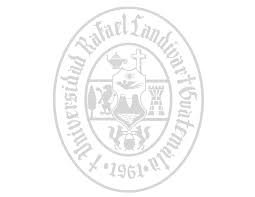
**FACULTAD DE INGENIERIA**

**CATEDRATICO: Moises Alonso**

**Proyecto Scanner**

**Catherine Rosario López Vicente**

**Carne: 1055816**

Marco Teórico

|  |  |
| --- | --- |
| Al lenguaje generado por medio de [una gramática regular.](http://decsai.ugr.es/~rosa/tutormc/teoria/GRAMATICAS%20REGULARES.htm#GRAMATICA_REGULAR) Son aquellos lenguajes cuyas cadenas está formadas por la concatenación de símbolos, en las cuales no hay relación entre una parte de la cadena y otra parte de la cadena.  Vemos que una gramática regular o de tipo 3 es aquella gramática donde las reglas de producción siguen la siguiente estructura: A→ uB  o A→ u donde u∈ T\* y A,B∈ V  Programa para calcular primero y seguir conjuntos de gramática dada  Antes de continuar, se recomienda estar familiarizado con los conceptos básicos del análisis de sintaxis, el análisis LL (1) y las reglas para calcular los conjuntos de una gramática Primero y Seguir.  as funciones follow y followfirst están involucradas en el cálculo del Follow Set de un Non-Terminal dado. El siguiente conjunto del símbolo de inicio siempre contendrá "$". Ahora el cálculo de Seguir se divide en tres casos generales:  Si un No-Terminal en el RHS de cualquier producción es seguido inmediatamente por un Terminal, entonces puede incluirse inmediatamente en el conjunto Seguir de ese No-Terminal.  Si un no terminal en el RHS de cualquier producción es seguido inmediatamente por un no terminal, entonces el primer conjunto de ese nuevo no terminal se incluye en el siguiente conjunto de nuestro no terminal original. En caso de encontrar un épsilon, es decir, "#", pase al siguiente símbolo en la producción. Nota: "#" nunca se incluye en el conjunto Seguir de cualquier no terminal.  Si se alcanza el final de una producción mientras se calcula el seguimiento, el conjunto Seguir de ese no terminal incluirá el conjunto Seguir del No terminal en el LHS de esa producción. Esto puede implementarse fácilmente por recursividad.  Suposiciones  Epsilon está representado por '#'.  Las producciones son de la forma A = B, donde 'A' es un solo no terminal y 'B' puede ser cualquier combinación de terminales y no terminales.  LHS de la primera regla de producción es el símbolo de inicio.  Grammer no se deja recursivo.  Cada producción de un terminal no se ingresa en una línea diferente.  Solo las letras mayúsculas son no terminales y todo lo demás es un terminal.  No utilice '!' o '$' ya que están reservados para fines especiales.  Explicación: Almacene la gramática en una matriz de caracteres 2D production. findfirstLa función es para calcular el primero de cualquier no terminal. Cálculo de firstcaídas en dos casos generales:  Si el primer símbolo en el RHS de la producción es un Terminal, entonces puede incluirse directamente en el primer conjunto.  Si el primer símbolo en el RHS de la producción es un No terminal, vuelva a llamar a la función findfirst en ese No terminal. Manejar estos casos como Recursión es la mejor solución posible. Aquí nuevamente, si el Primero del nuevo No Terminal contiene un épsilon, entonces tenemos que pasar al siguiente símbolo de la producción original, que nuevamente puede ser un Terminal o un No Terminal.  Diagramas | |
|  |  | |
|  |  | |
|  |  | |
|  |  | |
|  |  | |
|  |  | |
|  |  | |
|  |  | |

Gramatica Base

Inicio = SETS ;

SETS = "SETS" { ListSet } ;

ListSet = Id "=" Let

| Id "=" Dig

| Id "=" Chart

;

TOKENS = "TOKENS" { ListToken } ;

ListToken = IdT "=" TOKEN;

