Manual técnico

Este programa fue desarrollado en Java utilizando dos herramientas que permiten el análisis y ejecución del lenguaje que permite este programa

En JFLEX se realiza todo lo que respecta a análisis léxico donde se declaran palabras o símbolos reservados del sistema de la siguiente manera

```
// palabras reservadas
TK_PROGRAM = "program"
TK_VAR = "var"
DOSPUNTOS = ":"
```

De esta manera la herramienta se encarga de validar estos tokens y guardarlos con el nombre que se le da.

Este análisis léxico lo que hace únicamente es la verificación y validación de tokens que sean permitidos por el programador, en Jflex no se encarga de ver el orden ni la lógica del funcionamiento de este programa

Para el manejo en cup se pueden retornar como símbolos para que cup los lea y pueda realizar el análisis sintáctico de los tokens según estén en el código a analizar

```
{TK_PROGRAM} {return ( new Symbol(sym.TK_PROGRAM,yyline,yycolumn,yytext());}
```

El análisis sintáctico se encarga de leer toda la tabla de tokens dada por el analizador léxico y acá si se encarga de revisar el orden de dichos tokens, pero en esta parte no se encarga de revisar la lógica del código, es decir que si acá se intenta declarar un double pero se utiliza como char[] esta parte no se encarga de esto, ya que acá únicamente se analiza el orden de dichos tokens.

```
VAR DOSPUNTOS TIPO DOSPUNTOS DOSPUNTOS ID MENORQ GUION EXPRESION END PYCOMA
```

Así como en este código es lo que ve CUP o el analizador sintáctico, en esta parte no se revisa la lógica del programa como tal, sino que únicamente el orden, la parte de la lógica se encarga el analizador semántico

Para el uso de la herramienta cup se requiere de los tokens dados por jflex y de una gramática libre del contexto que es la gramática que le sirve al analizador sintáctico, en este caso se utilizó una gramática en formato BNF que es de la siguiente manera

NOTERMINAL ::= OTRONOTERMINAL TERMINAL

En una gramática libre del contexto se tiene la restricción de que un no terminal puede derivar en una cadena finita de no terminales y terminales

Para el uso de cup se requiere una declaración previa de los no terminales y los terminales a utilizar en el uso de la gramática

```
terminal DOSP, MENORQ, MAYORQ, PYCOMA, CORA, CORC, ARRB, PARA, PARC, COMA,
IGUAL, MENOS;
terminal TK_PROGRAM, TK_VAR, TK_CADENA, TK_CHAR, TK_END, TK_ARR, TK_SUM,
TK_RES, TK_DOUBLEERR, TK_CHARERR;
terminal TK_MUL, TK_DIV, TK_MOD, TK_MEDIA, TK_MEDIANA, TK_MODA, TK_VARIANZA,
TK_MAX, TK_MIN;
terminal TK_CONSOLE, TK_PRINT, TK_COLUMN, TK_GRAPHBAR, TK_GRAPHPIE,
TK_GRAPHLINE, TK_HISTOGRAM;
terminal TK_TITULO, TK_EJEX, TK_EJEY, TK_TITULOX, TK_TITULOY, TK_EXEC,
TK_VALUES, TK_LABEL;
terminal String ENTEROS, DECIMAL, STRINGT, NOMBRES;


nonterminal INICIO, CODIGO, OTROCODIGO, VARIABLE, ARREGLO, IDARREGLO,
LISTA_VALORES;
nonterminal OTRA_EXPRESION, TIPO, EXPRESION, OPERACION, ESTADISTICA, DATOS;
nonterminal IMPRESION, TIPOIMPRESION, ARREGLOIMP, GRAFICA;
nonterminal OTRAEXPRESIONIMP, GRAPHBAR, GRAPHPIE, GRAPHLINE, HISTOGRAM,
INSTRUCCIONESGB, INSTRUCCIONB;
nonterminal TITULOGRAFICA, EJEX, EJEY, TITULOX, TITULOY, INSTRUCCIONESGP,
INSTRUCCIONP, VALUESGRAPH, INSTRUCCIONESGL;
nonterminal INSTRUCCIONL, INSTRUCCIONESH, INSTRUCCIONH, LABELGRAPH;
nonterminal TITULOGRAFICAPIE, TITULOGRAFICALINEA, EJEXLINEA, EJEYLINEA,
TITULOXLINEA, TITULOYLINEA;
nonterminal VALUESGRAPHHI, TITULOGRAFICAHI;
```

