*Институт Транспорта и связи*



**Лабораторная работа 1**

По дисциплине

«Конструирование Компиляторов»

Тема: **Программирование Лексического Анализатора**

Студент: Денис Кобелев

Группа: 4401BD

Рига

2017 г.

1. **Теоретическое обоснование**

***Лексический анализатор*** (сканер) – часть компилятора, выполняющая просмотр исходного текста, распознавание и обработку лексем.

***Лексема*** – неделимая фундаментальная единица языка программирования (ключевое cлово, имя переменной, оператор).

Основные функции сканера:

* Чтение (сканирование) символов текста программы;
* Распознавание и выделение так называемых единиц синтаксических классов – лексем;
* Построение таблицы идентификаторов с учетом вложенности блоков и процедур;
* Построение таблицы констант;
* Исключение из текста программы комментариев;
* Замещение в памяти строк и тел процедур, представленных машинным кодом или кодом ассемблера;
* Обработка макрокоманд (генерация макрорасширений);
* Формирование файла диагностики лексических ошибок;
* Нумерация строк программы;
* Формирование различных метрик программы

1. **Текст заданного фрагмента индивидуального задания**

procedure TIndicator.SetValue(ALocation: TPoint; AModified: Boolean);  
begin  
case Operator of  
            '+': SetDisplay(Operand + R);  
            '-': SetDisplay(Operand - R);  
            '\*': SetDisplay(Operand \* R);  
            '/': if R = 0 then Error else SetDisplay(Operand / R);  
end;  
  if (Longint(Location) <> Longint(ALocation)) or  
    (Modified <> AModified) then Location := ALocation;  
    Modified := AModified;  
    DrawView;  
end

1. **Грамматические правила для имеющихся в сканируемом фрагменте программы лексем**

<ID> ::= letter | <ID> letter

<DELIM> ::= ( | ) | + | - | \* | / | ; | . | : | = | ‘

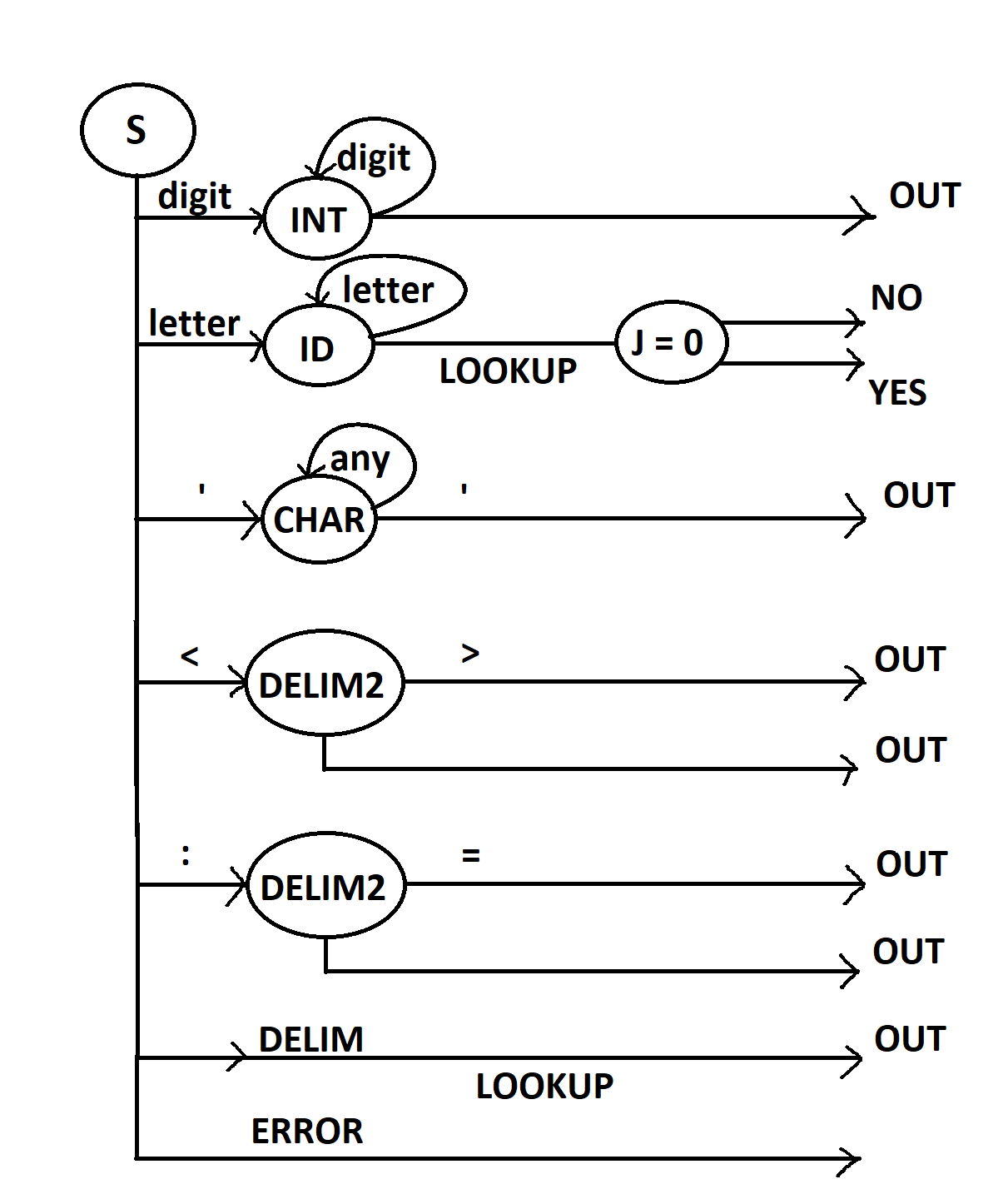
<COLON> ::= :

