,A,B,C...Z}

S= {<palabra>}

P= {<palabra>::= <letra> <palabra>| <letra>

<letra>:= a|b|c|...|z|A|B|C|...|Z

}

7) EBNF Numeros reales

G= (N,T,S,P)

N= {<numero\_real>, <digito>}

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

$S = \{ \langle \text{numero\_real} \rangle \}$

$P = \{$   
     $\langle \text{numero\_real} \rangle ::= [(+|-)] \{ \langle \text{digito} \rangle \}^+ [ , \{ \langle \text{digito} \rangle \}^+ ]$   
     $\langle \text{digito} \rangle ::= 0|1|2|3|4|5|6|7|8|9$   
 $\}$

BNF

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

$N = \{ \langle \text{numero\_real} \rangle, \langle \text{decimal} \rangle, \langle \text{digito} \rangle, \langle \text{numero\_entero} \rangle, \langle \text{numero\_entero\_sig} \rangle \}$

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

$S = \langle \text{numero\_real} \rangle$

$P = \{$   
     $\langle \text{numero\_real} \rangle ::= \langle \text{numero\_entero\_sig} \rangle | \langle \text{numero\_entero} \rangle \langle \text{decimal} \rangle$   
     $\langle \text{numero\_entero\_sig} \rangle ::= \langle \text{digito} \rangle | \langle \text{digito} \rangle \langle \text{numero\_entero} \rangle$   
     $\langle \text{decimal} \rangle ::= \langle \text{numero\_entero} \rangle$   
     $\langle \text{digito} \rangle ::= 0|1|2|3|4|5|6|7|8|9$   
 $\}$

La diferencia es que en BNF se requiere definir más no terminales y en la producción se tienen que definir cada uno de estos.

8)

**Conceptos**

/      \

C        onceptos  
 /        \  
 o        nceptos  
 /        \  
       n        ceptos  
       /        \  
       c        eptos  
       /        \  
       e        ptos  
       /        \  
       p        tos  
 /        \  
 t        os  
       /        \  
       o        s  
 /  
 s

# **Programacion <palabra>**

/        \  
 P<letra>        rogramacion<palabra>  
 /        \  
       r<letra>        ogramacion<palabra>  
 /        \  
       o<letra>        gramacion<palabra>  
 /        \  
       g<letra>        ramacion<palabra>  
 /        \  
 r<letra>        amacion<palabra>

