МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ МОЛДОВЫ

ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ МОЛДОВЫ

ФАКУЛЬТЕТ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ, ИНФОРМАТИКИ И МИКРОЭЛЕКТРОНИКИ

КАФЕДРА ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

**ОТЧЕТ**

По лабораторной работе №2

по дисциплине «Формальные языки и программирование компиляторов»

Тема: «Конечные автоматы»

Выполнил: студент группы TI – 155 Буянов Евгений

Проверила: преподаватель Скороходова Т.А.

Кишинев 2016

Задание

***Вариант № 5***

KA=(Q, ∑, δ, q0, F),

Q = { q0, q1, q2 , q3 },

∑ = { a, b },

F = { q3 }.

δ (q0, a ) = q1 ,

δ (q0, b ) = q0 ,

δ (q1, a ) = q2,

δ (q1, a ) = q3,

δ (q2, a ) = q3,

δ (q2, b) = q0.

1. Представить заданный конечный автомат в виде графа.

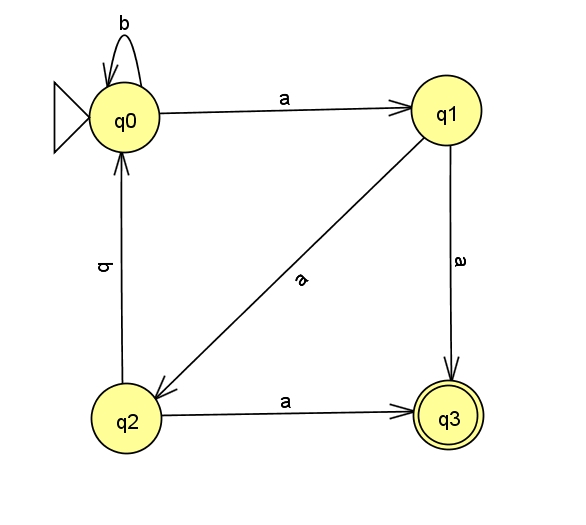


Рисунок 1-Недетерминированный конечный автомат

1. Является ли данный конечный автомат детерминированным?

Данный конечный автомат является недетерминированным, потому что из состояния q1 по a мы можем попасть в q2 и q3.

1. Если данный конечный автомат является недетерминированным, построить эквивалентный ему конечный детерминированный автомат.

|  |  |  |
| --- | --- | --- |
| Состояния | ∑ | |
| a | b |
| [q0] | [q1] | [q0] |
| [q1] | [q2 , q3] |  |
| [q2 , q3] | [q3] | [q0] |

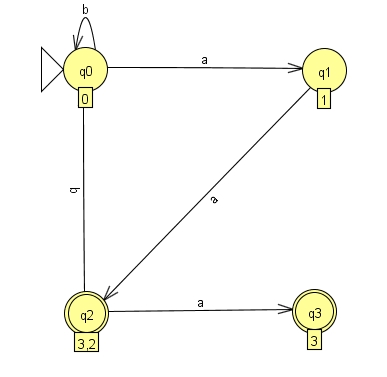


Рисунок 2-Детерминированный конечный автомат

1. Построить регулярную грамматику эквивалентную данному конечному автомату.

G=(VN, VT, P, S);

1. VN=Q,

VT= ∑,

S=,

P:{

;

;

;

;

;

;

;

;

}

1. Придумать цепочку над алфавитом ∑, по виду которой можно сказать, что она не подходит под общий вид цепочек, допускаемых данным конечным автоматом и следовательно не допускается данным конечным автоматом. Попытайтесь для этой цепочки написать последовательность конфигураций (q0, x) |— (qi1, x1) |— (qi2, x2) |— … |— (qf, ε), где qf ∈ F, чтобы убедиться, что автомат не допускает цепочку.

Цепочка над алфавитом ∑:

x=aababbabaa;

.

1. Для заданного конечного автомата AF=(Q, ∑, δ, q0, F) построить 5 цепочек, допускаемых этим автоматом. Длина цепочки должна быть не меньше, чем количество состояний во множестве Q плюс 2.

1) x=bbaabbaa;

2) x=aabbbaaa;

3) x=bbbaaa;

4) x=aabaabaa;

5) x=aabbaabbaa;

1. Написать общий вид цепочек, допускаемых данным конечным автоматом.

**L(KA)={(b\*a(a,aa,ab))\*}**

1. Для каждой построенной цепочки x написать последовательность конфигураций такую, что (q0, x) |— (qi1, x1) |— (qi2, x2) |— … |— (qf, ε), где qf ∈ F.
2. ;
3. ;
4. ;
5. ;
6. .
7. Для каждой из 5 построенных цепочек построить представление =uvw, удовлетворяющее свойствам леммы о разрастании.