TOKEN = Id Id "\*"

| "'" Caracteres "'"

| "'" Letra "'"

| "'" Caracteres "'" "'" Caracteres "'"

| "'" Letra "'" "'" Letra "'"

| "'" """" Id """" "'"| "'" "'" "'" Id "'" "'" "'"

| Id "(" Id "|" Id ")" "\*" "{" "RESERVADAS" "(" ")" "}"

;

ACTIONS = "ACTIONS" "RESERVADAS" "(" ")" "{" ListAction "}" ;

ListAction = IdAct "=" "'" IdRe "'" ;

ERRORS = Prefijo "ERROR" "=" {IdERROR} ;

IdERROR = Numero

| Numero Numero

;

Prefijo = Letra

| Letra Letra

;

IdRe = Letra

| Letra Letra

;

IdAct = Numero

| Numero Numero

;

Chart = "CHR" "(" Numero Numero")"

| "CHR" "(" Numero Numero Numero ")"

;

Let = "'" Letra "'" ".." "'" Letra "'" "+" "'" Letra "'" ".." "'" Letra "'"

| Let "+" "'" "\_" "'"

;

Dig = "'" Numero "'" ".." "'" Numero "'" ;

Id = "LETRA"

| "DIGITO"

| "CHARSET"

;

IdT = "TOKEN" Numero

| "TOKEN" Numero Numero

;

Numero = "0"| "1"| "2"| "3"| "4"| "5"| "6"| "7"| "8"| "9" ;

Letra = "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"| "o"| "p"| "q"| "r"| "s"| "t" | "u"| "v"| "w"| "x"| "y"| "z" ;

Caracteres = "="| "<"| ">"| "+"| "\_"| "\*"

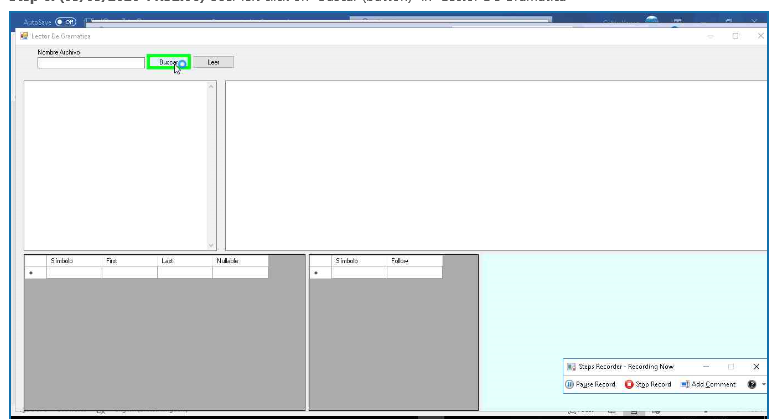
| "("| ")"| ";"| ":"| "."| ","| "|"

| "{"| "}"| "["| "]"

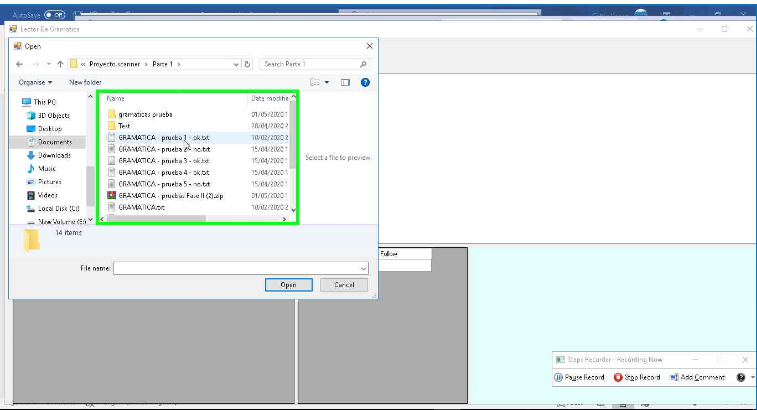
;