/        \  
a<letra>    macion<palabra>

/        \  
m<letra>    acion<palabra>

/        \  
a<letra>    cion<palabra>

/        \  
c<letra>    ion<palabra>

/        \  
i<letra>    on<palabra>

/        \  
o<letra>    n<palabra>

/  
n<letra>

**1255869**

<numero\_real>

|

<numero\_entero\_sig>

|

<numero\_entero> 1255869

/        \  
1 <digito>        <numero\_entero>255869

/        \  
2<digito>    <numero\_entero>55869

/        \  
5 <digito>    <numero\_entero>5869

/        \  
5<digito>    <numero\_entero>869

/        \  
8<digito>    <numero\_entero>69

```

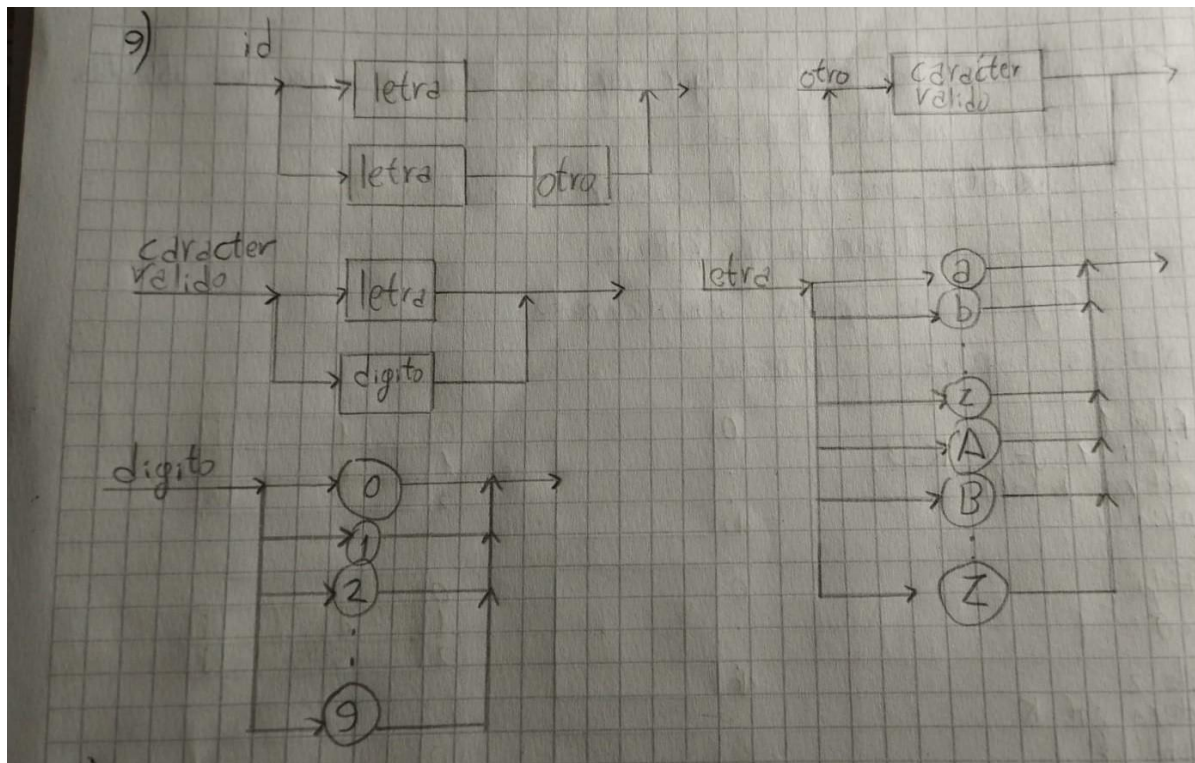
/      \
6<digito>  <numero_entero>9
/
9 <digito>
854,26
      <numero_real> 854,26
/      \
<numero_entero_sig>      <decimal>
|      |
<numero_entero>854      <numero_entero> 26
/      \      /      \
<digito>8  <numero_entero>54  <digito>2  <numero_entero>6
/      \      /
<digito>5  <numero_entero>4  <digito>6
/
<digito>4

```

### Conceptos de lenguajes:

Con la gramática definida no se puede representar un String, solo palabras o números reales.

9)



10) a.  $G = (N, T, S, P)$

$N = \{ \langle \text{expresión} \rangle, \langle \text{dígito} \rangle, \langle \text{numero} \rangle, \langle \text{variable} \rangle, \langle \text{operador} \rangle, \langle \text{letra} \rangle \}$

$T = \{ 0, 1, \dots, 9, a, b, \dots, z, A, B, \dots, Z, "+", "-", "*", "/" \}$

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

$P = \{ \langle \text{expresión} \rangle ::= (\langle \text{variable} \rangle | \langle \text{numero} \rangle) \{ \langle \text{operador} \rangle \} (\langle \text{variable} \rangle | \langle \text{numero} \rangle) \}$

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

$\langle \text{operador} \rangle ::= (+ | - | * | /)$

$\langle \text{numero} \rangle ::= \{ + | - \} \langle \text{dígito} \rangle \{ \langle \text{dígito} \rangle \}^*$

$\langle \text{variable} \rangle ::= (\langle \text{letra} \rangle | \langle \text{dígito} \rangle) \{ \langle \text{letra} \rangle | \langle \text{dígito} \rangle \}^*$

$\langle \text{letra} \rangle ::= a | b | \dots | z | A | B | \dots | Z$

}

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



$N = \{ \langle \text{expresion} \rangle, \langle \text{expConPriori} \rangle, \langle \text{digito} \rangle, \langle \text{numero} \rangle, \langle \text{variable} \rangle, \langle \text{sumRes} \rangle, \langle \text{divMul} \rangle, \langle \text{letra} \rangle \}$