К каждому слову можно применить лемму о разрастании т.к. для каждого слова соблюдаются следующие правила:

1.|,

2.,

3..

;

;

;

;

Проверяем истинность леммы при i=0:

;

;

Проверяем истинность леммы при i=2:

;

;

;

;

;

;

Проверяем истинность леммы при i=0:

;

;

Проверяем истинность леммы при i=2:

;

;

;

;

;

Проверяем истинность леммы при i=0:

;

;

Проверяем истинность леммы при i=2:

;

;

;

;

;

;

Проверяем истинность леммы при i=0:

;

;

Проверяем истинность леммы при i=2:

;

;

;

;

;

;

Проверяем истинность леммы при i=0:

;

;

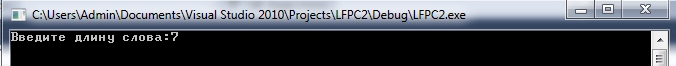
Проверяем истинность леммы при i=2:

;

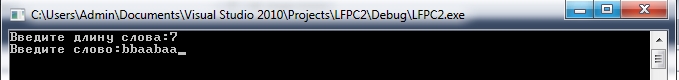
;

1. Программа проверяющая слова на допуск данному автомату.

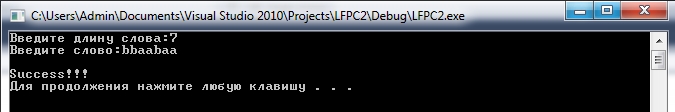
1)Описание: Пользователь вводит длину слова:



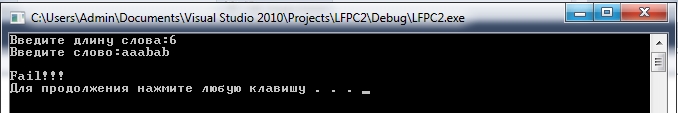
Затем вводит само слово:



После чего программа выводит на экран слово “Success” в случае если слово принадлежит автомату:



Если слово не принадлежит автомату того на экран выводится слово ”Fail”:



После нажатия на любую клавишу мы заново переходим в первое меню, где нас просят ввести длину слова;

2) Код программы:

#include<iostream>

#include<vector>

using namespace std;

void q0(vector<char>);

void q1(vector<char>);

void q2(vector<char>);

void q3(vector<char>);

void outWord(vector<char> vec){

for(vector<char>::const\_iterator i = vec.begin(); i != vec.end(); ++i)

cout << \*i ;

cout<<endl;

}

vector<char> inWord(vector<char> vec,int n){

char a;

for(int i=0;i<n;++i){

cin>>a;

vec.push\_back(a);

}

cout<<endl;

return vec;

}

void checkResult(bool check){

if(check) cout<<"Success!!!"<<endl;

else cout<<"Fail!!!"<<endl;

}

void verificateWord(vector<char> vec){

if(vec.front()=='a') q1(vec);

else if(vec.front()=='b') q0(vec);

}

void q0(vector<char> vec){

vec.erase(vec.begin());

if(vec.front()=='a') q1(vec);

else if(vec.front()=='b') q0(vec);

else if(true) checkResult(false);

}

void q1(vector<char> vec){

vec.erase(vec.begin());

if(vec.front()=='a') q2(vec);

else checkResult(false);

}

void q2(vector<char> vec){

vec.erase(vec.begin());

if(vec.size()==0) checkResult(true);

else{

if(vec.front()=='a') q3(vec);

else if(vec.front()=='b') q0(vec);

else if(true) checkResult(false);

}

}

void q3(vector<char> vec){

vec.erase(vec.begin());

if(vec.size()==0) checkResult(true);

else checkResult(false);

}

int main(){

setlocale (LC\_ALL, "RUS");

vector<char>word;

int n;

while(true){

start:

cout<<"Введите длину слова:";

cin>>n;

if(!word.empty())

word.clear();

cout<<"Введите слово:";

word=inWord(word,n);

if(word.size()<6){

cout<<"Слово слишком короткое."<<endl;

system("pause");

system("cls");

goto start;

}

verificateWord(word);

system("pause");

system("cls");

}

return 0;

}

**Вывод:**

В ходе проделанной работы я применил на практике знания, полученные на теоретических и практических занятиях. Лабораторная работа основывалась на конечных автоматов, построении теоремы FG, леммы о разрастании. Основой для данной работы стали преобразования недетерминированного конечного автомата в детерминированный. Программа составлена для определения слова на допуск.