САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ВОДНЫХ КОММУНИКАЦИЙ

Кафедра ВСиИ

**КУРСОВАЯ РАБОТА**

по дисциплине «Системное программное обеспечение»

**Моделирование работы конечного распознавателя для последовательности элементов типа «дата» в немецком формате, разделенных запятыми и заключённых в фигурные скобки**

*Вариант № 15*

Выполнил:

студент группы ИС-31

Мельников А.

Санкт-Петербург

2009 год

**Содержание**

Задание на курсовую работу

Введение

1 Составление формальной грамматики

2 Построение конечного автомата

3 Программное моделирование работы конечного автомата

4 Граф детерминированного автомата

5 Блок-схема

6 Примеры разбора строк

**Задание на курсовую работу**

*Моделирование работы конечного распознавателя для* последовательности элементов типа «дата» в немецком формате (ДД.ММ.ГГГГ), разделенных запятыми, при этом значение даты должно быть помещено в фигурные скобки, а год должен отображаться четырьмя символами, например, ({01.12.2001},{05.07.2003});

**Введение**

**Учебная цель.** Получение практических навыков построения моделей конечных распознавателей.

**Теоретические сведения.**

Недетерминированный конечный автомат (НКА) - это пятерка M = (Q, T, D, q0, F), где

* Q - конечное множество состояний;
* T - конечное множество допустимых входных символов (входной алфавит);
* D - функция переходов (отображающая множество Q(T{e}) во множество подмножеств множества Q), определяющая поведение управляющего устройства;



* q0 Q - начальное состояние управляющего устройства;



* F Q - множество заключительных состояний.



Недетерминизм автомата заключается в том, что, во-первых, находясь в некотором состоянии и обозревая текущий символ, автомат может перейти в одно из, вообще говоря, нескольких возможных состояний, и во-вторых, автомат может делать переходы по e.

Пусть M = (Q, T, D, q0, F) - НКА. Конфигурацией автомата M называется пара (q, w) QT\*, где q - текущее состояние управляющего устройства, а w - цепочка символов на входной ленте, состоящая из символа под головкой и всех символов справа от него. Конфигурация (q0, w) называется начальной, а конфигурация (q, e), где q F - заключительной (или допускающей).



Пусть M = (Q, T, D, q0, F) - НКА. Тактом автомата M называется бинарное отношение , определенное на конфигурациях M следующим образом: если p D(q, a), где a T {e}, то (q, aw) (p, w) для всех w T\*.



Будем обозначать символом + (\*) транзитивное (рефлексивно- транзитивное) замыкание отношения .



Говорят, что автомат M допускает цепочку w, если (q0, w) \*(q, e) для некоторого q F. Языком, допускаемым (распознаваемым, определяемым) автоматом M, (обозначается L(M)), называется множество входных цепочек, допускаемых автоматом M. Т.е.



Важным частным случаем недетерминированного конечного автомата является детерминированный конечный автомат, который на каждом такте работы имеет возможность перейти не более чем в одно состояние и не может делать переходы по e.

Пусть M = (Q, T, D, q0, F) - НКА. Будем называть M детерминированным конечным автоматом (ДКА), если выполнены следующие два условия:

* D(q, e) = для любого q Q, и



* D(q, a) содержит не более одного элемента для любых q Q и a T.



Так как функция переходов ДКА содержит не более одного элемента для любой пары аргументов, для ДКА мы будем пользоваться записью D(q, a) = p вместо D(q, a) = {p}.

Конечный автомат может быть изображен графически в виде диаграммы, представляющей собой ориентированный граф, в котором каждому состоянию соответствует вершина, а дуга, помеченная символом a T {e}, соединяет две вершины p и q, если p D(q, a). На диаграмме выделяются начальное и заключительные состояния.



**Конечный распознаватель** – это модель устройства с конечным числом состояний, которое отличает правильно образованные, или «допустимые» цепочки, от «недопустимых».

Примером задачи распознавания может служить проверка нечетности числа единиц в произвольной цепочке, состоящей из нулей и единиц. Соответствующий конечный автомат будет допускать все цепочки, содержащие нечетное число единиц, и отвергать все цепочки с четным их числом. Назовем его «контролером нечетности».

