Московский авиационный институт

(Национальный исследовательский университет)

## Факультет «Информационные технологии и прикладная математика»

**Курсовая работа**

**по курсу**

**«Системы программирования»**

**по теме**

**«Разработка и реализация автомата с магазинной памятью»**

**IV семестр**

|  |  |
| --- | --- |
| **Студент:** | Шевчук П.В. |
| **Группа:** | М8О-204Б |
| **Преподаватель:** | Семенов А.С. |
| **Оценка:** |  |
| **Дата:** |  |
| **Подпись:** |  |

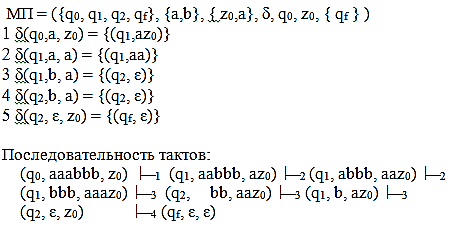
Москва

2018

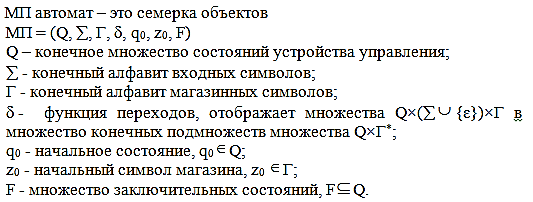
**Задание**

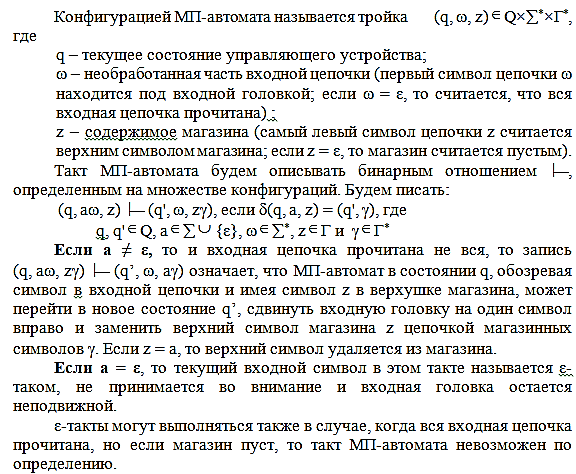
Построение МП-автомата, допускающего язык L = {anbn | n ≥ 0}.

Входная цепочка: aaabbb



**Теория**





**Код программы**

Приведены основные фрагменты кода.

*Program.cs*

using System;

using System.Collections.Generic;

using System.Collections;

using System.Linq;

using System.Data;

using System.Text;

namespace myCompiler

{

class Program

{

switch (Console.ReadLine())

{

...

case "7": //МП - автоматы

myMp Mp = new myMp(new ArrayList() { "q0", "q1", "q2", "qf" }, new ArrayList() { "a", "b" }, new ArrayList() { "z0", "a" }, "q0", new ArrayList() { "qf" });

Mp.addDeltaRule("q0", "a", "z0", new ArrayList() { "q1" }, new ArrayList() { "a", "z0" });

Mp.addDeltaRule("q1", "a", "a", new ArrayList() { "q1" }, new ArrayList() { "a", "a" });

Mp.addDeltaRule("q1", "b", "a", new ArrayList() { "q2" }, new ArrayList() { "e" });

Mp.addDeltaRule("q2", "b", "a", new ArrayList() { "q2" }, new ArrayList() { "e" });

Mp.addDeltaRule("q2", "e", "z0", new ArrayList() { "qf" }, new ArrayList() { "e" });

Console.Write("Debug Mp ");

Mp.debugDelta();

Console.WriteLine("\nEnter the line :");

Console.WriteLine(Mp.Execute\_(Console.ReadLine()).ToString());

break;

