Министерство образования и науки Российской Федерации
Федеральное государственное автономное учреждение высшего образования
«Севастопольский государственный университет»

Кафедра информационных
технологий и
компьютерных систем

ОТЧЁТ
о выполнении лабораторной работы № 3
на тему: «Синтаксический анализ методом LL(1)»
по дисциплине «Системное программное обеспечение»

Выполнил:
студент гр. ИВТб-32о
Демиденко А.А.
Проверил:
Фисун С.Н.

Севастополь
2015

Содержание

Цель работы.....3
Постановка задачи.....3
Ход работы.....3
Преобразование грамматики.....3
Тестовый пример.....5
Код программы.....5
Вывод.....9

Цель работы

Изучение методов синтаксического анализа, получение навыков в программировании алгоритма синтаксического разбора методом LL(1).

Постановка задачи

Для грамматики заданного языка PLO разработать алгоритм и программу, выполняющую синтаксический разбор входной цепочки, заданной в виде строки, по методу LL(1).

Ход работы

язык PLO-2:

<оператор> [<OP>] <идентификатор> [I]
<выражение> [E]
<условие> [C]
CALL [c]
BEGIN [b]
END [e]
IF [i]
THEN [t]
WHILE [w]
DO [d]
; [;]
= [=]

Преобразование грамматики

1) <OP> → <V>=E | c<PN> | b<OL>e | iCt<OP> | wCd<OP> | empty
2) <V>→ I
3) <PN> → I
4) <OL> → <OL>;<OP> | <OP>

Данная грамматика не отвечает требованиям, предъявляемым грамматике для которой можно разрабатывать алгоритм синтаксического разбора по методу LL(1), т.к. в правиле <OL> присутствует левая факторизация. Изменив правило <OL>, и добавив правило <X>, а также выделив каждый альтернативный вывод в новом правиле получили грамматику соответствующего вида:

1) <OP> → <V>=E

2) <OP> → c<PN>

3) <OP> → b<OL>e

4) <OP> → iCt<OP>

5) <OP> → wCd<OP>

6) <OP> → empty

7) <V>→ I

8) <PN> → I

9) <OL> → <OP><X>

10) <X> → ;<OP><X>

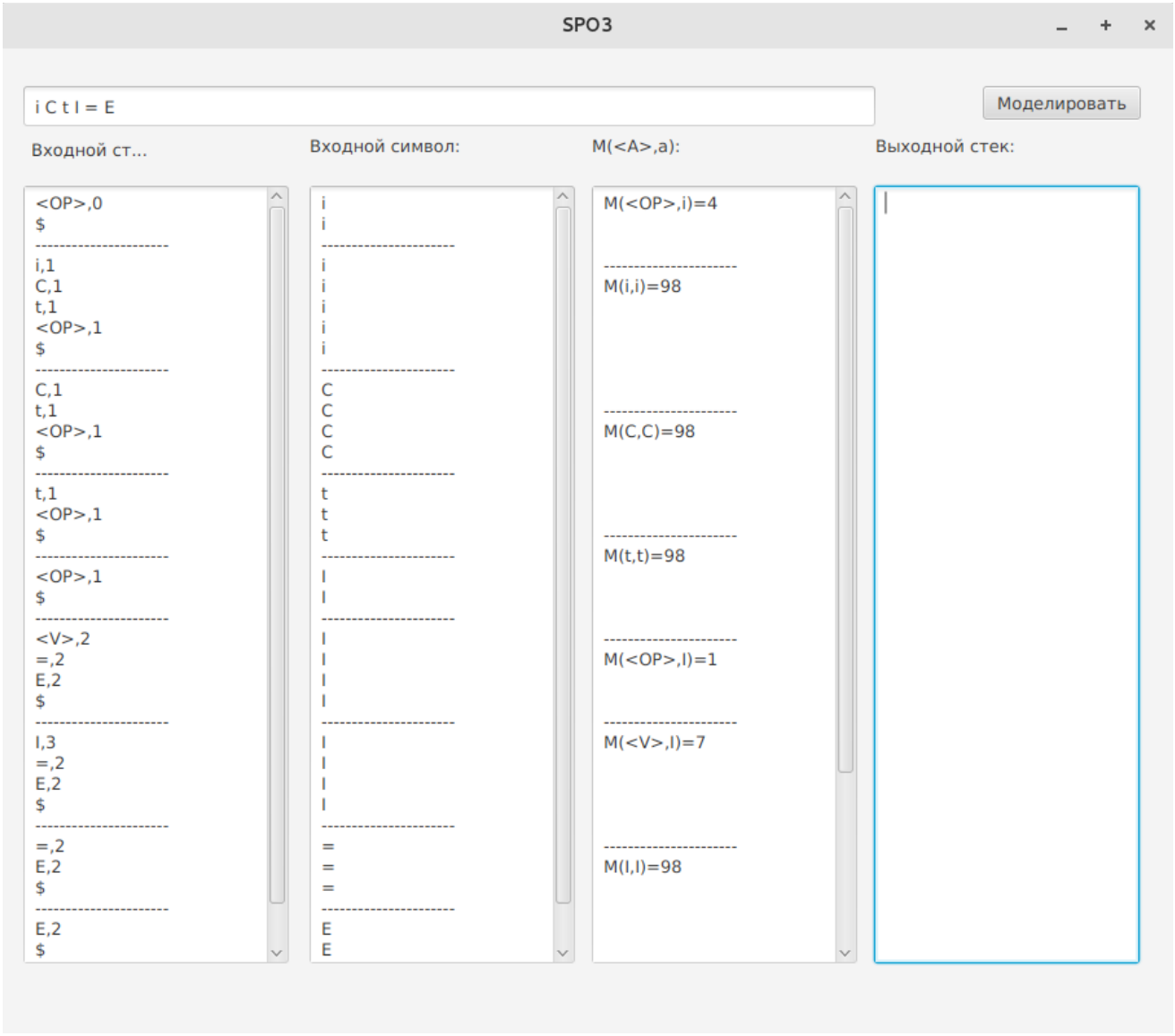
11) <X> → empty

В соответствии с полученной грамматикой была разработана управляющая таблица:

	l	**E**	**C**	**c**	**b**	**e**	**i**	**t**	**w**	**d**	**;**	**=**	**$**
<OP>	1			2	3		4		5				
<V>	7												
<PN>	8												
<OL>	9			9	9		9		9				
<X>											10		
l	c												
E		c											
C			c										
c				c									
b					c								
e						c							
i							c						
t								c					
w									c				
d										c			
;											c		
=												c	
$													д

На основании данной управляющей таблицы был разработан алгоритм и написана программа (см. приложение).

Тестовый пример



Код программы

```
package com.spo.lab3;

import java.net.URL;
import java.util.ResourceBundle;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.control.TextArea;
import javafx.scene.control.TextField;
import java.util.Stack;
import javafx.scene.control.Alert;
import javafx.scene.control.TreeView;

public class GUIController implements Initializable {
	@FXML
	private TextField inputString;
	@FXML
	private TextArea inputChar;
	@FXML
	private TextArea outputStackArea;
	@FXML
	private TextArea inputStackArea;
	@FXML
	private TextArea M;
	@FXML
```

```
	private TreeView outputTree;

	String outputStackGlobal = "";
	int[][] workTable = { { 1, 0, 0, 2, 3, 0, 4, 0, 5, 0, 0, 0, 0 }, { 7, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
			{ 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 }, { 9, 0, 0, 9, 9, 0, 9,
0, 9, 0, 0, 0, 0 },
			{ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 10, 0, 0 }, { 98, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0 },
			{ 0, 98, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 }, { 0, 0, 98, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0 },
			{ 0, 0, 0, 98, 0, 0, 0, 0, 0, 0, 0, 0, 0 }, { 0, 0, 0, 0, 98, 0,
0, 0, 0, 0, 0, 0, 0 },
			{ 0, 0, 0, 0, 0, 98, 0, 0, 0, 0, 0, 0, 0 }, { 0, 0, 0, 0, 0, 0,
98, 0, 0, 0, 0, 0, 0 },
			{ 0, 0, 0, 0, 0, 0, 0, 98, 0, 0, 0, 0, 0 }, { 0, 0, 0, 0, 0, 0,
0, 0, 98, 0, 0, 0, 0 },
			{ 0, 0, 0, 0, 0, 0, 0, 0, 0, 98, 0, 0, 0 }, { 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 98, 0, 0 },
			{ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 98, 0 }, { 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 99 } };

	String workStr;
	String workChar[];
	Stack<String> outputStack = new Stack();
	Stack<String> inputStack = new Stack();

	@FXML
	private void start() {
		M.clear();
		inputStackArea.clear();
		outputStackArea.clear();
		outputStackGlobal = "";
		inputChar.clear();
		workStr = inputString.getText();
		workChar = workStr.split("\\s+");
		inputStack.push("$");
		// inputStack.push("<OP>");
		int i = 0;
		M("<OP>", workChar);
		outputStackArea.setText(outputStackGlobal);

	}

	public String[] grammar(int n) {
		String arg[];
		switch (n) {
		case 1: {
			arg = new String[3];
			arg[0] = "<V>";
			arg[1] = "=";
			arg[2] = "E";
			return arg;
		}
		case 2: {
			arg = new String[2];
			arg[0] = "c";
			arg[1] = "<PN>";
			return arg;
		}
		case 3: {
			arg = new String[3];
			arg[0] = "b";
			arg[1] = "<OL>";
			arg[2] = "e";
			return arg;
		}
		case 4: {
			arg = new String[4];
```

```
			arg[0] = "i";
			arg[1] = "C";
			arg[2] = "t";
			arg[3] = "<OP>";
			return arg;
		}
		case 5: {
			arg = new String[4];
			arg[0] = "w";
			arg[1] = "C";
			arg[2] = "d";
			arg[3] = "<OP>";
			return arg;
		}
		case 6: {
			arg = new String[0];
			return arg;
		}
		case 7: {
			arg = new String[1];
			arg[0] = "I";
			return arg;
		}
		case 8: {
			arg = new String[1];
			arg[0] = "I";
			return arg;
		}
		case 9: {
			arg = new String[2];
			arg[0] = "<OP>";
			arg[1] = "<X>";
			return arg;
		}
		case 10: {
			arg = new String[3];
			arg[0] = ";";
			arg[1] = "<OP>";
			arg[2] = "<X>";
			return arg;
		}
		case 11: {
			arg = new String[0];
			return arg;
		}
		}
		String s[] = { "ex" };
		return s;
	}

	public int whatString(String pop) {
		switch (pop) {
		case "<OP>": return 0;
		case "<V>": return 1;
		case "<PV>": return 2;
		case "<OL>": return 3;
		case "<X>": return 4;
		case "I": return 5;
		case "E": return 6;
		case "C": return 7;
		case "c": return 8;
		case "b": return 9;
		case "e": return 10;
		case "i": return 11;
		case "t": return 12;
		case "w": return 13;
		case "d": return 14;
		case ";": return 15;
		case "=": return 16;
```

```
            case "$": return 17;
            }
            return 404;
      }

      public int whatTab(String ch) {
            switch (ch) {
            case "I": return 0;
            case "E": return 1;
            case "C": return 2;
            case "c": return 3;
            case "b": return 4;
            case "e": return 5;
            case "i": return 6;
            case "t": return 7;
            case "w": return 8;
            case "d": return 9;
            case ";": return 10;
            case "=": return 11;
            case "$": return 12;
            }

            return 404;
      }

      @Override
      public void initialize(URL url, ResourceBundle rb) {
            // TODO
      }

      private void M(String pop, String ch[]) {
            int numPrav;
            int i = 0;
            int count = 1;
            outputStackGlobal += pop + ",0" + "\n";
            inputChar.setText(inputChar.getText() + ch[i] + "\n");
            inputStackArea.setText(inputStackArea.getText() + pop + ",0" + "\n");
            inputChar.setText(inputChar.getText() + ch[i] + "\n");
            inputStackArea.setText(inputStackArea.getText() + inputStack.peek() +
"\n");
            inputChar.setText(inputChar.getText() + "----------------------" +
"\n");
            inputStackArea.setText(inputStackArea.getText() +
"----------------------" + "\n");

            while (i < ch.length) {
                  if (whatTab(ch[i]) != 404) {
                        numPrav = workTable[whatString(pop)][whatTab(ch[i])];
                        M.setText(M.getText() + "M(" + pop + "," + ch[i] + ")=" +
numPrav + "\n");

                        for (int k = inputStack.size() - 1; k >= 0; k--)
                              M.setText(M.getText() + "\n");

                        M.setText(M.getText() + "\n" + "----------------------" +
"\n");
                        if (numPrav == 98) {
                              i++;

                              for (int k = inputStack.size() - 1; k >= 0; k--) {
                                    inputChar.setText(inputChar.getText() + ch[i]
+ "\n");

      inputStackArea.setText(inputStackArea.getText() + inputStack.elementAt(k) +
"\n");
                              }

                              inputChar.setText(inputChar.getText() +
"----------------------" + "\n");
```

```
                              inputStackArea.setText(inputStackArea.getText() +
"----------------------" + "\n");
                              outputStackGlobal += inputStack.peek() + "\n";
                              pop = inputStack.pop();
                              String popArg[] = pop.split(",+");
                              pop = popArg[0];
                              if (popArg.length == 2)
                                    count = Integer.parseInt(popArg[1]) + 1;

                        } else {
                              if (numPrav == 99)
                                    break;

                              String popS[] = grammar(numPrav);
                              for (int j = popS.length - 1; j >= 0; j--)
                                    inputStack.push(popS[j] + "," + count);

                              for (int k = inputStack.size() - 1; k >= 0; k--) {
                                    inputChar.setText(inputChar.getText() + ch[i]
+ "\n");

      inputStackArea.setText(inputStackArea.getText() + inputStack.elementAt(k) +
"\n");
                              }

                              inputChar.setText(inputChar.getText() +
"----------------------" + "\n");
                              inputStackArea.setText(inputStackArea.getText() +
"----------------------" + "\n");
                              pop = inputStack.pop();
                              outputStackGlobal += pop + "\n";
                              String popArg[] = pop.split(",+");
                              count++;
                              pop = popArg[0];
                        }
                  } else {
                        Main m = new Main();
                        Alert alert = new Alert(Alert.AlertType.WARNING);
                        alert.initOwner(m.getPrimaryStage());
                        alert.setTitle("Ошибка");
                        alert.setHeaderText("Ошибка в последовательности");
                        alert.setContentText("Пожалуйста, исправте
последовательность");

                        alert.showAndWait();
                        break;
                  }
            }
      }

}
```

Вывод

В ходе данной лабораторной работы был изучен LL(1) метод синтаксического анализа, получены навыки в программировании алгоритма для грамматики, заданного языка PLO. Разработаны алгоритм и программа, выполняющие синтаксический разбор входной цепочки, заданной в виде строки, по методу LL(1). Проверка работоспособности программы была произведена на тестовом примере, и дала положительные результаты