На вход конечного автомата подается цепочка символов из конечного множества, называемого **входным алфавитом** автомата, и представляющего собой совокупность символов, для работы с которыми он предназначен. Как допускаемые, так и отвергаемые автоматом цепочки, состоят только из символов входного алфавита. Символы, не принадлежащие входному алфавиту, нельзя подавать на вход автомата. Входной алфавит контроллера нечетности состоит из двух символов: «0» и «1».

В каждый момент времени конечный автомат имеет дело лишь с одним входным символом, а информацию о предыдущих символах входной цепочки сохраняет с помощью конечного **множества состояний**. Согласно этому представлению, автомат помнит о прочитанных ранее символах только то, что при их обработке он перешел в некоторое состояние, которое и является памятью автомата о прошлом.

Работу автомата можно описать математически с помощью функции переходов, которая по текущему состоянию и текущему входному символу дает новое состояние автомата . Символически эта зависимость описывается так:



.



Некоторые состояния автомата выбираются в качестве **допускающих**, или **заключительных**. Если автомат, начав работу в начальном состоянии, при прочтении всей цепочки переходит в одно из допускающих состояний, то говорят, что эта цепочка допускается автоматом. Если последнее состояние автомата не является допускающим, то говорят, что автомат отвергает цепочку.

**1 Составление формальной грамматики**

Фраза языка представляет собой список, поэтому из начального символа грамматики должен выводится список:

R0: <предложение>::==<фраза>

R1: <фраза>::==<дата> | <дата>,<фраза>

Дата представляет собой линейную структуру:

R2: <дата>::=={<месяц>.<год>}

Аналогично год, месяц и день:

R3: <год>::==<цифра><цифра><цифра><цифра>

R4: <месяц>: :== <месяцб>. <деньб> |<месяцм>. <деньм>| <февраль> <деньф>

R5: <месяцб>::=01|03|05|07|08|10|12

R6: <месяцм>::=04|06|09|11

R7: <февраль>::=02

R8: <деньб>::==<цифра2><цифра>| 3<цифра1>

R9: <деньм>::==<цифра2><цифра>| 30

R10: <деньф>::==<цифра1><цифра>| 2<цифра3>

R11: <цифра>::==0|1|2|3|4|5|6|7|8|9|

R12: <цифра1>::==0|1

R13: <цифра2>::==0|1|2

R14: <цифра3>::==0|1|2|3|4|5|6|7|8

Таким образом, требуемую грамматику можно описать следующей структурой:

* Множество терминальных символов: {, }, ., , ,0,1,2,3,4,5,6,7,8,9.
* Множество нетерминальных символов: <фраза>, <дата>, <год>, <месяц>, <день>, <цифра>, <цифра1>, <цифра2>.
* Множество правил вывода R0,R1, R2, R3, R4, R5, R6, R7, R8, R9, R10, R11, R12, R13, R14.

**2 Построение конечного автомата**

Между конечными автоматами и автоматными грамматиками существует тесная связь: класс языков, допускаемых конечными автоматами, совпадает с классом языков, порождаемых автоматными грамматиками.