Manual de Usuario

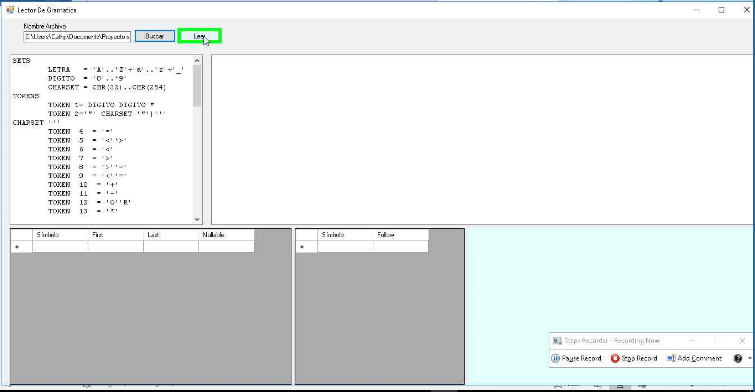
Se presiona buscar



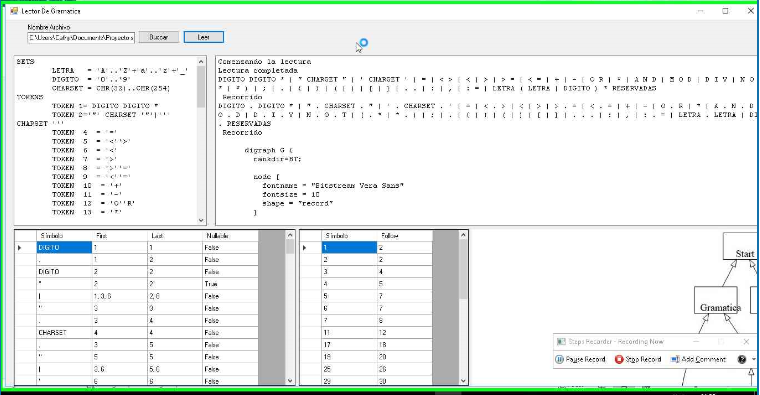
Se tiene Varios documentos de prueba



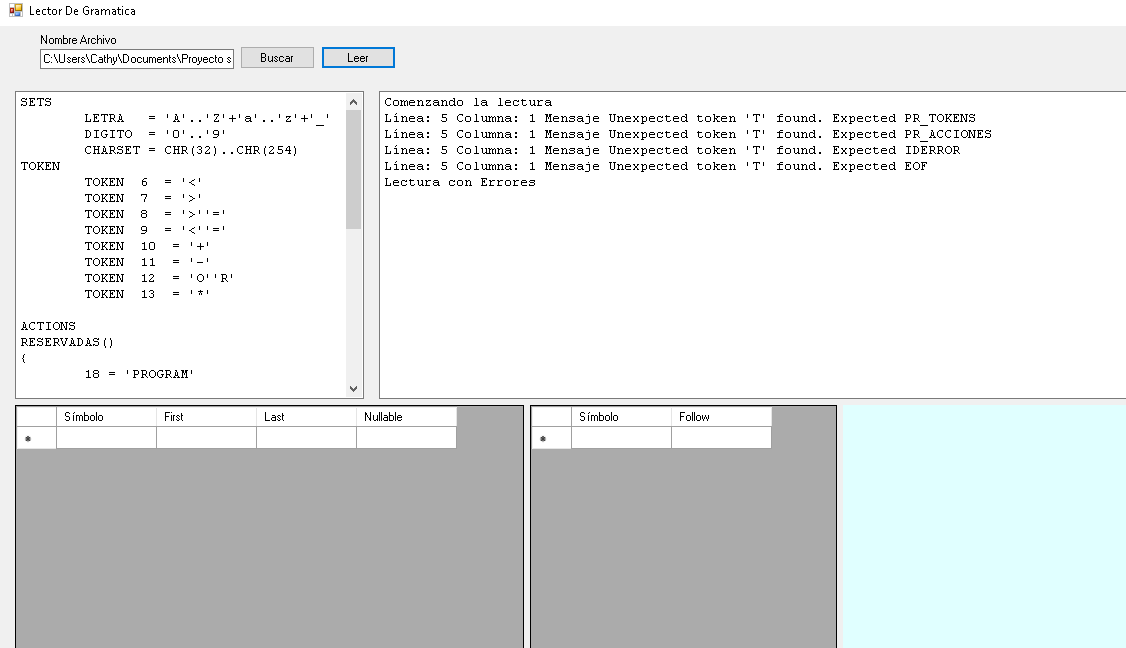
Empieza la Prueba presionando “Leer”



