PRÁCTICA DE PROCESADORES DE LENGUAJES

Curso 2009 – 2010

Entrega de Febrero

NOMBRE Y APELLIDOS: Juan Antonio Elices Crespo

DNI: 12411633-M

CENTRO ASOCIADO: Palencia

MAIL DE CONTACTO: juanantonio.elices@gmail.com

TELÉFONO DE CONTACTO: (+1) 505-712-2210

IDENTIFICADOR: jelices1

GRUPO (A ó B): B

1. El analizador léxico

Nuestro analizador léxico tiene 3 estados: yyinitial, comentario y cadena. El estado predefinido es yyinitial y se encuentra en este estado cuando una secuencia puede ser una palabra reservada.

Si en este estado aparece el simbolo '{' pasa al estado comentario y en caso de que aparezca el carácter “ pasaría al estado cadena.

Aunque el estado cadena no es necesario, ya que se podria haber facilmente implementado una expresion regular que reconociese cadenas, se optó por esta opción para aprender como funcionan los estados de jflex.

Tambien se desea comentar la utilización de la directiva `%eofval` para comprobar que al acabar el archivo no nos encontramos en un comentario. Lo cual sería motivo de un error léxico.

2. El analizador sintáctico

Se debe comentar que nuestra gramática literalmente es ambigua, pero CUP nos permite definir la grámatica de esta manera y solucionara los conflictos definiendo una prioridad y una asociatividad.

La solución al else ambiguo, en vez de buscar una gramatica complicada que no tuviese este problema. Se ha definido el else como un operador y se le ha dado una asociatividad por la izquierda. Por lo que en caso de que apezca el siguiente código:

```
if ( expresion1) then

 if (expresion2) then

      sentencias

   else sentencias
```

el else pertenezca al segundo if, es decir al if más próximo.

Otro aspecto digno de mencionar es que la diferenciación entre una expresión lógica y una aritmética no se ha realizado, se ha decidido posponer al análisis semántico.

3. Conclusiones

Las pruebas realizadas a esta gramatica han sido satisfactorias. Se ha realizado un programa de ejemplo bien construido con todas las funcionalidades y ha sido reconocido por el analizador sintáctico. A este programa se le han añadido algunos errores para ver si les reconocía, y en ningún caso ha reconocido dicha gramatica. Aunque sabemos que esto no garantiza la total correción, pensamos que nuestra gramatica se ajusta perfectamente al enunciado.

4. Gramatica

La gramatica la mostramos según el programa AnaGra de la Universidad de Zaragoza. Es decir una regla es NOTERMINAL: [TERMINAL |NOTERMINAL]*;

Los terminales los hemos puesto en minúsculas o si es un solo carácter esta entre comillas simples. Los no-terminales están en mayúsculas.

```
PROGRAM: program identificador ';' DECLARACIONES SUBPROGRAMAS CUERPO '.' ;

DECLARACIONES: SECCIONCONSTANTES SECCIONTIPOS SECCIONVARIABLES;



SECCIONCONSTANTES: const DECLARACIONCONSTANTES | ;

DECLARACIONCONSTANTES: DECLARACIONCONSTANTE | DECLARACIONCONSTANTE
DECLARACIONCONSTANTES;

DECLARACIONCONSTANTE: LISTAIDENTIFICADORES '=' VALORCONSTANTE ';';

LISTAIDENTIFICADORES: identificador ',' LISTAIDENTIFICADORES | identificador;

VALORCONSTANTE: true | false | literalEntero;



SECCIONTIPOS: type DECLARACIONTIPOS | ;

DECLARACIONTIPOS: DECLARACIONTIPO | DECLARACIONTIPO DECLARACIONTIPOS;

DECLARACIONTIPO: identificador '=' DEFINICIONTIPO;

DEFINICIONTIPO: record CAMPOSREGISTRO end ';';

CAMPOSREGISTRO: CAMPOREGISTRO | CAMPOREGISTRO CAMPOSREGISTRO;

CAMPOREGISTRO: LISTAIDENTIFICADORES ':' TIPOPRIMITIVO ';' ;

TIPOPRIMITIVO: integer | boolean | '^' integer;



SECCIONVARIABLES: var DECLARACIONVARIABLES | ;

DECLARACIONVARIABLES: DECLARACIONVARIABLE | DECLARACIONVARIABLE DECLARACIONVARIABLES;

DECLARACIONVARIABLE: LISTAIDENTIFICADORES ':' TIPOVARIABLE ';';

TIPOVARIABLE : TIPOPRIMITIVO | identificador;



SUBPROGRAMAS: SUBPROGRAMA SUBPROGRAMAS | ;

SUBPROGRAMA: PROCEDIMIENTO | FUNCION;

PROCEDIMIENTO: procedure identificador '(' DEFINICIONARGUMENTOS ')' ';' DECLARACIONES
SUBPROGRAMAS CUERPO ';';
```

```
DEFINICIONARGUMENTOS: ARGUMENTOS | ;

ARGUMENTOS: ARGUMENTO | ARGUMENTO ';' ARGUMENTOS;

ARGUMENTO: LISTAIDENTIFICADORES ':' TIPOPRIMITIVO;

FUNCION: function identificador '(' DEFINICIONARGUMENTOS ')' ':' TIPOPRIMITIVO ';'
DECLARACIONES SUBPROGRAMAS CUERPO ';';

CUERPO: begin LISTASENTENCIAS end;



LISTASENTENCIAS: SENTENCIA LISTASENTENCIAS | ;

SENTENCIA: SENTENCIAASIGNACION | SENTENCIAIF | SENTENCIAFOR |
LLAMADAFUNCIONOPROCEDIMIENTO ';' | SENTENCIAES;

SENTENCIAASIGNACION: REFERENCIA ':=' EXPRESION ';';

REFERENCIA: identificador | identificador '.' identificador;

EXPRESION: VALORCONSTANTE | '(' EXPRESION ')' | REFERENCIA |
LLAMADAFUNCIONOPROCEDIMIENTO | EXPRESIONPUNTERO;

LLAMADAFUNCIONOPROCEDIMIENTO: identificador '(' VALORPARAMETROS ')' | identificador '('
')';

VALORPARAMETROS: EXPRESION ',' VALORPARAMETROS | EXPRESION;

EXPRESIONPUNTERO: '@' identificador | identificador '^';



SENTENCIAIF: if '(' EXPRESION ')' then BLOQUESENTENCIAS PARTEELSE;

PARTEELSE: else BLOQUESENTENCIAS ;

BLOQUESENTENCIAS: SENTENCIA | begin LISTASENTENCIAS end ';';

SENTENCIAFOR: for '(' REFERENCIA ':=' EXPRESION to EXPRESION ')' do BLOQUESENTENCIAS;

SENTENCIAES: write '(' MOSTRADOPANTALLA ')' ';' | writeln '(' ')' ';';

MOSTRADOPANTALLA : cadena | EXPRESION ;
```