<LESS> ::= <

<DELIM2> ::= <COLON>= | <LESS>>

<INT> ::= digit | <INT> digit

1. **Диаграмму состояний соответствующего конечного автомата**



1. **Текст программы сканера**

class LexemeProcessor {  
 var delimiters: Set<Char> = *emptySet*()  
 var pushResult: (Result) -> Unit = **{}** fun process(char: Char) {  
 when(state){  
 START -> start(char)  
 INT -> processInt(char)  
 ID -> processId(char)  
 CONST -> processConst(char)  
 DELIM2\_ASSIGN -> processDelim2(char, '=')  
 DELIM2\_NE -> processDelim2(char, '>')  
 }  
 }  
  
 private var lexeme: String = ""  
  
 private var state: State = START  
  
 private fun isDelimiter(char: Char): Boolean = delimiters.contains(char)  
  
 private fun start(char: Char) {  
 lexeme = "" + char  
 state = when {  
 char.*isDigit*() -> INT  
 char.*isLetter*() -> ID  
 char == '\'' -> CONST  
 char == ':' -> DELIM2\_ASSIGN  
 char == '<' -> DELIM2\_NE  
 isDelimiter(char) -> return putResult("DELIM")  
 char.*isWhitespace*() -> return  
 else -> throw IllegalLexemeCharacter(char)  
  
 }  
 }  
  
 private fun processInt(char: Char) {  
 when {  
 char.*isDigit*() -> lexeme += char  
 else -> {putResult("LITERAL"); process(char) }  
 }  
 }  
  
 private fun processId(char: Char) {  
 when {  
 char.*isLetter*() -> lexeme += char  
 else -> {putResult("ID"); process(char) }  
  
 }  
 }  
  
 private fun processConst(char: Char) {  
 lexeme += char  
 if (char == '\'')  
 putResult("LITERAL")  
 }  
  
 fun processDelim2(char: Char, expected: Char) {  
 when (char) {  
 expected -> {lexeme += char; putResult("DELIM")}  
 else -> {putResult("DELIM"); process(char)}  
 }  
 }  
  
 fun putResult(type: String) {  
 pushResult(Result(lexeme, type))  
 state = START  
 }  
  
 class IllegalLexemeCharacter(char: Char) : Throwable("Unexpected character: " + char)  
 data class Result(val lexeme: String, val type: String)  
 enum class State {  
 START, INT, ID, CONST, DELIM2\_ASSIGN, DELIM2\_NE  
 }  
}

class LexicalAnalyser {  
 var literals: Set<String> = *emptySet*()  
 var identifiers: Set<String> = *emptySet*()  
 var keyWords: Set<String> = *emptySet*()  
 var results: List<LexicalEntry> = *emptyList*()  
  
 private var lexemeProcessor: LexemeProcessor = LexemeProcessor()  
  
