Conceptos y paradigmas de lenguajes de programación – Practica 2

1) ¿Cuál es la importancia de las sintaxis para un lenguaje? ¿Por qué?  
Sí. La sintaxis es muy importante ya que es el conjunto de reglas que definen como componer símbolos (letras, dígitos y otros caracteres) para formar los programas. Es, además, la disposición de palabras como elementos en una oración para mostrar su relación, describe la serie de símbolos que constituyen programas válidos.

2) ¿A qué se denomina regla lexicográfica y regla sintáctica?

Las reglas *lexicográficas* nos conducen a saber cómo se pueden formar las palabras del lenguaje a partir del alfabeto. Por ejemplo:

papa = PAPA en PASCAL.

papa <> PAPA en Ada o C.

Las reglas *lexicográficas* son un conjunto de reglas para combinar caracteres.

Las reglas *sintácticas* nos conducen a saber como se definen las sentencias y las expresiones.

La fase fundamental de cualquier traducción es agrupar esta serie de caracteres en sus constituyentes elementales: identificadores, delimitadores, símbolos de operadores, números, palabras clave, palabras pregonadas, espacios en blanco, comentarios, etc. Esta fase se conoce como “análisis léxico”, y las unidades básicas de programa que resultan del análisis léxico se llaman “elementos léxicos”

Para escribir una sintaxis, se comienza por un “lenguaje natural” y se finaliza con “métodos formales”.

La segunda etapa de la traducción es el “análisis sintáctico” (parsing:= para cada sentencia construye un árbol sintáctico o de derivación). En ella se identifican las estructuras de programa más grandes (enunciados, declaraciones, expresiones, etc.) usando los elementos léxicos producidos por el analizador léxico.

3) ¿Porqué cada lenguaje necesita definir su gramática? ¿Cómo se define una gramática?

Antes que nada aclaremos que una gramática se compone de un conjunto de reglas (llamadas producciones) que especifican las series de caracteres (o elementos léxicos) que forman programas permisibles en el lenguaje que se está definiendo. Una gramática formal es simplemente una gramática que se especifica usando una notación definida de manera estricta.

La gramática se puede definir como una cuádrupla formada por

G = (T,N,S,P)

T: Conjunto de símbolos terminales.

N: Conjunto de símbolos no terminales.

S: Símbolo distinguidos (“programa”)

P: Conjunto de reglas o producciones.

Para poder definir la gramática de un lenguaje, necesitamos de un lenguaje más general, para definir dicha gramática. Esto es lo que se denomina “Metalenguaje”.

Un metalenguaje se utiliza para poder definir las gramáticas; los más conocidos son BNF y EBNF.

4) Defina en BNF (Gramática de contexto libre desarrollada por Backus- Naur) la gramática para la definición de una palabra cualquiera. Por ejemplo la palabra “aula”

Definicion de la gramatica:

G = {N ={<aula>,<nombreAula>,<palabra>,<letra>,<numeroBancos>,<num>,<numero>}

T = {‘a’,’b’,…,’z’,’A’,’B’,…,’Z’,0,1,2,…,9}

S = {<aula>}

P = {

<aula> ::= <nombreAula><numeroBancos>

<nombreAula> ::= <palabra><numero>

<palabra> ::= <letra> | <letra><palabra>

<letra> ::= ‘A’ | ‘B’ | … ‘Z’

<numeroBancos> ::= <num> | <num><numeroBancos>

<num> ::= 0 | 1 | 2 | … 9

<numero> ::= <num>|<num><numero>

}}

5) ¿Qué inconveniente se presenta cuando una gramática es ambigua?

La ambigüedad es algo que tiene que ver con la sintaxis.

Una gramática se considera ambigua cuando existe en ella alguna cadena en el lenguaje que crea, que tiene dos árboles de análisis sintáctico distintos.

Si toda gramática para un lenguaje dado es ambigua, se dice que el lenguaje es inherentemente ambiguo.

6) La gramática en el ejercicio 4 no es ambigua porque está definido un único camino de evaluación de la palabra. Si la definición de palabra fuera (<letra>|<letra><palabra>|<palabra><letra>) si seria ambiguo (2 caminos).

7)EBNF para la definición de numero reales:

G = {N={<real>,<digito>,<coma>,<signo>,<decimal>}

T={0,1,…,9,”+”,”-“}

S={<real>}

P={<real> ::= [<signo>]{<digito>}+ [<decimal>];

<signo> ::= (+ | -);

<digito> ::= 0 | 1 | 2 | ... | 9;

<decimal> ::= <coma>{<digito>}+;

<coma> ::= ‘,’}} (forma de vale)

P={<real> ::= [<signo>]{<digito>}+ [, {<digito>}+];

<signo> ::= (+ | -);

<digito> ::= 0 | 1 | 2 | ... | 9;}

8) Conceptos:

<palabra> := conceptos

<carácter> := c <palabra> := onceptos

<carácter> := o <palabra> := nceptos

<carácter> := n <palabra> ceptos

<carácter> := c <palabra> eptos

<carácter> := e <palabra>:=ptos

<carácter> := p <palabra> := tos

<carácter> := t <palabra>:=os

<carácter> := o <palabra> := s

9) Defina con BNF la gramática para una expresión numérica, dónde

intervienen variables y números. Considerar los operadores +, -, \* y / sin orden de prioridad. No considerar el uso de paréntesis.

