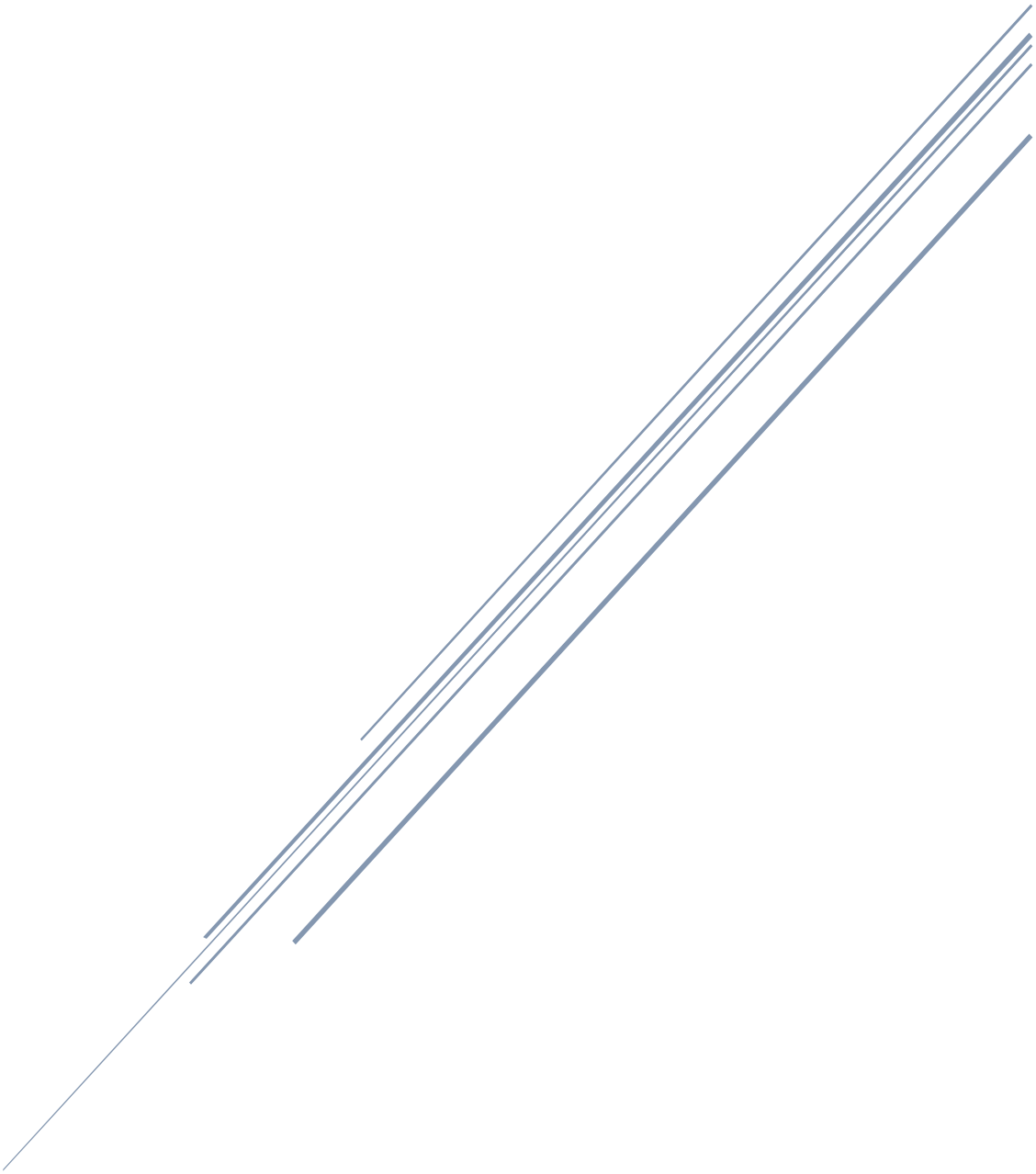
GRAMATICA TYPESTY

Explicación de la gramática.



Universidad San Carlos de Guatemala
Ariel Rubelce Macario Coronado

Gramática.

Sea la gramática G = Terminales, No terminales, P, donde P son las Producciones, a continuación, se definen los terminales y No terminales.

Terminales.

