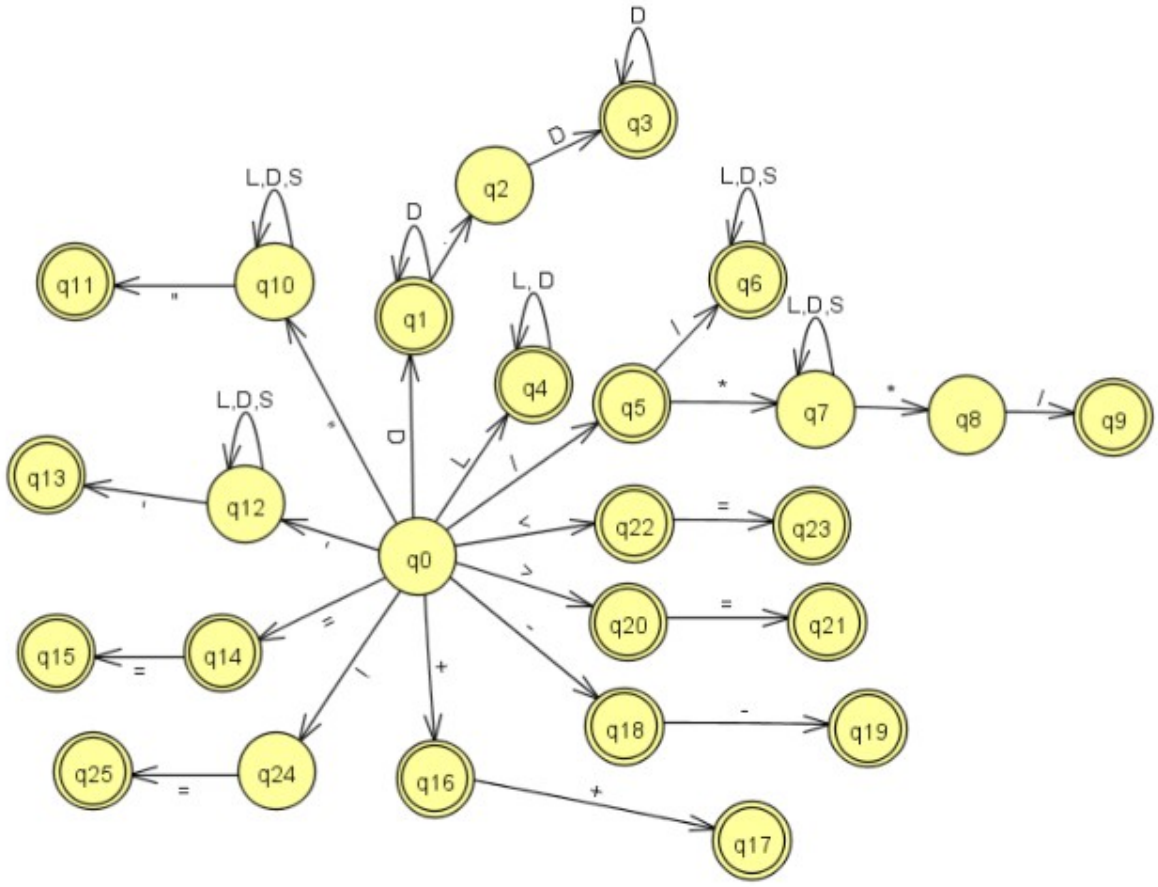
Manual Técnico

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)* */ | =(=|ε) | != | +(+|ε) | -(-|ε) | >(=|ε) | <(=|ε)



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

<InsideClass>

S_CLOSE_KEY

<InsideClass> :=

| <Type> ID <FunctionOrNot> <InsideClass>

| WR_VOID <MainOrNot> <InsideClass>

| ε

<FunctionOrNot> :=

| S_OPEN_PARENTHESIS <Parameter> S_CLOSE_PARENTHESIS S_OPEN_KEY

<SentencesList>

<SentencesListFunction>

S_CLOSE_KEY

| <IDList> <OptAssignment> S_SEMICOLON

<SentencesListFunction> :=

| <ReturnFunction> <SentencesList> <SentencesListFunction>

| ε

<ReturnFunction> := WR_RETURN <Expression> S_SEMICOLON

<MainOrNot> :=

| WR_MAIN S_OPEN_PARENTHESIS S_CLOSE_PARENTHESIS S_OPEN_KEY

<SentencesList>

S_CLOSE_KEY

| ID S_OPEN_PARENTHESIS <Parameter> S_CLOSE_PARENTHESIS S_OPEN_KEY
<SentencesList>
<SentencesListMethod>
S_CLOSE_KEY

<SentencesListMethod> :=
| <ReturnMethod> <SentencesList> <SentencesListMethod>
| ε