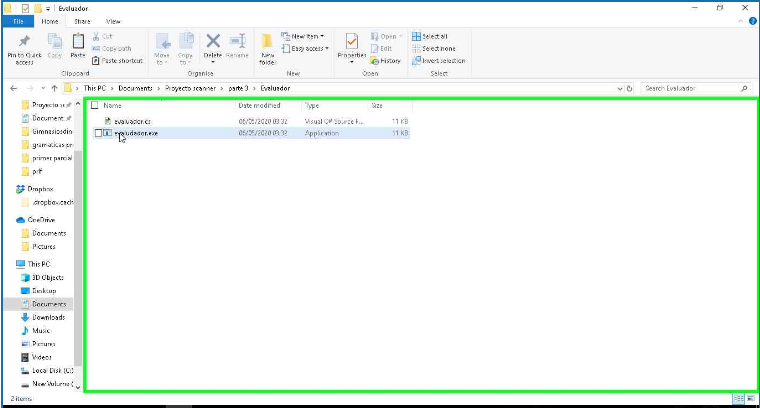
Se muestra la lectura del documento y con eso las tablas pedidas y el arbol

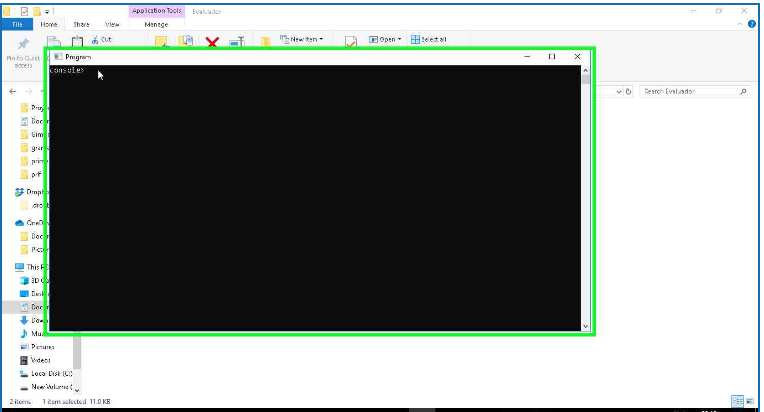


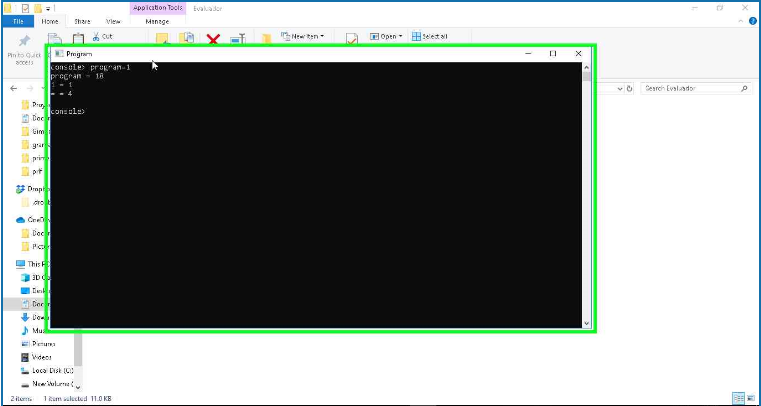
Intentamos con un documento que no este correcto y muestra los errores



Fase 3







Al no tener la información correcta no envia nada de vuelta