SIMBOLO	NOMBRE
"=="	'IGUALIGUAL'
"++"	'MASMAS';
"--"	'MENOSMENOS';
">="	'MAYORIGUAL';
"<="	'MENORIGUAL';
"!="	'DIFERENCIA';
"<"	'MENOR';
","	'COMA';
">"	'MAYOR'
"LIST"	'LIST';
"ADD"	'ADD';
"."	'PUNTO';
"+"	'MAS';
"*"	'MULTI';
"/"	'DIV';
"-"	'MENOS';
"="	'IGUAL';
"^"	'EXPONENTE';
"%"	'MODULO';
"?"	'INTERROGACION';
":"	'DOSPTS';
"!"	'NOT';
"||"	'OR';
"&&"	'AND';
"("	'PARA';
")"	'PARC';
"["	'CORA';
"]"	'CORC';
";"	'PTCOMA';
"{"	'LLAVEA';
"}"	'LLAVEC';
"\\N"	"SALTOLN";
"\\\\"	"BARRAINV";
"\\\""	"COMILLADOBLE";

Los Símbolos anteriormente mostrados son aquellos que el lenguaje usara para dar realizar un análisis léxico, son lenguajes propios del lenguaje, estos símbolos son conocidos con el alfabeto.

Palabra Reservada	Nombre
"if"	'if';
"else"	'else';
"switch"	'switch';
"case"	'case';
"Default"	'default';
"break"	'break';
"continue"	'continue';
"return"	'return';
"while"	'while';
"do"	'do';
"for	'for';
"void"	'void';
"print"	'print';
"tolower	'tolower';
"toupper	'toupper';
"length"	'length';
"truncate"	'truncate';
"round	'round';
"TypeOf"	'typeof';
"tostring"	'tostring';
"tochararray"	'tochararray';
"exec"	'exec';
"int"	'int';
"boolean"	'boolean';
"double"	'double';
"char"	'char';
"string"	'string';
"true"	'true';
"false"	'false';
"new"	'new';

Tambien debemos de definer las palabras reservadas para el lenguajes ya que con este se realiza el análisis léxico.

Expresiones regulares.

Expresión	Nombre
`[/][*][^*]*[*]+([^/*][^*]*[*]+)*[/`	Comentario Simple
`"//".*`	Comentario multilínea.
`[0-9]+("."[0-9]+)?\b`	números
`([a-zA-Z])([a-zA-Z0-9_])*`	identificador
`["\""]([^\""])*["\""]`	Cadena
`\'[^\"]?\'`	Carácter.

Las expresiones regulares nos sirven para describir patrones, como por ejemplo reconocer números, cadenas de textos y esto nos facilita el reconocer con el lenguaje.

No Terminales.

Los No terminales son aquellos que nos inician una nueva producción que es lo que debe de seguir, se usan terminales y no terminales.

- INICIO
- OPCIONESCUERPO
- CUERPO
- OPCIONESMETODS
- CUERPOMETODO
- BREAK
- DEC_VAR
- RETURN
- INICIALIZACION
- AUM
- CASTEO
- TIPO
- EXP
- DECLAVECT
- LISTVALORES
- LISTAPARAMETROS
- PARAMETROS
- ACCESS
- MODDIFIC
- DECLALIST
- ADDLIST
- IFS
- ELSEIFS
- CONELSEIF
- SWITCHS
- CASES
- CONCASE
- ELSEIF
- WHILES
- FORS
- INIFOR
- DOWHILE
- LOGICO
- FUNCIONES
- METODOS
- LLAMADAS
- CALLS
- PRINT
- TOLOWER
- TOUPPER
- LENGTH
- TRUNCATE
- ROUND
- TYPEOF
- TOSTRING
- TOCHARARRAY
- EXEC

PRODUCCIONES.

INICIO DE LAS PRODUCCIONES.

INICIO: OPCIONESCUERPO EOF

| error ptycoma

Esta producción inicia la todas las producciones, ya que todos los no terminales se desencadenan a partir de esta producción.

OPCIONESCUERPO: OPCIONESCUERPO CUERPO

| CUERPO

Esta producción hace que no nos pueden venir distintos y muchas producciones ya que aplicamos una recursividad por la izquierda.

CUERPO: DEC_VAR

| INICIALIZACION ptcoma CUERPO

| METODOS

| INICIALIZACION

| FUNCIONES

| PRINT

| EXEC

Esta producción nos ayuda a que los no terminales para declarar variables, asignar variables, ejecutar funciones, métodos y el exec y estos sean ejecutados.