G={N={<expresion>, <variable>, <numero>, <operador>}

T={0..9,’a’..’z’,’A’..’Z’,’+’,’-‘,’\*’,’/’}

S={<expresión>}

P= {<expresión> ::= <operando><operador><expresion>|<operando>

<operando> ::= <variable>|<numero>

<variable> ::= ’a’|..|’z’|’A’|..|’Z’

<numero>::= 0..9

<operador>::= ’+’,’-‘,’\*’,’/’

}

10) A la gramática definida en el ejercicio 9 agregarle prioridad de operadores.

G= {N= {<expresion>, <variable>, <numero>, <termino>, <op1>, <op2>, <elem>, <digito>, <letra>, <sec>}

T= {0..9, ’A’..’Z’ , ’+’,’-‘,’\*’,’/’}

S={<expresión>}

P= {<expresión> ::= <termino>| <termino> <op1> <expresión>

<termino>::= <elem>|<elem><op2><termino>

<elem>::= <numero>|<variable>

<numero>::= <digito>|<digito><numero>

<digito>::= 0|..|9

<variable>::= <letra>|<letra><sec>

<letra>::= ‘A’|..|’Z’

<sec>::= <letra>|<numero>|<letra><sec>|<num><sec>

<op1>::= +|-

<op2>::= \*|/}}

11) A la gramática definida en el ejercicio 10 agregarle el uso de paréntesis.

G= {N= {<expresion>, <variable>, <numero>, <termino>, <op1>, <op2>, <elem>, <digito>, <letra>, <sec>}

T= {0..9, ’A’..’Z’, ’+’,’-‘,’\*’,’/’ }

S={<expresión>}

P= {<expresión> ::= <termino>| <termino> <op1> <expresión>

<termino>::= <elem>|<elem><op2><termino>| (<expresión>)

<elem>::= <numero>|<variable>

<numero>::= <digito>|<digito><numero>

<digito>::= 0|..|9

<variable>::= <letra>|<letra><sec>

<letra>::= ‘A’|..|’Z’

<sec>::= <letra>|<numero>|<letra><sec>|<num><sec>

<op1>::= +|-

<op2>::= \*|/}}

a + b \* (c - a)

termino

op1

elem

op2

expresion=term → b \* (c - a)

expresion=(exp)→(c – a)

12) Defina la gramática para la sentencia for de C, utilizando EBNF.

G={N={<sentenciafor>, <sentencia>, <exp1>, <exp2>, <exp3>, <id>, <num>, <letra>, <car>}

T= {a..z, A..Z, 0..9, @, -, \_, #, for, <, (, ), {, }, +, ;}

S= {<sentenciafor>}

P= {<sentenciafor>::= for ’(‘<exp1>; <exp2>; <exp3> ‘)’ ‘{‘ { <sentencia> ;}+ ‘}’

<exp1>::= <id>=<num>

<id>::= <letra> {(<letra>|<num>|<car>)}\* → asi obliga a que si o si empieze con una letra el id

<letra>::=a|..|z| A|..|Z

<num>::= 0|..|9

<car>::= \_|-|@|#|..

<exp2>::=<id> ’<’ <num>

<exp3>::= <id>++ }

}

13) Defina la gramática para la definición de clases en Java.

G={N={<clase>,<parametro>,<nombreClase>,<cuerpo>,<letraMay>,<letra>,<atributo>,<metodos>..}

T={class, public, private, protected, package, void, A..Z, 0..9, int, char, string..}

S={<clase>}

P={<clase> ::= Class <nombreClase> '{'<cuerpo>'}' extends

<parametro> ::= { <tipo> <nom> { , <nom> }\* }+

<nomreClase> ::= <letraMay> {<letraMay>|<letra>|<num>|<car>}\*

<letraMay> ::= A|B|..|Z

<sec> ::= <letra> {<letra>|<car>|<num>}\*

<letra> ::= a|b|..|z

<car> ::= @|\_|- …

<num> ::= 0|1|..|9

<cuerpo> ::= {<atributo>}\*{<metodo>}\*

<atributo> ::= <acceso><tipo><sec>

<acceso> ::= protected|private|public|package

<tipo> ::= int|Integer|string..

<metodo> ::= <acceso><tipoM><sec>{<parametro>}+'{'<cuerpo>'}'

<tipoM> ::= <tipo>| void}}

14)

a) Realice la gramática en EBNF para la declaración de variables en Delphi

G={N={<variable>,<palClave>,<nom>,<tipo>,<letra>,<sec>,<num>,<car>}

T={var, :, ;, A, ..,Z, a,..,z,0,..,9, integer, real, char, string...}

S={<variable>}

P={<variable>::=<palClave>{<nom>{,<nom>}\* : <tipo>;}+

<palClave>::=var

<nom>::=<letra>{(<nom>|<num>|<car>)}\*

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

<car>::=@|-|\_|...

<num>::=0|..|9

<tipo>::=integer|char|string|real..}

}

b) Realice lo mismo con diagramas sintácticos.

15)  
a) Defina con BNF una gramática para un procedure en ADA. Luego defínalo

mostrando las diferencias con EBNF.

b) Realice lo mismo con diagramas sintácticos.

16)  
 La siguiente gramática escrita en BNF intenta definir la sentencia de

asignación en

Pascal:

P:{

<asignación>::=<variable>:=<expresión>,

<expresión>::= <identificador>;

<identificador>::= <variable>|<numero>;

<variable>::= a | b | c

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

}

Indicar si es correcta y completa. En caso de no serlo corregirla