Для построения конечного автомата составленную грамматику путем введения дополнительных состояний надо преобразовать к автоматному виду, в результате получится следующая таблица переходов:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | { | } | . | , |
| да |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| нет |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| день | нет | нет | нет | нет | нет | нет | нет | нет | нет | нет | нет | нет | денф | нет |
| денб | Дб1 | Дб1 | Дб1 | Цф1 | нет | нет | нет | нет | нет | нет | нет | нет | нет | нет |
| Дб1 | Дб2 | Дб2 | Дб2 | Дб2 | Дб2 | Дб2 | Дб2 | Дб2 | Дб2 | Дб2 | нет | нет | нет | нет |
| Дб2 | нет | нет | нет | нет | нет | нет | нет | нет | нет | нет | нет | нет | мес | нет |
| Цф1 | Дб2 | Дб2 | нет | нет | нет | нет | нет | нет | нет | нет | нет | нет | нет | нет |
| денм | Дб1 | Дб1 | Дб1 | Цф0 | нет | нет | нет | нет | нет | нет | нет | нет | нет | нет |
| Цф0 | Дб2 | нет | нет | нет | нет | нет | нет | нет | нет | нет | нет | нет | нет | нет |
| денф | Дб1 | Дб1 | Цф3 | нет | нет | нет | нет | нет | нет | нет | нет | нет | нет | нет |
| Цф3 | Дб2 | Дб2 | Дб2 | Дб2 | Дб2 | Дб2 | Дб2 | Дб2 | нет | нет | нет | нет | нет | нет |
| мес | Мес0 | Мес1 | нет | нет | нет | нет | нет | нет | нет | нет | нет | нет | нет | нет |
| Мес0 | нет | месб | фев | месб | месм | месб | месм | месб | месб | месм | нет | нет | нет | нет |
| Мес1 | месб | месм | месб | нет | нет | нет | нет | нет | нет | нет | нет | нет | нет | нет |
| месб | нет | нет | нет | нет | нет | нет | нет | нет | нет | нет | нет | нет | денб | нет |
| месм | нет | нет | нет | нет | нет | нет | нет | нет | нет | нет | нет | нет | денм | нет |
| дата | нет | нет | нет | нет | нет | нет | нет | нет | нет | нет | год | нет | нет | нет |
| год | Цг1 | Цг1 | Цг1 | Цг1 | Цг1 | Цг1 | Цг1 | Цг1 | Цг1 | Цг1 | нет | нет | нет | нет |
| Цг1 | Цг2 | Цг2 | Цг2 | Цг2 | Цг2 | Цг2 | Цг2 | Цг2 | Цг2 | Цг2 | нет | нет | нет | нет |
| Цг2 | Цг3 | Цг3 | Цг3 | Цг3 | Цг3 | Цг3 | Цг3 | Цг3 | Цг3 | Цг3 | нет | нет | нет | нет |
| Цг3 | Цг4 | Цг4 | Цг4 | Цг4 | Цг4 | Цг4 | Цг4 | Цг4 | Цг4 | Цг4 | нет | нет | нет | нет |
| Цг4 | нет | нет | нет | нет | нет | нет | нет | нет | нет | нет | нет | да | нет | день |

**3 Граф детерминированного автомата**

Для того, чтобы улучшить зрительное восприятие и облегчить понимание сложных синтаксических описаний, часто применяют представление правил грамматики в виде синтаксических диаграмм.

Синтаксическая диаграмма представляет собой ориентированный граф для каждого правила грамматики.

**4 Программное моделирование работы конечного автомата**

#include "iostream.h"

#include "stdio.h"

#include "conio.h"

int main()

{int i,j,kol,tsost,slsost,tsymb;

int tabl[24][14]={{0,0,0,0,0,0,0,0,0,0,0,0,0,0},//da

{1,1,1,1,1,1,1,1,1,1,1,1,1,1},//net

{1,1,1,1,1,1,1,1,1,1,3,1,1,1},

{4,4,4,5,1,1,1,1,1,1,1,1,1,1},

{1,6,6,6,6,6,6,6,6,6,1,1,1,1},

{8,9,1,1,1,1,1,1,1,1,1,1,1,1},

{1,1,1,1,1,1,1,1,1,1,1,1,20,1},

{16,16,16,16,16,16,16,16,16,16,1,1,1,1},

{1,1,1,1,1,1,1,1,1,1,1,1,10,1},

{1,1,1,1,1,1,1,1,1,1,1,1,11,1},

{12,13,1,1,1,1,1,1,1,1,1,1,1,1},

{14,15,1,1,1,1,1,1,1,1,1,1,1,1},

{1,1,1,1,23,1,23,1,1,23,1,1,1,1},

{23,1,1,1,1,1,1,1,1,1,1,1,1,1},

{1,23,1,23,1,23,1,23,23,1,1,1,1,1},

{23,1,23,1,1,1,1,1,1,1,1,1,1,1},

{17,17,17,17,17,17,17,17,17,17,1,1,1,1},

{18,18,18,18,18,18,18,18,18,18,1,1,1,1},

{19,19,19,19,19,19,19,19,19,19,1,1,1,1},

{1,1,1,1,1,1,1,1,1,1,1,0,1,2},

{21,22,1,1,1,1,1,1,1,1,1,1,1,1},

{1,23,23,23,23,23,23,23,23,23,1,1,1,1},

{23,23,23,1,1,1,1,1,1,1,1,1,1,1},

{1,1,1,1,1,1,1,1,1,1,1,1,7,1},

};

printf("matrica\n");

for (i=0;i<24;i++) {for (j=0;j<14;j++) printf("%4d",tabl[i][j]); printf("\n");};

char ch, inpstr[80] ;

printf("\n ENTER STRING ");

i=0;

while ((ch=getch()) !=13 && i<80)

{putch(ch);

inpstr[i++]=ch;}

inpstr[i]='\0';

kol=i-1;

printf("\n input string:");

printf(inpstr);

printf("\n");

tsost=2;

for (i=0;i<=kol;i=i+1)

