Escuela de Ingeniería en Computación

Compiladores e Intérpretes

Profesora:
Erika Marín

Proyecto II - Parser

Estudiante:
Gabriela Garro Abdykerimov

16 de Noviembre del 2016

Evaluación de funcionalidades

Funcionalidad	**Estado**	**Justificación**
Reconocimiento de errores léxicos	100%	
Reconocimiento de estructura general del programa	100%	
Reconocimiento de sección de constantes (que se encuentre y que no)	100%	
Reconocimiento de sección de globales (VAR) (que se encuentre y que no)	100%	
Reconocimiento de sección de funciones (que se encuentre y que no)	100%	
Declaración y recuperación de errores de un procedimiento	100%	
Declaración y recuperación de errores de una función	100%	
Estructura de WHILE	100%	
Estructura de FOR DO	100%	
Estructura de REPEAT UNTIL	100%	
Estructura de CASE OF	100%	
Estructura IF THEN	0%	Solo reconoce IF THEN ELSE
Estructura de IF THEN ELSE	100%	

Pruebas y resultados

Código de prueba	**Output**
PROGRAM programa	Errores léxicos: Errores sintácticos: Error #1 : Error fatal de sintaxis al final del programa.

PROGRAM Prueba
 	
VAR
 	
numero,limite,contador:int; // todo esta bien
 	
:int; //hizo falta el identificador
 	
respuesta : ; // falta tipo
 	
coleccion2: ARRAY[5] longint; // error porque falta el OF
 	
BEGIN
END
	Errores léxicos:
	
	Errores sintácticos:
	Error #1 : Mala declaracion de tipo
	Error #2 : Esperando identificador
	Error #3 : Mala declaracion de tipo
	Error #4 en la línea 11, columna 23 : Error de sintaxis en token longint.
	Error #5 : Mala declaracion de tipo

PROGRAM programa
 	
CONST
pi = 3.14;
nombre = "Pepe";
 	
VAR
var0: LONGINT;
var1,var5: STRING;
var3: INT;
var4: ARRAY[1..100] OF STRING;
var7: int;
 	
BEGIN
 	
END
	Errores léxicos:
	
	Errores sintácticos:

PROGRAM programa	Errores léxicos:

CONST
pi = 3.14;
nombre = "Pepe";
tres = tres; //una constante no puede ser una variable
 	
BEGIN
 	
END
	Errores sintácticos:
	Error #1 en la línea 6, columna 8 : Error de sintaxis en token tres.
	Error #2 : Error en declaración de constante

PROGRAM programa
 	
PROCEDURE funcion2(x, y) //no declaro tipos de los parametros
BEGIN
funcion2:=0;
END
 	
BEGIN
 	
END
	Errores léxicos:
	
	Errores sintácticos:
	Error #1 : Error en lista de parámetros

PROGRAM programa
 	
PROCEDURE funcion3(int, int) //no definio los identificadores
BEGIN
funcion3:=0;
END
 	
BEGIN
 	
END
	Errores léxicos:
	
	Errores sintácticos:
	Error #1 en la línea 3, columna 21 : Error de sintaxis en token int.
	Error en lista de parámetros

PROGRAM programa
 	
FUNCTION funcion4 () // es una funcion por lo que debe tener tipo de retorno
BEGIN
funcion4:=0;
END
 	
BEGIN
 	
END
	Errores léxicos:
	
	Errores sintácticos:
	Error #1 en la línea 4, columna 1 : Error de sintaxis en token BEGIN.
	Las funciones deben de tener valor de retorno

PROGRAM programa
 	
BEGIN
 	
WHILE (a>3) DO
BEGIN
i := 0;
sdsd+;
END
END
	Errores léxicos:
	
	Errores sintácticos:
	Error #1 : Error en sentencia

PROGRAM programa
 	
BEGIN
 	
for X:= 1 to 10 do
BEGIN
while (a 8) do
BEGIN
i:=0;
END
END
END
	Errores léxicos:
	