En la gramática utilizada para este proyecto estos serian los no terminales y los terminales utilizados por cup, también los terminales son los símbolos que retorna jflex para en esta parte cup comprobar su orden, la gramática utilizada en este proyecto es la siguiente

```
INICIO ::= TK_PROGRAM CODIGO OTROCODIGO TK_END TK_PROGRAM

CODIGO ::= VARIABLE | ARREGLO | OPERACION | ESTADISTICA | IMPRESION |
GRAFICA
OTROCODIGO ::= CODIGO OTROCODIGO
VARIABLE ::= TK_VAR DOSP TIPO DOSP DOSP NOMBRES MENORQ MENOS EXPRESION
TK_END PYCOMA
ARREGLO ::= TK_ARR DOSP TIPO DOSP DOSP IDARREGLO MENORQ MENOS LISTA_VALORES
TK_END PYCOMA
TIPO ::= TK_CHAR | TK_DOUBLE

IDARREGLO ::= ARROBA NOMBRE
LISTA_VALORES ::= CORA EXPRESION OTRA_EXPRESION CORC
OTRA_EXPRESION ::= COMA EXPRESION OTRA_EXPRESION | E
EXPRESION ::= ENTEROS | DECIMAL | STRING | NOMBRE | OPERACION | IDARREGLO |
ESTADISTICA
OPERACION ::=  TK_SUM PARA EXPRESION COMA EXPRESION PARC
          |TK_RES PARA EXPRESION COMA EXPRESION PARC
          |TK_MUL PARA EXPRESION COMA EXPRESION PARC
          |TK_DIV PARA EXPRESION COMA EXPRESION PARC
          |TK_MOD PARA EXPRESION COMA EXPRESION PARC
ESTADISTICA ::=  TK_MEDIA PARA DATOS PARC
        |TK_MEDIANA PARA DATOS PARC
        |TK_MODA PARA DATOS PARC
        |TK_VARIANZA PARA DATOS PARC
        |TK_MAX PARA DATOS PARC
        |TK_MIN PARA DATOS PARC
DATOS ::= IDARREGLO | LISTA_VALORES
IMPRESION ::= TK_CONSOLE DOSP DOSP TIPOIMPRESION TK_END PYCOMA
TIPOIMPRESION ::= TK_PRINT IGUAL EXPRESION OTRAEXPRESIONIMP
        | TK_COLUM IGUAL EXPRESION MENOS MAYORQ ARREGLOIMP
ARREGLOIMP ::= IDARREGLO | LISTA_VALORES
OTRAEXPRESIONIMP ::= OTRAEXPRESIONIMP COMA EXPRESION | E
GRAFICA ::= GRAPHBAR | GRAPHPIE | GRAPHLINE | HISTOGRAM
GRAPHBAR ::= TK_GRAPHBAR PARA INSTRUCCIONESGB TK_EXEC TK_GRAPHBAR TK_END
PYCOMA TK_END PYCOMA
INSTRUCCIONESGB ::= INSTRUCCIONESGB INSTRUCCIONB | E
INSTRUCCIONB ::= TITULOGRAFICA | EJEX | EJEY | TITULOX | TITULOY
TITULOGRAFICA ::= TK_TITULO DOSP DOSP TIPO IGUAL EXPRESION TK_END PYCOMA
EJEX ::= TK_EJEX DOSP DOSP TIPO IGUAL ARREGLOIMP TK_END PYCOMA
EJEY ::= TK_EJEY DOSP DOPS TIPO IGUAL ARREGLOIMO TK_END PYCOMA
TITULOX ::= TK_TITULOX DOSP DOSP TIPO IGUAL EXPRESION TK_END PYCOMA