 init {  
 lexemeProcessor.pushResult = this::processResult  
 }  
  
  
 fun analyse(input: String) {  
 for ((i, char) in input.*withIndex*()) {  
 try {  
 lexemeProcessor.process(char)  
 } catch (e: LexemeProcessor.IllegalLexemeCharacter) {  
 val analysedPart = input.subSequence(0, i)  
 val split = analysedPart.*split*("\r\n")  
 val line = split.size  
 val position = split.*last*().length  
 throw AnalisationException(e.message + " at " + line + ":" + position)  
 }  
 }  
 lexemeProcessor.process(' ')  
 }  
  
 fun loadKeyWords(keyWords: Set<String>){  
 this.keyWords = keyWords  
 }  
  
 fun loadDelimiters(delimiters: Set<Char>){  
 lexemeProcessor.delimiters = delimiters  
 }  
  
 fun clean() {  
 literals = *emptySet*()  
 identifiers = *emptySet*()  
 results = *emptyList*()  
 }  
  
 private fun processResult(result: LexemeProcessor.Result) {  
 var pointer = 0  
 var (lexeme, type) = result  
 if (type == "ID") {  
 type = getIdType(lexeme)  
 }  
  
 val wordSet = when(type) {  
 "ID" -> {identifiers += lexeme; identifiers}  
 "KEYWORD" -> keyWords  
 "DELIM" -> keyWords  
 "LITERAL" -> {literals += lexeme; literals}  
 else -> throw InvalidLexemeTypeException(type)  
 }  
 pointer = wordSet.*indexOf*(lexeme)  
  
 results += LexicalEntry(lexeme, type, pointer)  
 }  
  
 private fun isKeyWord(lexeme: String): Boolean = keyWords.contains(lexeme)  
 private fun getIdType(lexeme: String): String = if (isKeyWord(lexeme)) "KEYWORD" else "ID"  
  
 data class LexicalEntry(val lexeme: String, val type: String, val pointer: Int) {  
 override fun toString(): String = String.*format*("%10s | %7s | %3d", lexeme, type, pointer)  
 }  
 class InvalidLexemeTypeException(type: String) : Throwable("Unrecognized lexeme type: " + type)  
 class AnalisationException(message: String) : Throwable(message)  
}

1. **Таблица распознанных лексем с указанием собственно лексемы, типа лексемы и её кодового представления**

Result Table:   
 0: procedure | KEYWORD | 0  
 1: TIndicator | ID | 0  
 2: . | DELIM | 16  
 3: SetValue | ID | 1  
 4: ( | DELIM | 12  
 5: ALocation | ID | 2  
 6: : | DELIM | 14  
 7: TPoint | ID | 3  
 8: ; | DELIM | 15  
 9: AModified | ID | 4  
 10: : | DELIM | 14  
 11: Boolean | ID | 5  
 12: ) | DELIM | 13  
 13: ; | DELIM | 15  
 14: begin | KEYWORD | 1  
 15: case | KEYWORD | 2  
 16: Operator | ID | 6  
 17: of | KEYWORD | 3  
 18: '+' | LITERAL | 0  
 19: : | DELIM | 14  
 20: SetDisplay | ID | 7  
 21: ( | DELIM | 12  
 22: Operand | ID | 8  
 23: + | DELIM | 17  
 24: R | ID | 9  
 25: ) | DELIM | 13  
 26: ; | DELIM | 15  
 27: '-' | LITERAL | 1  
 28: : | DELIM | 14  
 29: SetDisplay | ID | 7  
 30: ( | DELIM | 12  
 31: Operand | ID | 8  
 32: - | DELIM | 19  
 33: R | ID | 9  
 34: ) | DELIM | 13  
 35: ; | DELIM | 15  
 36: '\*' | LITERAL | 2  
 37: : | DELIM | 14  
 38: SetDisplay | ID | 7  
 39: ( | DELIM | 12  
 40: Operand | ID | 8  
 41: \* | DELIM | 18  
 42: R | ID | 9  
 43: ) | DELIM | 13  
 44: ; | DELIM | 15  
 45: '/' | LITERAL | 3  
 46: : | DELIM | 14  
 47: if | KEYWORD | 5  
 48: R | ID | 9  
 49: = | DELIM | 21  
 50: 0 | LITERAL | 4  
 51: then | KEYWORD | 7  
 52: Error | ID | 10  
 53: else | KEYWORD | 8  
 54: SetDisplay | ID | 7  
 55: ( | DELIM | 12  
 56: Operand | ID | 8  
 57: / | DELIM | 20  
 58: R | ID | 9  
 59: ) | DELIM | 13  
 60: ; | DELIM | 15  
 61: end | KEYWORD | 4  
 62: ; | DELIM | 15  
 63: if | KEYWORD | 5  
 64: ( | DELIM | 12  
 65: Longint | ID | 11  
 66: ( | DELIM | 12  
 67: Loc | ID | 12