	Errores sintácticos:
	Error #1 en la línea 7, columna 21 : Error de sintaxis en token 8.
	Error en condición booleana

PROGRAM programa
 	
BEGIN
 	
for X< 1 to 10 do
BEGIN
while (a > 8) do
BEGIN
i:=0;
END
END
END
	Errores léxicos:
	
	Errores sintácticos:
	Error #1 en la línea 11, columna 7 : Error de sintaxis en token END.
	Expresión no es una asignación

PROGRAM programa
 	
BEGIN
 	
REPEAT
I:= era;
y = 3;
UNTIL z>2
 	
END
	Errores léxicos:
	
	Errores sintácticos:
	Error #1 : Error en sentencia

PROGRAM programa
 	
BEGIN
 	
IF s>=9 THEN
chuichui:=eeee;
ELSE
chuichui:=elele;
END
	Errores léxicos:
	
	Errores sintácticos:

PROGRAM programa
 	
BEGIN
 	
IF s>=9 THEN
chuichui:=eeee;
(*ELSE
chuichui:=elele;*)
END
	Errores léxicos:
	
	Errores sintácticos:
	Error #1 en la línea 9, columna 1 : Error de sintaxis en token END.
	Error #2 : Error fatal de sintaxis al final del programa.

Gramática

```
declaracion_programa ::= PR_PROGRAM IDENTIFICADOR PR_BEGIN
seccion_instrucciones PR_END
        | PR_PROGRAM IDENTIFICADOR secciones_opcionales PR_BEGIN
seccion_instrucciones PR_END;

    secciones_opcionales ::= seccion_constantes seccion_globales
seccion_funciones
        | seccion_constantes
        | seccion_globales
        | seccion_funciones
        | seccion_constantes seccion_globales
        | seccion_globales seccion_funciones
        | seccion_constantes seccion_funciones;

    seccion_constantes ::= TIPO_CONST seccion_constantes1;

    seccion_constantes1 ::= declaracion_constante
        | declaracion_constante seccion_constantes1
        ;

    declaracion_constante ::= IDENTIFICADOR OP_IGUAL
declaracion_constante1 SEMI;

    seccion_globales ::= PR_VAR seccion_globales1
        ;

    seccion_globales1 ::= declaracion_global
        | declaracion_global seccion_globales1
        ;

    declaracion_global ::=  IDENTIFICADOR OP_COMA
declaracion_global1
        | declaracion_global1
        ;

    declaracion_global1 ::= IDENTIFICADOR OP_DOSPUNTOS
declaracion_global2 SEMI;

    seccion_funciones ::= function | procedure
        | function seccion_funciones
        | procedure seccion_funciones
        ;

    function ::= PR_FUNCTION IDENTIFICADOR:id1 PARENTIZQ
parametros PARENTDER OP_DOSPUNTOS
```

```
        tipo PR_BEGIN seccion_instrucciones IDENTIFICADOR:id2
OP_DOSPUNTOSIGUAL retorno SEMI PR_END;

    procedure ::= PR_PROCEDURE IDENTIFICADOR:id1 PARENTIZQ
parametros PARENTDER
        PR_BEGIN seccion_instrucciones IDENTIFICADOR:id2
OP_DOSPUNTOSIGUAL retorno SEMI PR_END;

    parametros ::= //IDENTIFICADOR OP_DOSPUNTOS tipo
        | IDENTIFICADOR OP_DOSPUNTOS tipo parametros1;

    parametros1 ::= //OP_COMA IDENTIFICADOR OP_DOSPUNTOS tipo
        | OP_COMA IDENTIFICADOR OP_DOSPUNTOS tipo parametros1;

    seccion_instrucciones ::= | seccion_instrucciones1
seccion_instrucciones;

