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 Гугин И.В.

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:

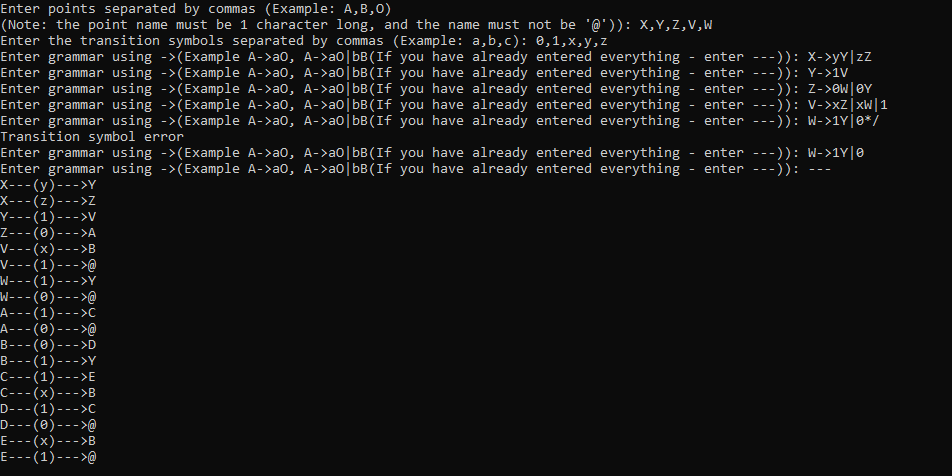
G=({X,Y,Z,V,W},{0,1,x,y,z},P,E), P:

1)X->yY|xX, 2)Y->1V; 3)Z->0W|0Y; 4)V->xZ|xW|1; 5)W->1Y|0

**Реализация:**

Программа написана на языке С++. Для хранения информации о переходах будем использовать 3 вектора. В первом хранится то, откуда был произведён переход, в 2 с помощью какого символа, в 3 куда. Например первый переход из Y в V с помощью 1, тогда в первый вектор запишем Y, в второй 1, в третий V. Если нам понадобиться доступ к этому переходу мы сможем обратиться по соответственному номеру. При запуске программу будут запрошены список точек и символов перехода, для дальнейшего использования в проверках и выборе символов для новых точек при переходе к ДКА. Информация о точках и символах перехода так же храниться в двух соответствующих векторах.

**Результат работы:**



Сам код:

#include <iostream>

#include <vector>

#include <string>

using namespace std;

int **main**()

{

vector<char> from;

vector<char> to;

vector<char> with;

vector<char> points;

vector<char> step;

vector<int> pos;

string g;

string alph = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

char c;

char b;

char a;

char d;

cout << "Enter points separated by commas (Example: A,B,O)"<< endl << "(Note: the point name must be 1 character long, and the name must not be '@')): ";

cin>>g;

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

c=g[i];

if(c!=','){

points.push\_back(c);

}

}

cout << "Enter the transition symbols separated by commas (Example: a,b,c): ";

cin>>g;

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

c=g[i];

if(c!=','){

step.push\_back(c);

}

}

while(1){

cout << "Enter grammar using ->(Example A->aO, A->aO|bB(If you have already entered everything - enter ---)): ";

cin>>g;

int flag=0;

int flag2=0;

if(g=="---") break;

if(g[1]=='-' && g[2]=='>'){

flag2=1;

}

if(flag2==0){

cout<<"Reminder: use ->"<<endl;

continue;

}

flag2=0;

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

c=g[0];

if(c==points[i]) flag2=1;

if(flag2==1) break;

}

if(flag2==0){

cout<<"Point error"<<endl;

continue;

}

flag2=0;

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

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

c=g[4];

if(c==points[i]) flag2=1;

if(flag2==1) break;

}

if(flag2==0){

cout<<"Point error"<<endl;

continue;

}

}

flag2=0;

if(g.size()==8){

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

c=g[7];

if(c==points[i]) flag2=1;

if(flag2==1) break;

}

if(flag2==0){

cout<<"Point error"<<endl;

continue;

}

}

flag2=0;

if(g.size()==11){

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

c=g[10];

if(c==points[i]) flag2=1;

if(flag2==1) break;

}

if(flag2==0){

cout<<"Point error"<<endl;

continue;

}

}

flag2=0;

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

c=g[4];

if(c==points[i]) flag2=1;

if(flag2==1) break;

}

if(flag2==0){

cout<<"Point error"<<endl;

continue;

}

flag2=0;

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

c=g[3];

if(c==step[i]) flag2=1;

if(flag2==1) break;

}

if(flag2==0){

cout<<"Transition symbol error"<<endl;

continue;

}

flag2=0;

if(g.size()>5){

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

c=g[6];

if(c==step[i]) flag2=1;

if(flag2==1) break;

}

if(flag2==0){

cout<<"Transition symbol error"<<endl;

continue;

}

}

flag2=0;

if(g.size()>8){

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

c=g[9];

if(c==step[i]) flag2=1;

if(flag2==1) break;

}

if(flag2==0){

cout<<"Transition symbol error"<<endl;

continue;

}

}

flag2=0;

if(g.size()==5 || g.size()==4){

if(g[1]=='-' && g[2]=='>'){

c=g[0];;

from.push\_back(c);

c=g[3];

with.push\_back(c);

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

c='@';

to.push\_back(c);

}

else{

c=g[4];

to.push\_back(c);

}

}

flag=1;

}

if(g.size()==7 || g.size()==8){

if(g[1]=='-' && g[2]=='>' && g[5]=='|'){

c=g[0];

from.push\_back(c);

from.push\_back(c);

c=g[3];

with.push\_back(c);

c=g[6];

with.push\_back(c);

c=g[4];

to.push\_back(c);

if(g.size()==7){

c='@';

to.push\_back(c);

}

else{

c=g[7];

to.push\_back(c);

}

}

flag=1;

}

if(g.size()==10 || g.size()==11){

if(g[1]=='-' && g[2]=='>' && g[5]=='|' && g[8]=='|'){

c=g[0];

from.push\_back(c);

from.push\_back(c);

from.push\_back(c);

c=g[3];

with.push\_back(c);

c=g[6];

with.push\_back(c);

c=g[9];

with.push\_back(c);

c=g[4];

to.push\_back(c);

c=g[7];

to.push\_back(c);

if(g.size()==10){

c='@';

to.push\_back(c);

}

else{

c=g[10];

to.push\_back(c);

}

}

flag=1;

}

if(flag=0) cout << "Input error" << endl;

}

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

c=points[i];

for(int j=0;j<alph.size();j++){

b=alph[j];

if(b==c){

alph.erase(j,1);

}

}

}

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

c=from[i];

for(int j=0;j<from.size();j++){

b=from[j];

if(c==b && with[i]==with[j] && to[i]!=to[j] && i!=j){

a=to[i];

d=to[j];

to[i]=alph[0];

for(int p = 0;p<to.size();p++){

if(from[p]==a){

from.push\_back(to[i]);

with.push\_back(with[p]);

to.push\_back(to[p]);

}

if(from[p]==d){

from.push\_back(to[i]);

with.push\_back(with[p]);

to.push\_back(to[p]);

}

}

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

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

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

alph.erase(0,1);

}

}

}

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

cout<<from[i]<<"---("<<with[i]<<")--->"<<to[i]<<endl;

}

return 0;

}

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