1. **Отпечатанный диагностический файл для исходного программного фрагмента**
   1. **Без лексических ошибок**

Key Words:   
 0: procedure  
 1: begin  
 2: case  
 3: of  
 4: end  
 5: if  
 6: or  
 7: then  
 8: else  
 9: boolean  
 10: <>  
 11: :=  
 12: (  
 13: )  
 14: :  
 15: ;  
 16: .  
 17: +  
 18: \*  
 19: -  
 20: /  
 21: =  
 22: '

Literals:   
 0: '+'  
 1: '-'  
 2: '\*'  
 3: '/'  
 4: 0

Identifiers:   
 0: TIndicator  
 1: SetValue  
 2: ALocation  
 3: TPoint  
 4: AModified  
 5: Boolean  
 6: Operator  
 7: SetDisplay  
 8: Operand  
 9: R  
 10: Error  
 11: Longint  
 12: Location  
 13: Modified  
 14: DrawView  
  
Result Table:   
 0: procedure | KEYWORD | 0  
 1: TIndicator | ID | 0  
 2: . | DELIM | 16  
 3: SetValue | ID | 1  
 4: ( | DELIM | 12  
 5: ALocation | ID | 2  
 6: : | DELIM | 14  
 7: TPoint | ID | 3  
 8: ; | DELIM | 15  
 9: AModified | ID | 4  
 10: : | DELIM | 14  
 11: Boolean | ID | 5  
 12: ) | DELIM | 13  
 13: ; | DELIM | 15  
 14: begin | KEYWORD | 1  
 15: case | KEYWORD | 2  
 16: Operator | ID | 6  
 17: of | KEYWORD | 3  
 18: '+' | LITERAL | 0  
 19: : | DELIM | 14  
 20: SetDisplay | ID | 7  
 21: ( | DELIM | 12  
 22: Operand | ID | 8  
 23: + | DELIM | 17  
 24: R | ID | 9  
 25: ) | DELIM | 13  
 26: ; | DELIM | 15  
 27: '-' | LITERAL | 1  
 28: : | DELIM | 14  
 29: SetDisplay | ID | 7  
 30: ( | DELIM | 12  
 31: Operand | ID | 8  
 32: - | DELIM | 19  
 33: R | ID | 9  
 34: ) | DELIM | 13  
 35: ; | DELIM | 15  
 36: '\*' | LITERAL | 2  
 37: : | DELIM | 14  
 38: SetDisplay | ID | 7  
 39: ( | DELIM | 12  
 40: Operand | ID | 8  
 41: \* | DELIM | 18  
 42: R | ID | 9  
 43: ) | DELIM | 13  
 44: ; | DELIM | 15  
 45: '/' | LITERAL | 3  
 46: : | DELIM | 14  
 47: if | KEYWORD | 5  
 48: R | ID | 9  
 49: = | DELIM | 21  
 50: 0 | LITERAL | 4  
 51: then | KEYWORD | 7  
 52: Error | ID | 10  
 53: else | KEYWORD | 8  
 54: SetDisplay | ID | 7  
 55: ( | DELIM | 12  
 56: Operand | ID | 8  
 57: / | DELIM | 20  
 58: R | ID | 9  
 59: ) | DELIM | 13  
 60: ; | DELIM | 15  
 61: end | KEYWORD | 4  
 62: ; | DELIM | 15  
 63: if | KEYWORD | 5  
 64: ( | DELIM | 12  
 65: Longint | ID | 11  
 66: ( | DELIM | 12  
 67: Location | ID | 12  
 68: ) | DELIM | 13  
 69: <> | DELIM | 10  
 70: Longint | ID | 11  
 71: ( | DELIM | 12  
 72: ALocation | ID | 2  
 73: ) | DELIM | 13  
 74: ) | DELIM | 13  
 75: or | KEYWORD | 6  
 76: ( | DELIM | 12  
 77: Modified | ID | 13  
 78: <> | DELIM | 10  
 79: AModified | ID | 4  
 80: ) | DELIM | 13  
 81: then | KEYWORD | 7  
 82: Location | ID | 12  
 83: := | DELIM | 11  
 84: ALocation | ID | 2  
 85: ; | DELIM | 15  
 86: Modified | ID | 13  
 87: := | DELIM | 11  
 88: AModified | ID | 4  
 89: ; | DELIM | 15  
 90: DrawView | ID | 14  
 91: ; | DELIM | 15  
 92: end | KEYWORD | 4