OPCIONESMETODS: OPCIONESMETODS CUERPOMETODO

| CUERPOMETODO

Esta producción hace que nos no puedan venir distintos y muchas producciones ya que aplicamos una recursividad por la izquierda y esta producción se ejecuta en, los métodos, funciones, sentencias cíclicas, sentencias de control.

CUERPOMETODO: SWITCHS

| WHILES

| IFS

| DEC_VAR

| INICIALIZACION ptcoma CUERPOMETODO

| INICIALIZACION

| DECLAVECT

| MODDIFIC

| DECLALIST

| ADDLIST

| BREAK

| FORS

| DOWHILE

| RETURN

| continue

| CALLS

| PRINT

Esta producción nos ayuda a que los no terminales para declarar variables, asignar variables, ejecutar funciones, métodos, while, for,do while, declarar vector, asignar vector, break y agregar a lista, return y estos sean ejecutados.

BREAK: break ptcoma

Esta producción ejecuta la sentencia break, ya involucramos terminales y por ende es una producción final.

DEC_VAR: TIPO identificador

| TIPO identificador igual EXP ptcoma

| TIPO identificador igual CASTEO ptcoma

| TIPO identificador igual ACCESS ptcoma

| TIPO identificador igual TOLOWEER ptcoma

| TIPO identificador igual TOUPPER ptcoma

| TIPO identificador igual LENGTH ptcoma

| TIPO identificador igual TRUNCATE ptcoma

| TIPO identificador igual ROUND ptcoma

| TIPO identificador igual TYPEOF ptcoma

| TIPO identificador igual TOSTRING ptcoma

| TIPO identificador igual TOCHARARRAY ptcoma

| TIPO identificador

Esta producciones nos ayuda a la definir la declaración de variables de ambos tipos, también para las funciones nativas.

RETURN: return EXP ptcoma

Esta producción nos ayuda a definir el return que usado para funciones o sentencias cíclicas o de control.

INICIALIZACION: identificador igual EXP ptcoma

| AUM ptcoma

Esta producción ayuda a iniciar o asignar un valor a una variable ya existen en el lenguaje.

AUM : identificador menosmenos

| identificador masmas

Estas producciones nos ayudan a manejar los incrementos o decrementos en el lenguaje.

CASTEO: parA TIPO parC EXP

Esta producción nos ayuda a manejar y definir los casteos de las variables.

TIPO: char

| boolean

| double

| int

| string

Esta producción nos ayuda a definir los tipos que puede tener una variable, estos pueden ser un tipo char, boolean, double, int o string.

EXP: EXP mas EXP

| EXP menos EXP

| parA EXP parC {$$ = $2}

| EXP div EXP

| EXP multi EXP

| EXP exponente EXP

| menos EXP %prec

| EXP modulo EXP

| identificador

| LLAMADAS

| cadena

| caracter

| numeros

| true

| false

| EXP menor EXP

| EXP mayor EXP

| EXP menorigual

| EXP mayorigual EXP

| EXP diferencia EXP

| EXP igualigual EXP

| EXP or EXP

| EXP and EXP

| not EXP

| identificador masmas

| identificador menosmenos

Esta producción nos ayuda a realizar una operación aritmética, una operación relaciones y operaciones lógicas, también acá usamos las expresiones regulares antes definidas, para lograr identificar las cadenas, caracteres, números e identificadores.

DECLAVECT: TIPO corA corC identificador igual new TIPO corA numeros corC

| TIPO corA corC identificador igual llaveA EXPRESIONES llaveC

Producciones que nos ayudan a declarar y definir un vector.

LISTAVALORES: LISTAVALORES coma EXP

| EXP

Esta producción es la que utilizamos para los valores y las expresiones.

LISTAPARAMETROS: LISTAPARAMETROS coma PARAMETROS

| PARAMETROS

Esta producciones la hacemos recursiva por la izquierda ya que necesitamos recibir uno o mas parámetros por ende lo hacemos de esta manera.

PARAMETROS: TIPO identificador

Producción para definir los parámetros para las funciones o métodos

ACCESS: identificador corA numeros corC

| identificador corA corA numeros corC corC

Esta producción define el cómo acceder a los dato de un vector o lista.

