**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**Чорноморський національний університет   
імені Петра Могили**

**Факультет комп’ютерних наук**

**Кафедра інженерії програмного забезпечення**

**ЛАБОРАТОРНА РОБОТА №3**

з дисципліни «Теорія автоматів та формальних мов»

на тему: **«**Предиктивний синтаксичний аналізатор**»**

121 – ЛР.ПЗ.03 – 308.1710809

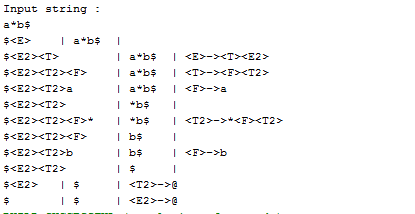
Виконав: студент 3 курсу, групи 308

спеціальності 121 «Інженерія програмного забезпечення», **Зурілов І.М.**

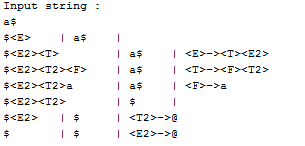
Керівник: професор, **Фісун М.Т.**

**м. Миколаїв – 2019 рік**

Код синтаксичного аналізатора наведено у додатку.



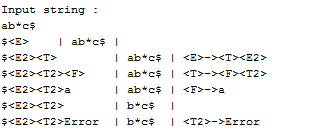
**Рис.1**



**Рис.2**



**Рис.3**



**Рис.4**

**Додаток**

package SyntaxAnalyzer;

import java.util.ArrayList;

import java.util.Collections;

import java.util.HashMap;

import java.util.Map;

import java.util.Scanner;

public class SyntaxAnalyzer {

private static Map<String, Map> nonterminal = new HashMap<>();

public static void main(String[] args) {

init();

Scanner scanner = new Scanner(System.in);

System.out.println("Input string :");

String text = scanner.nextLine();

analize(text);

}

private static void analize(String text) {

boolean flag = false;

ArrayList<Character> symbols = new ArrayList<>();

for (char c : text.toCharArray()) {

symbols.add(c);

}

ArrayList<String> stack = new ArrayList<>();

if (stack.isEmpty()) {

if (nonterminal.get("<E>").containsKey(String.valueOf(symbols.get(0)))){

stack.add("$");

stack.add("<E>");

print(stack, symbols, "");

} else {

stack.add("Error");

print(stack, symbols, "<E>");

stack.clear();

}

}

while (!stack.isEmpty()){

String last = stack.get(stack.size() - 1);

if(last.equals(String.valueOf(symbols.get(0))) && !last.equals("(")){

stack.remove(stack.lastIndexOf(last));

symbols.remove(0);

if (!stack.isEmpty()) {

print(stack, symbols, "");

}

} else if(nonterminal.containsKey(last) && nonterminal.get(last).containsKey(String.valueOf(symbols.get(0)))){

stack.remove(stack.lastIndexOf(last));

if (last.equals("<F>") && flag){

symbols.remove(0);

flag = false;

}

ArrayList<String> localNonterminal = ((ArrayList<String>) nonterminal.get(last).get(String.valueOf(symbols.get(0))));

ArrayList<String> temp = (ArrayList<String>) localNonterminal.clone();

Collections.reverse(temp);

if(!temp.contains("@")) {

stack.addAll(temp);

}

print(stack, symbols, last);

} else {

if(last.equals("(")){

stack.remove(last);

flag = true;

print(stack, symbols, "");

} else {

stack.add("Error");

print(stack, symbols, last);

stack.clear();

}

}

}

}

private static void init() {

Map<String, ArrayList> inputSymbol = new HashMap<>();

ArrayList<String> temp = new ArrayList<>();

temp.add("<T>");

temp.add("<E2>");

inputSymbol.put("a", temp);

inputSymbol.put("b", temp);

inputSymbol.put("c", temp);

inputSymbol.put("(", temp);

nonterminal.put("<E>", inputSymbol);

inputSymbol = new HashMap<String, ArrayList>();

temp = new ArrayList<String>();

temp.add("+");

temp.add("<T>");

temp.add("<E2>");

inputSymbol.put("+", temp);

temp = new ArrayList<String>();

temp.add("@");

inputSymbol.put(")", temp);

inputSymbol.put("$", temp);

nonterminal.put("<E2>", inputSymbol);

inputSymbol = new HashMap<String, ArrayList>();

temp = new ArrayList<String>();

temp.add("<F>");

temp.add("<T2>");

inputSymbol.put("a", temp);

inputSymbol.put("b", temp);

inputSymbol.put("c", temp);

inputSymbol.put("(", temp);

nonterminal.put("<T>", inputSymbol);

inputSymbol = new HashMap<String, ArrayList>();

temp = new ArrayList<String>();

temp.add("@");

inputSymbol.put("+", temp);

inputSymbol.put(")", temp);

inputSymbol.put("$", temp);

temp = new ArrayList<String>();

temp.add("\*");

temp.add("<F>");

temp.add("<T2>");

inputSymbol.put("\*", temp);

nonterminal.put("<T2>", inputSymbol);

inputSymbol = new HashMap<String, ArrayList>();

temp = new ArrayList<String>();

temp.add("a");

inputSymbol.put("a", temp);

temp = new ArrayList<String>();

temp.add("b");

inputSymbol.put("b", temp);

temp = new ArrayList<String>();

temp.add("c");

inputSymbol.put("c", temp);

temp = new ArrayList<String>();

temp.add("(");

temp.add("<E>");

temp.add(")");

inputSymbol.put("(", temp);

nonterminal.put("<F>", inputSymbol);

}

private static void print(ArrayList<String> stack, ArrayList<Character> symbols, String last) {

String st = "";

String sym = "";

String prod = last + "->";

for (String s : stack) {

st = st.concat(s);

}

for (char c : symbols) {

sym = sym.concat(String.valueOf(c));

}

if(nonterminal.containsKey(last) && nonterminal.get(last).containsKey(String.valueOf(symbols.get(0)))){

ArrayList<String> temp = (ArrayList<String>) nonterminal.get(last).get(String.valueOf(symbols.get(0)));

for (String s : temp) {

prod = prod.concat(s);

}

} else if (last.equals("")){

prod = "";

} else if(stack.contains("Error")){

prod = prod.concat("Error");

}

System.out.println(st + "\t| " + sym + "\t| " + prod);

}

}