<ReturnMethod> := WR_RETURN S_SEMICOLON

<SentencesListLoops> :=
| WR_BREAK S_SEMICOLON
| WR_CONTINUE S_SEMICOLON
| ε

<SentencesListSwitch> :=
| WR_BREAK S_SEMICOLON
| ε

<Type> := WR_Int | WR_Double | WR_Char | WR_String | WR_Bool

<Parameter> := <ParameterDeclaration> | ε
<ParameterDeclaration> := <Type> ID <ParameterList>
<ParameterList> := S_Comma <ParameterDeclaration> | ε

<SentencesList> :=
| <DeclarationSentence> <SentencesList>

| <AsignmentOrCallSentence> <SentencesList>

| <PrintSentence> <SentencesList>

| <IfElseSentence> <SentencesList>

| <SwitchSentence> <SentencesList>

| <ForSentence> <SentencesList>

| <WhileSentence> <SentencesList>

| <DoWhileSentence> <SentencesList>

| ε

<DeclarationSentence> :=

<Type> <VariablesDeclaration>

<VariablesDeclaration> := ID <IDList> <OptAsignment> S_SEMICOLON

<IDList> := S_COMMA ID <IDList> | ε

<OptAsignment> := S_EQUALS <Expression> | ε

<AssignmentOrCallSentence> := ID <OptAorCall> S_SEMICOLON

<OptAorCall> := S_EQUALS <Expression>

| S_Open_Parenthesis <ParameterListCall> S_Close_Parenthesis

<PrintSentence> := WR_CONSOLE S_POINT WR_WRITE S_Open_Parenthesis <Impression> S_Close_Parenthesis S_SEMICOLON

<Impression> :=

<Expression>

| ε

<IfElseSentence> :=

WR_IF S_OPEN_PARENTHESIS <Expression> S_CLOSE_PARENTHESIS S_OPEN_KEY

<SentencesList>

S_CLOSE_KEY <OptElse>

<OptElse> :=

| WR_ELSE <ElseIfOpt>

| ε

<ElseIfOpt> :=

| S_OPEN_KEY

<SentencesList>

S_CLOSE_KEY

| WR_IF S_OPEN_PARENTHESIS <Expression> S_CLOSE_PARENTHESIS S_OPEN_KEY

<SentencesList>

S_CLOSE_KEY <OptElse>

<SwitchSentence> :=

WR_SWITCH S_Open_Parenthesis <Expression> S_Close_Parenthesis S_Open_Key

<CaseList>

<OptDefault>

S_Close_Key

<CaseList> :=

WR_CASE <Expression> S_TWO_POINTS

<SentencesList>

<SentencesListSwitch>

<CaseList>

| ε

<OptDefault> :=

WR_DEFAULT S_TWO_POINTS

<SentencesList>

<SentencesListSwitch>

| ε

<ForSentence> :=

WR_FOR S_Open_Parenthesis <OptType> <AssignmentFor> <Expression> S_SEMICOLON <Expression> <OptIncDec> S_Close_Parenthesis S_Open_Key

<SentencesList>

<SentencesListLoops>

S_Close_Key

<OptType> := <Type> | ε

<AssignmentFor> := ID S_EQUALS <Expression> S_SEMICOLON

<OptIncDec> := S_INCREMENT | S_DECREMENT | ε

<WhileSentence> :=

WR_WHILE S_Open_Parenthesis <Expression> S_Close_Parenthesis S_Open_Key

<SentencesList>

<SentencesListLoops>

S_Close_Key

<DoWhileSentence> :=

WR_DO S_Open_Key

<SentencesList>

<SentencesListLoops>

S_Close_Key WR_WHILE S_Open_Parenthesis <Expression> S_Close_Parenthesis S_SEMICOLON