TITULOY ::= TK_TITULOY DOSP DOSP TIPO IGUAL EXPRESION TK_END PYCOMA
GRAPHPIE ::= TK_GRAPHPIE PARA INSTRUCCIONESGP TK_EXEC TK_GRAPHPIE TK_END
PYCOMA PARC TK_END PYCOMA
INSTRUCCIONESGP ::= INSTRUCCIONESGP INSTRUCCIONP
```

```
INSTRUCCIONP ::= TITULOGRAFICAPIE | VALUESGRAPH | LABELGRAPH
TITULOGRAFICAPIE ::= TK_TITULO DOSP DOSP TIPO IGUAL EXPRESION TK_END PYCOMA
VALUESGRAPH ::= TK_VALUES DOSP DOSP TIPO IGUAL ARREGLOIMP TK_END PYCOMA
LABELGRAPH ::= TK_LABEL DOSP DOSP TIPO IGUAL ARREGLOIMP TK_END PYCOMA
GRAPHLINE ::= TK_GRAPHLINE PARA INSTRUCCIONESGL TK_EXEC TK_GRAPHLINE TK_END
PYCOMA PARC TK_END PYCOMA
INSTRUCCIONESGL ::= INSTRUCCIONESGL INSTRUCCIONL | E
INSTRUCCIONL ::= TITULOGRAFICALINEA | EJEXLINEA | EJEYLINEA | TITULOXLINEA |
TITULOYLINEA
TITULOGRAFICALINEA ::= TK_TITULO DOSP DOSP TIPO IGUAL EXPRESION TK_END
PYCOMA
EJEXLINEA ::= TK_EJEY DOSP DOSP TIPO IGUAL ARREGLOIMP TK_END PYCOMA
TITULOXLINEA ::= DOSP DOSP TIPO IGUAL EXPRESION TK_END PYCOMA
TITULOYLINEA :: TK_TITULOY DOSP DOSP TIPO IGUAL EXPRESION TK_END PYCOMA
HISTOGRAM ::= TK_HISTOGRAM PARA INSTRUCCIONESH TK_EXEC EK_HISTOGRAM TK_END
PYCOMA PARC TK_END PYCOMA
INSTRUCCIONESH ::= INSTRUCCIONESH INSTRUCCIONH | E
INSTRUCCIONH ::= TITULOGRAFICAHI | VALUESGRAPHHI
TITULOGRAFICAHI :: TK_TITULO DOSP DOSP TIPO IGUAL EXPRESION TK_END PYCOMA
VALUESGRAPHHI ::= TK_VALIES DOSP DOSP TIPO IGUAL ARREGLOIMP TK_END PYCOMA
```

Estas son las reglas que definen cómo se pueden combinar los terminales y no terminales para formar estructuras más grandes. Cada regla de producción tiene una parte izquierda (el no terminal que se está definiendo) y una parte derecha (la secuencia de terminales y no terminales que forman el no terminal de la izquierda).

La herramienta CUP nos permite trabajar código de java dentro de {: /* El código acá */ :}

Lo que nos permite realizar las funciones respectivas en lo que corresponde a este lenguaje, por ejemplo

```
VARIABLE ::= TK_VAR DOSP TIPO:t DOSP DOSP NOMBRES:n MENORQ MENOS EXPRESION:e
TK_END PYCOMA
{:
    data.put(n.toString(), e);
    }
:};
```

En esta parte se esta haciendo uso de un hashmap llamado data que este es el que no servirá para manejar estas variables, y en esta parte se puede manejar cualquier código de java según se necesite.