* 1. **С сознательно допущенными лексическими ошибками**

Unexpected character: ? at 9:17  
  
Key Words:   
 0: procedure  
 1: begin  
 2: case  
 3: of  
 4: end  
 5: if  
 6: or  
 7: then  
 8: else  
 9: boolean  
 10: <>  
 11: :=  
 12: (  
 13: )  
 14: :  
 15: ;  
 16: .  
 17: +  
 18: \*  
 19: -  
 20: /  
 21: =  
 22: '  
  
Literals:   
 0: '+'  
 1: '-'  
 2: '\*'  
 3: '/'  
 4: 0  
  
Identifiers:   
 0: TIndicator  
 1: SetValue  
 2: ALocation  
 3: TPoint  
 4: AModified  
 5: Boolean  
 6: Operator  
 7: SetDisplay  
 8: Operand  
 9: R  
 10: Error  
 11: Longint  
 12: Loc  
  
Result Table:   
 0: procedure | KEYWORD | 0  
 1: TIndicator | ID | 0  
 2: . | DELIM | 16  
 3: SetValue | ID | 1  
 4: ( | DELIM | 12  
 5: ALocation | ID | 2  
 6: : | DELIM | 14  
 7: TPoint | ID | 3  
 8: ; | DELIM | 15  
 9: AModified | ID | 4  
 10: : | DELIM | 14  
 11: Boolean | ID | 5  
 12: ) | DELIM | 13  
 13: ; | DELIM | 15  
 14: begin | KEYWORD | 1  
 15: case | KEYWORD | 2  
 16: Operator | ID | 6  
 17: of | KEYWORD | 3  
 18: '+' | LITERAL | 0  
 19: : | DELIM | 14  
 20: SetDisplay | ID | 7  
 21: ( | DELIM | 12  
 22: Operand | ID | 8  
 23: + | DELIM | 17  
 24: R | ID | 9  
 25: ) | DELIM | 13  
 26: ; | DELIM | 15  
 27: '-' | LITERAL | 1  
 28: : | DELIM | 14  
 29: SetDisplay | ID | 7  
 30: ( | DELIM | 12  
 31: Operand | ID | 8  
 32: - | DELIM | 19  
 33: R | ID | 9  
 34: ) | DELIM | 13  
 35: ; | DELIM | 15  
 36: '\*' | LITERAL | 2  
 37: : | DELIM | 14  
 38: SetDisplay | ID | 7  
 39: ( | DELIM | 12  
 40: Operand | ID | 8  
 41: \* | DELIM | 18  
 42: R | ID | 9  
 43: ) | DELIM | 13  
 44: ; | DELIM | 15  
 45: '/' | LITERAL | 3  
 46: : | DELIM | 14  
 47: if | KEYWORD | 5  
 48: R | ID | 9  
 49: = | DELIM | 21  
 50: 0 | LITERAL | 4  
 51: then | KEYWORD | 7  
 52: Error | ID | 10  
 53: else | KEYWORD | 8  
 54: SetDisplay | ID | 7  
 55: ( | DELIM | 12  
 56: Operand | ID | 8  
 57: / | DELIM | 20  
 58: R | ID | 9  
 59: ) | DELIM | 13  
 60: ; | DELIM | 15  
 61: end | KEYWORD | 4  
 62: ; | DELIM | 15  
 63: if | KEYWORD | 5  
 64: ( | DELIM | 12  
 65: Longint | ID | 11  
 66: ( | DELIM | 12  
 67: Loc | ID | 12

1. **Выводы**

В ходе выполнения работы было необходимо разработать программу выполняющую роль лексического анализатора для предоставленного примера на языке Pascal.

В результате получилась программа способная анализировать лексику данного конкретного примера и выдавать сообщения об ошибке в тексте. Программа была написана на языке Kotlin из-за личных предпочтений автора. Весь вывод и подача входных данных (а именно текста программы) выполняются в отдельных текстовом файлах.