<ParameterListCall> := <Expression> <PList> | ε

<PList> := S_COMMA <Expression> <PList> | ε

<Expression> := <OptNot> <E> <OptComparisonSymbol> <AndOrOpt>

<OptNot> :=

S_NOT <OptNot>

| ε

<AndOrOpt> :=

S_AND <Expression>

| S_OR <Expression>

| ε

<OptComparisonSymbol> :=

S_EQUALS_EQUALS <E>

| S_MAJOR <E>

| S_LESS <E>

| S_MAJOR_EQUALS <E>

| S_LESS_EQUALS <E>

| S_DIFFERENT <E>

| ε

<E> := <T> <EP>

<EP> :=

S_PLUS<T><EP>

| S_MINUS<T><EP>

| ε

<T> := <F><TP>

<TP> :=

S_PRODUCT <F> <TP>

| S_DIVISION <F> <TP>

| ε

<F> :=

<OptNot> <FF>

<FF> :=

INTEGER

| DECIMAL

| STRING

| ID <OptUseFunction>

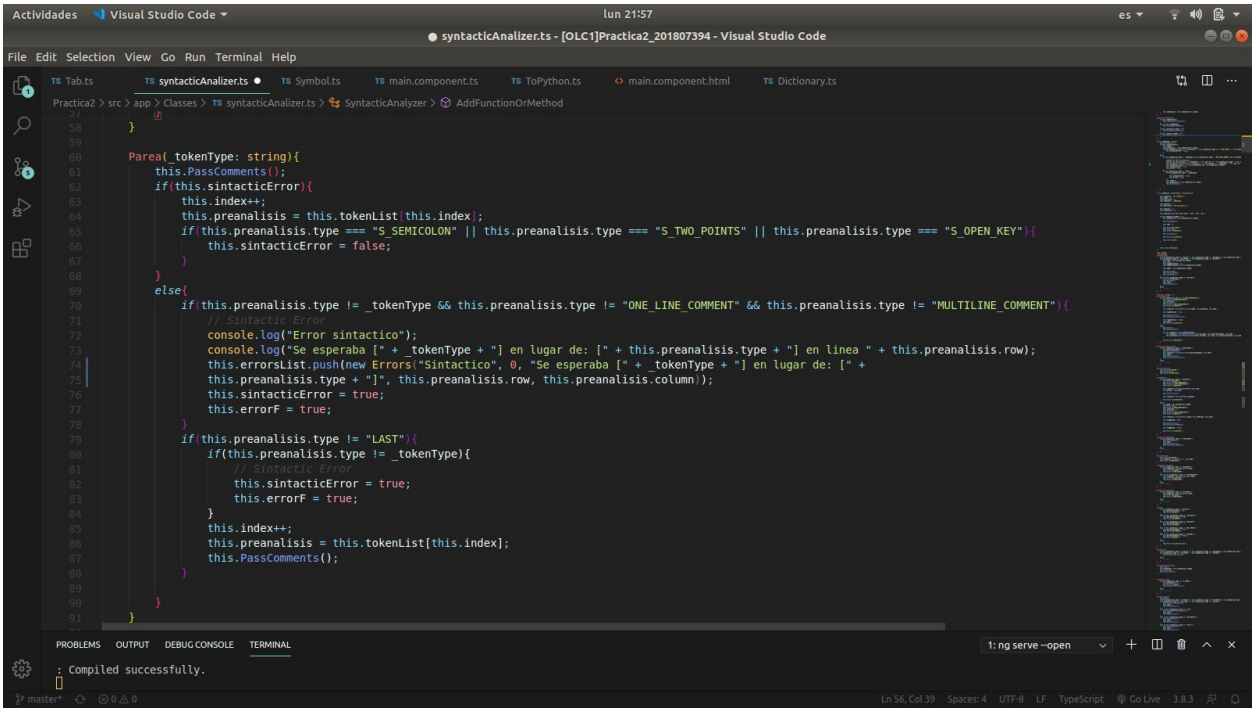
| WR_TRUE

| WR_FALSE

| S_OPEN_PARENTHESIS <E> S_CLOSE_PARENTHESIS

<OptUseFunction> := S_OPEN_PARENTHESIS <ParameterListCall> S_CLOSE_PARENTHESIS | ε

Se implemento de manera que cada no terminar sea una llamada a una función y cada terminal una llamada al método Parea con parámetro el token esperado. El método Parea se encuentra a continuación:


Actividades
Visual Studio Code
lun 21:57
es
syntacticAnalizer.ts - [OLC1]Practica2_201807394 - Visual Studio Code
File Edit Selection View Go Run Terminal Help
Tab.ts
syntacticAnalizer.ts
Symbol.ts
main.component.ts
ToPython.ts
main.component.html
Dictionary.ts
Practica2 > src > app > Classes > syntacticAnalizer.ts > SyntacticAnalyzer > AddFunctionOrMethod
}
Parea(_tokenType: string){
this.PassComments();
if(this.sintacticError){
this.index++;
this.preanalisis = this.tokenList[this.index];
if(this.preanalisis.type === "S_SEMICOLON" || this.preanalisis.type === "S_TWO_POINTS" || this.preanalisis.type === "S_OPEN_KEY"){
this.sintacticError = false;
}
}
else{
if(this.preanalisis.type != _tokenType && this.preanalisis.type != "ONE_LINE_COMMENT" && this.preanalisis.type != "MULTILINE_COMMENT"){
// Sintactic Error
console.log("Error sintactico");
console.log("Se esperaba [" + _tokenType + "] en lugar de: [" + this.preanalisis.type + "] en linea " + this.preanalisis.row);
this.errorsList.push(new Errors("Sintactico", 0, "Se esperaba [" + _tokenType + "] en lugar de: [" +
this.preanalisis.type + "]", this.preanalisis.row, this.preanalisis.column));
this.sintacticError = true;
this.errorF = true;
}
if(this.preanalisis.type != "LAST"){
if(this.preanalisis.type != _tokenType){
// Sintactic Error
this.sintacticError = true;
this.errorF = true;
}
this.index++;
this.preanalisis = this.tokenList[this.index];
this.PassComments();
}
}
}
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
1: ng serve --open
: Compiled successfully.
master*
Ln 56, Col 39
Spaces: 4
UTF-8
LF
TypeScript
Go Live
3.8.3