Нижегородский государственный технический университет им. Р.Е. Алексеева.

Институт радиоэлектроники и информационных технологий.

Кафедра «Прикладная информатика и математика».

Отчёт по лабораторной работе:

«Построение конечного автомата по регулярной грамматике»

Выполнил: студент 2 курса группы 23-ПМ-1 Наплавков А.А.

Научный руководитель: Санников Н.А.

Нижний Новгород

2024

**Введение:**

В данной лабораторной работе требуется написать программу обладающую следующими функциями:

1)Ввод произвольной формальной грамматики с клавиатуры и проверка на принадлежность к классу регулярных грамматик;

2)Построение по заданной регулярной грамматике конечного автомата;

3)Преобразование недетерминированного конечного автомата к детерминированному конечному автомату;

4)Вывод графа результирующего конечного автомата на экран;

Так же необходимо сделать проверку работы программа используя следующую грамматику:

G=({S,C,D},{0,1},P,E), P:

1)S->1C|0D, 2)C->0D|0S; 3)D->1C|1S|0

**Реализация:**

Программа реализована на языке С++. Для хранения информации о переходах использовалось несколько векторов с разной структурой . В первом хранится то, откуда был произведён переход, в 2 с помощью какого символа, в 3 куда.

**Сам код**:

#include <iostream>

#include <vector>

#include <string>

#include <algorithm>

#include <utility>

using namespace std;

string gram1;

string gram2;

int counter;

vector <char> gramG;

vector <char> gramP;

pair <char, char> GS;

pair <pair<char, char>, char> P;

vector <pair<pair<char, char>, char>> GPS;

string alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

int main(){

cout << "Insert grammatic G like this: {K,L,M,N} \n";

cin >> gram1;

for (int i = 0; i <= gram1.size(); i++) {

if (gram1[i] != '{' and gram1[i] != '}' and gram1[i] != ',')

gramG.push\_back(gram1[i]);

}

cout << "Insert grammatic P like this: {a,b,+,-} \n";

cin >> gram2;

for (int i = 0; i <= gram2.size(); i++) {

if (gram2[i] != '{' and gram2[i] != '}' and gram2[i] != ',')

gramP.push\_back(gram2[i]);

}

cout << "Enter the number of clicks \n";

cin >> counter;

for (int i = 0; i <= counter-1; i++) {

cout << "Insert " << i + 1 << " letter of grammar, by using -> \n";

cin >> gram1;

if (gram1[1] != '-' || gram1[2] != '>')

{

cout << "incorrect insert \n ";

i--;

continue;

}

if (gram1.size() == 4) {

gramG.push\_back('@');

GS = make\_pair(gram1[0], '@');

P = make\_pair(GS, gram1[3]);

GPS.push\_back(P);

}

GS = make\_pair(gram1[0], gram1[4]);

P = make\_pair(GS, gram1[3]);

GPS.push\_back(P);

if (gram1[5] == '|' && (!(!gram1[6])) && (!gram1[7])) {

gramG.push\_back('@');

GS = make\_pair(gram1[0], '@');

P = make\_pair(GS, gram1[6]);

GPS.push\_back(P);

}

else if (gram1[5] == '|' && (!(!gram1[6])) && (!(!gram1[7]))) {

GS = make\_pair(gram1[0], gram1[7]);

P = make\_pair(GS, gram1[6]);

GPS.push\_back(P);

}

}

for (int i = 0; i <= GPS.size() - 1; i++) {

cout << GPS[i].first.first << " -> " << GPS[i].second << " -> " << GPS[i].first.second <<" \n";

}

for (int i = 0; i <= gramG.size()-1; i++) {

for (int j = 0; j <= (alphabet.size() - gramG.size()+1); j++) {

if (gramG[i] == alphabet[j]) {

alphabet.erase(j, 1);

}

}

}

for (int i = 0;i<=GPS.size()-1;i++) {

char letter\_first = GPS[i].first.first;

char transfer = GPS[i].second;

char letter\_second = GPS[i].first.second;

for (int j = i+1; j <= GPS.size()-1; j++) {

char previos1 = letter\_second;

char previos2 = GPS[j].first.second;

char alph = alphabet[0];

if (letter\_first == GPS[j].first.first && transfer == GPS[j].second) {

GPS[i].first.second = alphabet[0];

alphabet.erase(0, 1);

GPS.erase(GPS.begin() + j);

for (int l = 0; l <= GPS.size() - 1; l++) {

if (GPS[l].first.second == previos2 || GPS[l].first.second == previos1) {

GPS[l].first.second = alph;

}

if (GPS[l].first.first == previos1 || GPS[l].first.first == previos2) {

GPS[l].first.first = alph;

}

}

}

}

}

for (int i = 0; i <= GPS.size() - 1; i++) {

cout << "\n" << GPS[i].first.first << " -> " << GPS[i].second << " -> " << GPS[i].first.second << " \n";

}

return 0;

}

**Вывод:** В данной лабораторной работе я научился создавать программы способные строить на основе регулярной грамматики детерминированные конечный автоматы