    seccion_instrucciones1 ::= bloque_while | bloque_if |
bloque_case | bloque_repeat | bloque_for
        | error:e ;

    bloque_while ::= PR_WHILE PARENTIZQ condicion_booleana
PARENTDER PR_DO
        PR_BEGIN cuerpo_estructura_control PR_END
        | PR_WHILE error:e PR_DO PR_BEGIN
cuerpo_estructura_control PR_END;

    bloque_for ::= PR_FOR IDENTIFICADOR OP_DOSPUNTOSIGUAL ENTERO
PR_TO ENTERO PR_DO
        PR_BEGIN cuerpo_estructura_control PR_END
        | PR_FOR error:e PR_TO ENTERO PR_DO
        PR_BEGIN cuerpo_estructura_control PR_END
        | PR_FOR IDENTIFICADOR OP_DOSPUNTOSIGUAL ENTERO error:e
ENTERO PR_DO
        PR_BEGIN cuerpo_estructura_control PR_END;

    bloque_repeat ::= PR_REPEAT cuerpo_estructura_control
PR_UNTIL condicion_booleana;

    bloque_if ::= PR_IF condicion_booleana PR_THEN sentencia SEMI
PR_ELSE sentencia SEMI
        | error:e SEMI ;

    bloque_case ::= PR_CASE IDENTIFICADOR PR_OF
bloque_constantes_case PR_ELSE sentencia SEMI;

    bloque_constantes_case ::= | bloque_constantes_case1
bloque_constantes_case;
```

```
    bloque_constantes_case1 ::= declaracion_constante1
OP_DOSPUNTOS cuerpo_estructura_control

        | declaracion_constante1 error:e
cuerpo_estructura_control;

    declaracion_constante1 ::= STRING | ENTERO | REAL;

    condicion_booleana ::= condicion_booleana1 | condicion_not
        | PARENTIZQ condicion_booleana1 PARENTDER
operador_booleano PARENTIZQ condicion_booleana1 PARENTDER
        | PR_TRUE | PR_FALSE ;

    condicion_not ::= OP_NOT IDENTIFICADOR
        | OP_NOT PARENTIZQ condicion_booleana1 PARENTDER;

    condicion_booleana1 ::= condicion_booleana2 operador_booleano
condicion_booleana2;

    condicion_booleana2 ::= IDENTIFICADOR | ENTERO | REAL;

    cuerpo_estructura_control ::= | cuerpo_estructura_control1
cuerpo_estructura_control;

    cuerpo_estructura_control1 ::= sentencia SEMI | bloque_while
| bloque_if | bloque_case
        | bloque_repeat | bloque_for;

    sentencia ::= IDENTIFICADOR OP_DOSPUNTOSIGUAL retorno
        | IDENTIFICADOR OP_SUMASUMA
        | IDENTIFICADOR OP_MENOSMENOS
        | IDENTIFICADOR OP_DOSPUNTOSIGUAL retorno
operador_aritmetico retorno
        | IDENTIFICADOR error:e;

    retorno ::= IDENTIFICADOR | ENTERO | REAL | STRING;

    tipo ::= TIPO_INT | TIPO_LONGINT | TIPO_SHORTINT | TIPO_CHAR
| TIPO_STRING | TIPO_BOOLEAN | TIPO_REAL;

    array ::= TIPO_ARRAY OP_BRACKETIZQ ENTERO OP_PUNTO OP_PUNTO
ENTERO OP_BRACKETDER PR_OF tipo
        | TIPO_ARRAY OP_BRACKETIZQ ENTERO OP_BRACKETDER PR_OF
tipo;

    operador_aritmetico ::= OP_SUMA | OP_RESTA | OP_MULT |
OP_DIVISION | OP_MOD | PARENTIZQ
        | PARENTDER | OP_MASIGUAL | OP_MENOSIGUAL | OP_MULTIGUAL
| OP_DIVIGUAL | OP_DIV ;
```

```
    operador_booleano ::= OP_IGUAL | OP_MAYORIGUAL | OP_MAYOR |
OP_MENORIGUAL | OP_MENOR
        | OP_MENORMAYOR | OP_OR | OP_AND ;
```