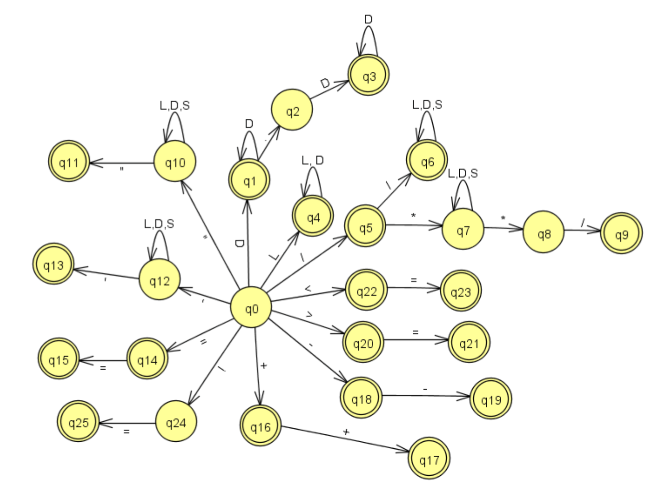
**Manual Técnico**

Dentro de este manual se detallará el proceso para determinar el AFD para el analizador léxico y la gramática libre de contexto para el análisis sintáctico.

Para el AFD se utilizó la siguiente expresión regular:

Donde D son los dígitos, L las letras y S los símbolos. En el caso de /\* (L|D|S)\* \*/ se omite el \* de los símbolos para poder saber cuando es que pasa al siguiente estado.

**D D\* | D D\*.D D\* | L (L|D|S)\* | ‘ (L|D|S)\* ’ | “ (L|D|S)\* ” | // (L|D|S)\* | /\* (L|D|S)\* \*/ | =(=|ɛ) | != | +(+|ɛ) | -(-|ɛ) | >(=|ɛ) | <(=|ɛ)**



Para la programación se agregaron los símbolos que faltan de token y las palabras reservadas se detectan cuando se guarda un token. Se utilizaron estados, los cuales están a continuación:

Para la gramática libre de contexto:

Ahora para determinar la gramática libre de contexto se armo viendo ejemplos redactados en el enunciado de la practica y ejemplos de entradas que pueden entrar en el proyecto.

La gramática está a continuación:

<Start> :=

WR\_Class ID S\_Open\_Key

<FunctionList>

WR\_Void WR\_Main S\_Open\_Parenthesis S\_Close\_Parenthesis S\_Open\_Key

<SentencesList>

S\_Close\_Key

<FunctionList>

S\_Close\_Key

<FunctionList> :=

<Type> ID S\_Open\_Parenthesis <Parameter> S\_Close\_Parenthesis S\_Open\_Key

<SentencesList>

<ReturnOrNot>

S\_Close\_Key <FunctionList>

<ReturnOrNot> := WR\_Return <Expression> | ɛ

<Type> := WR\_Int | WR\_Double | WR\_Char | WR\_String | WR\_Bool | WR\_Void

<Parameter> := <ParameterDeclaration> | ɛ

<ParameterDeclaration> := <Type> ID <ParameterList>

<ParameterList> := S\_Comma <ParameterDeclaration> | ɛ

<SentencesList> :=

| <DeclarationSentence> <SentencesList> /

| <AsignmentSentence> <SentencesList> /

| <PrintSentence> <SentencesList> /

| <IfElseSentence> <SentencesList> /

| <SwitchSentence> <SentencesList> /

| <ForSentence> <SentencesList>

| <WhileSentence> <SentencesList> /

| <DoWhileSentence> <SentencesList> /

| ɛ

<DeclarationSentence> :=

<Type> <VariablesDeclaration>

<VariablesDeclaration> := ID <IDList> <OptAsignment> S\_SEMICOLON

<IDList> := S\_COMMA ID <IDList> | ɛ

<OptAsignment> := S\_EQUALS <Expression> | ɛ

<AsignmentSentence> := ID S\_EQUALS <Expression> S\_SEMICOLON

<PrintSentence> := WR\_CONSOLE S\_POINT WR\_WRITE S\_Open\_Parenthesis <Expression> S\_Close\_Parenthesis S\_SEMICOLON

<IfElseSentence> :=

WR\_IF S\_Open\_Parenthesis <Expression> S\_Close\_Parenthesis S\_Open\_Key

<SentencesList>

S\_Close\_Key <OptElse>

<OptElse> :=

WR\_ELSE <OptElseIf> S\_Open\_Key

<SentencesList>

S\_Close\_Key

| ɛ

<OptElseIf> := WR\_IF S\_Open\_Parenthesis <Expression> S\_Close\_Parenthesis

<SwitchSentence> :=

WR\_SWITCH S\_Open\_Parenthesis ID S\_Close\_Parenthesis S\_Open\_Key

<CaseList>

<OptDefault>

S\_Close\_Key

<CaseList> :=

WR\_CASE <CaseValue> S\_TWO\_POINTS

<SentencesList>

WR\_BREAK S\_SEMICOLON

| ɛ

<CaseValue> := STRING | INTEGER | DECIMAL

<OptDefault> :=

WR\_DEFAULT S\_TWO\_POINTS

<SentencesList>

WR\_BREAK S\_SEMICOLON | ɛ

<ForSentence> :=

WR\_FOR S\_Open\_Parenthesis <Expression> S\_Close\_Parenthesis

<WhileSentence> :=

WR\_WHILE S\_Open\_Parenthesis <Expression> S\_Close\_Parenthesis S\_Open\_Key

<SentencesList>

S\_Close\_Key

<DoWhileSentence> :=

WR\_DO S\_Open\_Key

<SentencesList>

S\_Close\_Key WR\_WHILE S\_Open\_Parenthesis <Expression> S\_Close\_Parenthesis S\_SEMICOLON

<Expression> := <E> <OptComparisonSymbol>

<OptComparisonSymbol> :=

S\_EQUALS\_EQUALS <E>

| S\_MAJOR <E>

| S\_LESS <E>

| S\_MAJOR\_EQUALS <E>

| S\_LESS\_EQUALS <E>

| S\_DIFFERENT <E>

| ɛ

<E> := <T> <EP>

<EP> :=

S\_PLUS<T><EP>

| S\_MINUS<T><EP>

| ɛ

<T> := <F><TP>

<TP> :=

S\_PRODUCT <F>

| S\_DIVISION <F>

| ɛ

<F> :=

INTEGER

| DECIMAL

| STRING

| ID

| WR\_TRUE

| WR\_FALSE