**ИТМО Кафедра Информатики и прикладной математики**

Домашнее задание №2

«Построение НКА и ДКА по регулярному выражению»   
Вариант 7

**Выполнил: студент группы P3217**

**Плюхин Дмитрий**

**Преподаватель: Лаздин А.В.**

**2017 год**

1. **Задание**

По заданному регулярному выражению ba((ab)|(ac))\*

• Построить недетерминированный КА

• По полученному НДА построить ДКА

• Для ДКА написать программу-распознаватель предложений языка, порождаемого регулярным выражением.

Продемонстрировать работу распознавателя на различных примерах (не менее трех правильных) предложений.

1. **Недетерминированный КА**

b

a

ε

ε

ε

c

ε

ε

a

ε

b

a

ε

1. **Детерминированный КА**

Избавляемся от ε – переходов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Новое состояние | ε- замыкание | A | b | c |
| 1 | ε(1)={1} | ∅ | ε(2) | ∅ |
| 2 | ε(2)={2} | ε(3) | ∅ | ∅ |
| 3 | ε(3)={4,5,6,12} | {ε(7), ε(8)} | ∅ | ∅ |
| 4 | ε(7)={7} | ∅ | ε(9) | ∅ |
| 5 | ε(8)={8} | ∅ | ∅ | ε(10) |
| 6 | ε(9)={4,5,6,9,11,12} | { ε(7), ε(8)} | ∅ | ∅ |
| 7 | ε(10)={4,5,6,10,11,12} | { ε(7), ε(8)} | ∅ | ∅ |

a

a

b

a

a

b

a

a

a

c

Избавляемся от недетерминированностей

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Новое состояние | Старые состояния | A | b | c |
| 1 | 1 | ∅ | 2 | ∅ |
| 2 | 2 | 3 | ∅ | ∅ |
| 3 | 3 | 4,5 | ∅ | ∅ |
| 4 | 4,5 | ∅ | 6 | 7 |
| 5 | 6 | 4,5 | ∅ | ∅ |
| 6 | 7 | 4,5 | ∅ | ∅ |

a

b

a

a

b

с

a

1. **Код программы**

#include <stdio.h>

#include <string.h>

#include <fstream>

#include <windows.h>

#include <iostream>

#include <cstdlib>

#include <time.h>

#include <sstream>

using namespace std;

typedef struct \_Node Node, \*PNode;

typedef struct \_Link{

PNode nextNode;

char symbol;

} Link, \*PLink;

typedef struct \_Node{

unsigned numberOfLinks;

bool isFinish;

PLink links;

} Node, \*PNode;

bool checkSentence(PNode startNode, string sentence){

int length = sentence.length();

bool broken = false;

PNode currentNode = startNode;

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

for (int j = 0; j < currentNode->numberOfLinks; j++){

if (currentNode->links[j].symbol == sentence[i]){

currentNode = currentNode->links[j].nextNode;

broken = true;

break;

}

}

if (!broken) return false;

broken = false;

}

if (currentNode->isFinish == true) return true;

return false;

}

PNode getDKA(){

PNode startNode = (PNode)malloc(sizeof(Node));

PNode currentNode;

//state 1

startNode->isFinish = false;

startNode->numberOfLinks = 1;

startNode->links = (PLink)malloc(sizeof(Link));

startNode->links[0].nextNode = (PNode)malloc(sizeof(Node));

startNode->links[0].symbol = 'b';

//state 2

currentNode = startNode->links[0].nextNode;

currentNode->isFinish = false;

currentNode->numberOfLinks = 1;

currentNode->links = (PLink)malloc(sizeof(Link));

currentNode->links[0].nextNode = (PNode)malloc(sizeof(Node));

currentNode->links[0].symbol = 'a';

//state 3

currentNode = currentNode->links[0].nextNode;

currentNode->isFinish = true;

currentNode->numberOfLinks = 1;

currentNode->links = (PLink)malloc(sizeof(Link));

currentNode->links[0].nextNode = (PNode)malloc(sizeof(Node));

currentNode->links[0].symbol = 'a';

//state 4

currentNode = currentNode->links[0].nextNode;

currentNode->isFinish = false;

currentNode->numberOfLinks = 2;

currentNode->links = (PLink)malloc(2\*sizeof(Link));

currentNode->links[0].nextNode = (PNode)malloc(sizeof(Node));

currentNode->links[0].symbol = 'b';

currentNode->links[1].nextNode = (PNode)malloc(sizeof(Node));

currentNode->links[1].symbol = 'c';

PNode pFourthNode = currentNode;

//state 5

currentNode = currentNode->links[0].nextNode;

currentNode->isFinish = true;

currentNode->numberOfLinks = 1;

currentNode->links = (PLink)malloc(sizeof(Link));

currentNode->links[0].nextNode = pFourthNode;

currentNode->links[0].symbol = 'a';

//state 6

currentNode = pFourthNode->links[1].nextNode;

currentNode->isFinish = true;

currentNode->numberOfLinks = 1;

currentNode->links = (PLink)malloc(sizeof(Link));

currentNode->links[0].nextNode = pFourthNode;

currentNode->links[0].symbol = 'a';

return startNode;

}

int main(int argc, char\* argv[]){

char identifier[256];

PNode startNode = getDKA();

string sentence = "";

while(true){

cout << "Type sentence for checking : " << endl;

getline(cin, sentence);

cout << "Result : ";

if (checkSentence(startNode, sentence)){

cout << "valid";

} else {

cout << "invalid";

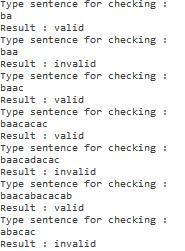
}

cout << endl;

}

}

1. **Результаты работы программы**

****