MODDIFIC: identificador corA numeros corC igual EXP ptcoma

|identificador corA corA numeros corC corC igual EXP ptcoma

Esta producción define como se hará la modificación de los vectores o listas.

DECLALIST: list menor TIPO mayor identificador igual new list menor TIPO mayor p tcoma

Esta producción define la declaración de una lista.

ADDLIST: identificador punto add parA EXP parC ptcoma

Producción que define como sea agregan los datos a una lista.

IFS: if parA EXP parC llaveA OPCIONESMETODS llaveC

|if parA EXP parC llaveA OPCIONESMETODS llaveC else llaveA OPCIONESME TODS llaveC

Esta producción nos ayuda a definir el if y el if else, con su expresión y su lista de instrucciones.

| if parA EXP parC llaveA OPCIONESMETODS llaveC ELSEIFS

| if parA EXP parC llaveA OPCIONESMETODS llaveC ELSEIFS else llaveA OPCI ONESMETODS llaveC

producción que nos ayuda a manejar el if con else if y else, ya que nos facilita manejarlo con distintas producciones.

ELSEIFS: ELSEIFS CONELSEIF

|CONELSEIF

Producción que no hac recursivo el else if, por que podremos recibir n else if y para ello necesitamos un recursividad por la izquierda.

CONELSEIF: else if parA EXP parC llaveA OPCIONESMETODS llaveC

Nos ayuda definir el else if con su expresión y su lista de instrucciones.

ELSEIF: else if parA EXP parC llaveA CUERPOMETODO llaveC

Produccion que nos ayuda manejar el else cuando existe varios else if.

SWITCHS: switch parA EXP parC llaveA default dospts OPCIONESMETODS llaveC

| switch parA EXP parC llaveA CASES llaveC

Esta produccion nos ayuda a definir la sentencia Switchm con nus opciones, con Case o Default.

CASES: CASES CONCASE

|CONCASE

Esta producción se hace recursiva por que debemos de manejar varios casteos y por ende debemos de repetir.

CONCASE: case EXP dospts OPCIONESMETODS

Producción que ayuda a definir los Cases.

WHILES: while parA EXP parC llaveA OPCIONESMETODS llaveC

Producción que nos ayuda a definir los whiles, con su expresión y sus instrucciones.

FORS: for parA INIFOR EXP ptcoma AUM parC llaveA OPCIONESMETODS llaveC
producción que no ayuda a definir los for, con su expresión, su variable y su incremento o decremento

INIFOR: DEC_VAR

| INICIALIZACION

producción que nos ayuda a definir que tipo de variable va entrar al for, puede ser una nueva variable o bien usar otra.

DOWHILE: do llaveA OPCIONESMETODS llaveC while parA EXP parC ptcoma

producción que nos ayuda a definir los Do While con sus expresiones y sus instrucciones.

METODOS: void identificador parA LISTAPARAMETROS parC llaveA OPCIONESMETODS llaveC

| void identificador parA parC llaveA OPCIONESMETODS llaveC

Producciones que nos ayuda a definir métodos con o sin parámetros.

LLAMADAS: identificador parA LISTAVALORES parC

| identificador parA parC

producción que nos ayuda a definir las llamadas de los métodos, estas llamadas pueden o no pueden enviar datos, depende si es un método con o sin parámetros.

CALLS: LLAMADAS ptcoma

Producciones que nos sirve para cuando necesitamos hacer un llamamda pero esta con un punto y coma.

;

PRINT: print parA EXP parC ptcoma

| print parA parC ptcoma

Producciones que nos ayuda a definir la instrucción Print.

EXEC: exec identificador parA parC ptcoma

| exec identificador parA LISTAVALORES parC ptcoma

producción que nos ayuda a definir el Exec, este puede ser con o sin valores todo depende del método o función.

Acciones semánticas.

Las acciones semánticas que se tomara para la construcción del árbol fueron aquellas que dan sentido a nuestra sintaxis, primero nos tenemos que fijar que en nuestro entrada no venga ningún error de tipo sintáctico o léxico, ya que si viene un nuestro árbol no será graficado, ya luego vamos validando las operaciones o instrucciones a realizar, estas son dadas por la entrada, debemos de construir el árbol empezando de un exec, luego vamos a construir el árbol según encontremos las instrucciones hasta llegar a una variable o un Terminal en este caso.