{ tsymb=inpstr[i];

switch (tsymb)

{ case '0': slsost=tabl[tsost][0]; break;

case '1': slsost=tabl[tsost][1]; break;

case '2': slsost=tabl[tsost][2]; break;

case '3': slsost=tabl[tsost][3]; break;

case '4': slsost=tabl[tsost][4]; break;

case '5': slsost=tabl[tsost][5]; break;

case '6': slsost=tabl[tsost][6]; break;

case '7': slsost=tabl[tsost][7]; break;

case '8': slsost=tabl[tsost][8]; break;

case '9': slsost=tabl[tsost][9]; break;

case '{': slsost=tabl[tsost][10]; break;

case '}': slsost=tabl[tsost][11]; break;

case '.': slsost=tabl[tsost][12]; break;

case ',': slsost=tabl[tsost][13]; break;

default: slsost=1;}

printf("%5d\n",slsost);

tsost=slsost;

};

switch (slsost)

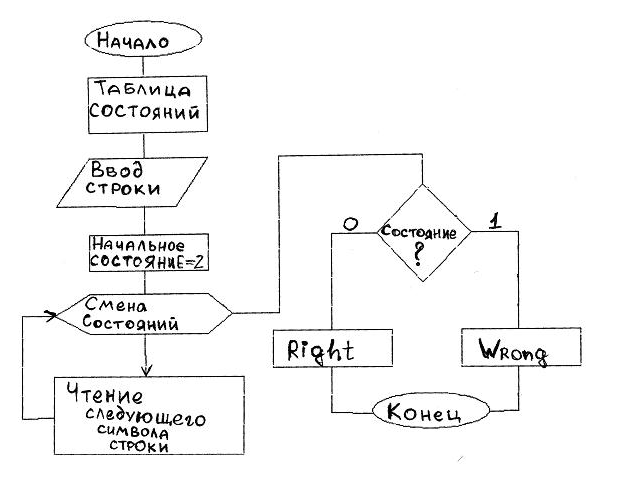
{ case 1:cout<<"\n STRING is WRONG \n"; break;

case 0:cout<<"\n STRING is RIGHT \n";break;}

return 0;

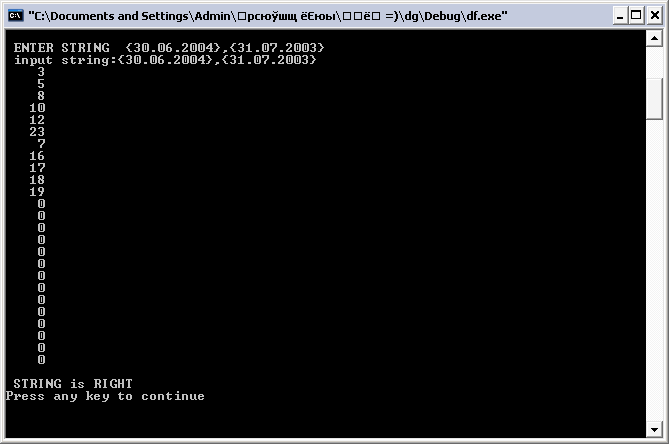
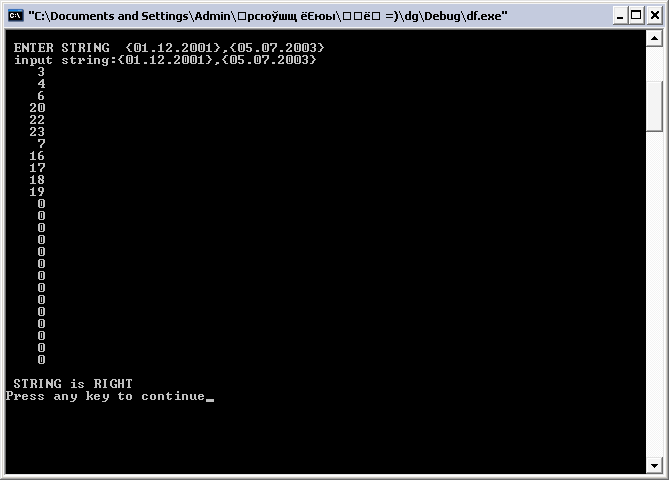
};

**5 Блок-схема**



**6 Примеры**

Правильные строки:



Неправильные строки:

