|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ Информатики и систем управления

КАФЕДРА Теоретической информатики и компьютерных технологий

**Лабораторная работа № 5**

**«Лексический распознаватель»**

***по курсу «Конструирование компиляторов»***

Студент *Ионов Т.Р. 61Б*

Преподаватель *Коновалов А.B.*

*Москва, 2022 г.*

СОДЕРЖАНИЕ

[1 Цель работы 3](#_Toc106224508)

[2 Индивидуальное задание 3](#_Toc106224509)

[3 Реализация 3](#_Toc106224510)

[4 Тестирование 4](#_Toc106224511)

[Вывод 4](#_Toc106224512)

# 1 Цель работы

Целью данной работы является изучение использования детерминированных конечных автоматов с размеченными заключительными состояниями (лексических распознавателей) для решения задачи лексического анализа.

# 2 Индивидуальное задание

Class, super, -, ->, строковые литералы ограничены двойными кавычками, могут содержаться escape-последовательности вида \x, x – любой символ, не могу пересекать границы строк текста.

3 Реализация

Листинг 1 − файл Automata.java.

package com.company;

import java.util.\*;

public class Automata {

public final static int[][] table = {

/\* s u p e r c l a - > " \ num ws a-z EOL UNK\*/

/\* START \*/{ 1, 10, 10, 10, 10, 6, 10, 10, 11, -1, 13, -1, 15, 16, 10, 16, -1},

/\* ID\_1 \*/{10, 2, 10, 10, 10, 10, 10, 10, -1, -1, -1, -1, 10, -1, 10, -1, -1},

/\* ID\_2 \*/{10, 10, 3, 10, 10, 10, 10, 10, -1, -1, -1, -1, 10, -1, 10, -1, -1},

/\* ID\_3 \*/{10, 10, 10, 4, 10, 10, 10, 10, -1, -1, -1, -1, 10, -1, 10, -1, -1},

/\* ID\_4 \*/{10, 10, 10, 10, 5, 10, 10, 10, -1, -1, -1, -1, 10, -1, 10, -1, -1},

/\* KEY\_5 \*/{10, 10, 10, 10, 10, 10, 10, 10, -1, -1, -1, -1, -1, -1, 10, -1, -1},

/\* ID\_6 \*/{10, 10, 10, 10, 10, 10, 7, 10, -1, -1, -1, -1, 10, -1, 10, -1, -1},

/\* ID\_7 \*/{10, 10, 10, 10, 10, 10, 10, 8, -1, -1, -1, -1, 10, -1, 10, -1, -1},

/\* ID\_8 \*/{ 9, 10, 10, 10, 10, 10, 10, 10, -1, -1, -1, -1, 10, -1, 10, -1, -1},

/\* ID\_9 \*/{ 5, 10, 10, 10, 10, 10, 10, 10, -1, -1, -1, -1, 10, -1, 10, -1, -1},

/\* ID\_10 \*/{10, 10, 10, 10, 10, 10, 10, 10, -1, -1, -1, -1, 10, -1, 10, -1, -1},

/\* OP\_11 \*/{-1, -1, -1, -1, -1, -1, -1, -1, -1, 12, -1, -1, -1, -1, -1, -1, -1},

/\* OP\_12 \*/{-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},

/\* LIT\_13 \*/{13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 14, 17, 13, 13, 13, -1, -1},

/\* LIT\_14 \*/{-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},

/\* NUM\_15 \*/{-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 15, -1, -1, -1, -1},

/\* WS\_16 \*/{-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 16, -1, 16, -1},

/\* ESC\_17 \*/{13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 16, -1},

};

private SortedMap<Position, String> messages;

private ArrayList<Token> tokens;

public HashMap<String, Integer> idents = new HashMap<>();

private int ident\_order = 0;

private String program;

private Position pos;

private int state;

public Automata(String program) {

this.program = program;

this.pos = new Position(program);

this.state = 0;

this.messages = new TreeMap<>();

this.tokens = new ArrayList<>();

}

public HashMap<String, Integer> getMap() {

return idents;

}

private int getCode(char c) {

/\* s u p e r c l a - > " num ws a-z\*/

switch (c) {

case 's':

return 0;

case 'u':

return 1;

case 'p':

return 2;

case 'e':

return 3;

case 'r':

return 4;

case 'c':

return 5;

case 'l':

return 6;

case 'a':

return 7;

case '-':

return 8;

case '>':

return 9;

case '\"':

return 10;

case '\n':

return 15;

case '\\':

return 11;

}

if (c >= '0' && c <= '9')

return 12;

if (c == ' ' || c == '\t' || c == '\r')

return 13;

if ((c >= 'a' && c <= 'z') || (c >= 'A' && c <= 'Z'))

return 14;

return 16;

}

private String getStateName(int state) {

return switch (state) {

case 1, 2, 3, 4, 6, 7, 8, 9, 10 -> "IDENT";

case 5 -> "KEYWORD";

case 11, 12 -> "OPERATION";

case 14 -> "STRING LITERAL";

case 15 -> "NUMBER";

case 16 -> "WHITESPACE";

default -> "ERROR";

};

}

public void run() {

while (!pos.isEOF()) {

StringBuilder word = new StringBuilder();

state = 0;

boolean finalState = false;

boolean errorState = false;

Position start = new Position(pos);

while (!pos.isEOF()) {

char curr\_char = program.charAt(pos.getIndex());

int jump\_code = getCode(curr\_char);

int next\_state = table[state][jump\_code];

if (-1 == next\_state) {

if (state == 0) {

errorState = true;

} else {

finalState = true;

}

break;

}

state = next\_state;

pos = pos.next();

word.append(curr\_char);

if (pos.isEOF() && curr\_char != '\\') {

finalState = true;

break;

}

}

if (finalState) {

Fragment frag = new Fragment(start, pos);

String value = word.toString().replaceAll("\n", "");

String domain = getStateName(state);

if (Objects.equals(domain, "IDENT")){

if (idents.containsKey(value)) {

int id = idents.get(value);

tokens.add(new Token(domain, String.valueOf(id), frag));

} else {

idents.put(value, ident\_order);

tokens.add(new Token(domain, String.valueOf(ident\_order), frag));

ident\_order++;

}

} else {

tokens.add(new Token(domain, value, frag));

}

continue;

}

if (errorState) {

messages.put(new Position(pos), "Unexpected char");

}

pos = pos.next();

}

Fragment frag = new Fragment(pos, pos);

tokens.add(new Token("EOF" , " " , frag));

}

public boolean hasNextToken() {

return !tokens.isEmpty();

}

public Token nextToken() {

return tokens.remove(0);

}

public void output\_messages() {

if (messages.isEmpty())

return;

System.out.println("\nMessages:");

for (Map.Entry<Position, String> entry : messages.entrySet()) {

System.out.print("ERROR ");

System.out.print("(" + entry.getKey().getLine() + ", " +

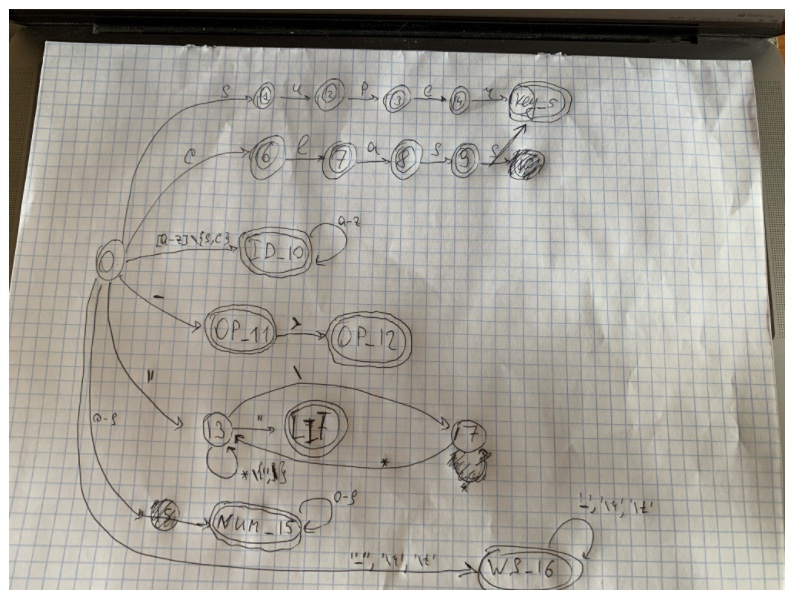
entry.getKey().getPos() + "): ");

System.out.println(entry.getValue());

}

}

}



# 4 Тестирование

Вход:

class

super

clasuper

"a\"defqwe"

1231

Результат:

KEYWORD (1, 1)-(1, 6): class

WHITESPACE (1, 6)-(2, 1):

KEYWORD (2, 1)-(2, 6): super

WHITESPACE (2, 6)-(3, 1):

IDENT (3, 1)-(3, 9): 0

WHITESPACE (3, 9)-(4, 1):

STRING LITERAL (4, 1)-(4, 12): "a\"defqwe"

WHITESPACE (4, 12)-(5, 1):

NUMBER (5, 1)-(5, 4): 123

EOF (5, 4)-(5, 4):

0 : clasuper

# Вывод

В результате выполнения данной лабораторной работы были изучены использования детерминированных конечных автоматов с размеченными заключительными состояниями (лексических распознавателей) для решения задачи лексического анализа.