$T = \{ 0, 1, \dots, 9, a, b, \dots, z, A, B, \dots, Z, "+", "-", "*", "/" \}$

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

$P = \{ \langle \text{expresion} \rangle ::= (\langle \text{expConPriori} \rangle) \{ \langle \text{sumRes} \rangle \langle \text{expConPriori} \rangle^* \}$

$\langle \text{expConPriori} \rangle ::= [+|-](\langle \text{numero} \rangle | \langle \text{variable} \rangle) \{ \langle \text{divMul} \rangle \{ \langle \text{numero} \rangle | \langle \text{variable} \rangle \}^* \}$

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

$\langle \text{sumRes} \rangle ::= (+|-)$

$\langle \text{divMul} \rangle ::= (*|/)$

$\langle \text{numero} \rangle ::= \{ +|- \} \langle \text{digito} \rangle \{ \langle \text{digito} \rangle \}^*$

$\langle \text{variable} \rangle ::= (\langle \text{letra} \rangle | \langle \text{digito} \rangle) \{ \langle \text{letra} \rangle | \langle \text{digito} \rangle \}^*$

$\langle \text{letra} \rangle ::= a | b | \dots | z | A | B | \dots | Z$

$\}$

c. Separar en términos según el operador y resolver los que tengan mayor prioridad, en este caso \* y / y luego +, -.

11) No están definidas la T y la S, es decir, faltan el conjunto de los terminales que se utilizan y el símbolo distinguido.

A su vez, faltan agregar en N,  $\langle \text{sentencia\_if} \rangle$ ,  $\langle \text{sentencia\_while} \rangle$ ,  $\langle \text{sentencia\_switch} \rangle$

En caso de que las sentencias que se utilizan no estén definidas aparte (sub-gramática), habría que realizar la producción de ellas acá (tales como  $\langle \text{otro} \rangle$ ,

$\langle \text{sentencia\_asignacion} \rangle$ ,  $\langle \text{llamada\_a\_funcion} \rangle$ ,  $\langle \text{sentencia\_if} \rangle$ ,  $\langle \text{sentencia\_for} \rangle$ ,

$\langle \text{sentencia\_while} \rangle$ ,  $\langle \text{sentencia\_switch} \rangle$ ,  $\langle \text{operacion} \rangle$ ,  $\langle \text{numero} \rangle$ )

$\langle \text{bloque} \rangle ::= \langle \text{sentencia} \rangle | \langle \text{sentencia} \rangle \langle \text{bloque} \rangle | \langle \text{bloque} \rangle \langle \text{sentencia} \rangle$ ; es ambiguo

12)  $G = \{ N, T, S, P \}$

N= {<tag\_div>, <div\_abertura>, <bloque>, <div\_cierre>, <atributos>, <palabra>, <cadena>, <letra>, <numero>, <carácter\_especial>, <tag\_span>}

T= {"A..Z", "a..z", "+...:", "<", ">", "/"}

S= {<tag\_div>}

P= { <tag\_div> ::= <div\_abertura>[<bloque>]<div\_cierre>

<div\_abertura> ::= <div{<atributo>}\*>

<div\_cierre> ::= /

<atributo> ::= <palabra> = "<cadena>"

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

<cadena> ::= {( <letra> | <numero> | <carácter\_especial> )}+

<letra> ::= (A|B|..|Z|a|b|..|z)

<numero> ::= (0|1|..|9)

<carácter\_especial> ::= (+|-|/|\*|...| |;|)

<bloque> ::= [( <cadena> | <tag\_div> | <tag\_span> | ... )]

<tag\_span> ::= <span{<atributo>}\*>[<bloque>]</span>

}

- 13) La definición de la gramática para los números primos es igual a la de los números enteros, solamente habría que agregar la semántica para verificar si es un número primo o no.

NOTAS: Cuando se trata de una sentencia no es necesario definirlas (sentencia\_if | sentencia\_while | ...)

Cuando se tiene que suponer que por ej un return está definido como una subgramática se especifica.