}

}

}

*MyMP.cs*

using System;

using System.Collections.Generic;

using System.Collections;

using System.Linq;

using System.Text;

namespace myCompiler

{

class DeltaQSigmaGamma

{

// структура Delta отображения

private string LeftQ = null; // исходное состояние

private string LeftT = null; // символ входной цепочки

private string LeftZ = null; // верхний символ магазин

private ArrayList RightQ = null; // множество следующих состояний

private ArrayList RightZ = null; // множество символов магазина

public string leftQ { get { return LeftQ; } set { LeftQ = value; } }

public string leftT { get { return LeftT; } set { LeftT = value; } }

public string leftZ { get { return LeftZ; } set { LeftZ = value; } }

public ArrayList rightQ { get { return RightQ; } set { RightQ = value; } }

public ArrayList rightZ { get { return RightZ; } set { RightZ = value; } }

// Delta ( q1 , a , z ) = { {q} , {z1z2...} }

// LeftQ LeftT LeftZ RightQ RightZ

public DeltaQSigmaGamma(string LeftQ, string LeftT, string LeftZ, ArrayList RightQ, ArrayList RightZ)

{

this.LeftQ = LeftQ;

this.LeftT = LeftT;

this.LeftZ = LeftZ;

this.RightQ = RightQ;

this.RightZ = RightZ;

}

} // end class Delta

class myMp : Automate //МП = {}

{

// Q - множество состояний МП - автоматa

// Sigma - алфавит входных символов

// DeltaList - правила перехода

// Q0 - начальное состояние

// F - множество конечных состояний

public ArrayList Gamma = null; //алфавит магазинных символов

public string z0 = null; //начальный символ магазина

public Stack Z = null;

private int c = 1;

//

public myMp(ArrayList Q, ArrayList Sigma, ArrayList Gamma, string Q0, ArrayList F)

: base(Q, Sigma, F, Q0)

{

this.Gamma = Gamma;

this.Z = new Stack();

Q0 = Q[0].ToString(); // начальное состояние

Z.Push(Q0); // начальный символ в магазине

this.F = F; // пустое множество заключительных состояний

}

//

public myMp(myGrammar KCgrammar)

: base(new ArrayList() { "q" }, KCgrammar.T, new ArrayList() { }, "q")

{

this.Gamma = new ArrayList();

this.Z = new Stack();

foreach (string v1 in KCgrammar.V) // магазинные символы

Gamma.Add(v1);

foreach (string t1 in KCgrammar.T)

Gamma.Add(t1);

Q0 = Q[0].ToString(); // начальное состояние

Z.Push(KCgrammar.S0); // начальный символ в магазине

F = new ArrayList(); // пустое множество заключительных состояний

DeltaQSigmaGamma delta = null;

foreach (string v1 in KCgrammar.V)

{ // сопоставление правил с отображениями

ArrayList q1 = new ArrayList();

ArrayList z1 = new ArrayList();

foreach (Prule rule in KCgrammar.Prules)

{

if (rule.leftNoTerm == v1)

{

Stack zb = new Stack();

ArrayList rr = new ArrayList(rule.rightChain);

rr.Reverse();

foreach (string s in rr)

zb.Push(s);

z1.Add(zb);

q1.Add(Q0);

}

}

delta = new DeltaQSigmaGamma(Q0, "e", v1, q1, z1);

DeltaList.Add(delta);

}

foreach (string t1 in KCgrammar.T)

{

Stack e = new Stack();

e.Push("e");

delta = new DeltaQSigmaGamma(Q0, t1, t1, new ArrayList() { Q0 }, new ArrayList() { e });

DeltaList.Add(delta);

}

}

public void addDeltaRule(string LeftQ, string LeftT, string LeftZ, ArrayList RightQ, ArrayList RightZ)

{

DeltaList.Add(new DeltaQSigmaGamma(LeftQ, LeftT, LeftZ, RightQ, RightZ));

}

public bool Execute\_(string str)

{

string currState = this.Q0;

DeltaQSigmaGamma delta = null;

int i = 0;

str = str + "e";

for (; ; ) // empty step

{

delta = findDelta(currState, str[i].ToString());

if (delta == null) return false;

if (delta.leftT != "e")

{

for (; i < str.Length;)

{

this.Q = delta.rightQ;

currState = arrToStr(delta.rightQ);

if (delta.leftZ == Z.Peek().ToString() && delta.rightZ[0].ToString() == "e")

{

this.Z.Pop();

}

else this.Z.Push(delta.leftT);

i++;

break;

}

}

else if (delta.leftT == "e")

{

this.Q = delta.rightQ;

this.Z.Pop();

if (this.Z.Count == 0) return true;

else return false;

}

} // end for

} // end Execute\_

//

// поиск правила по состоянию.

private DeltaQSigmaGamma findDelta(string Q, string a)

{

foreach (DeltaQSigmaGamma delta in this.DeltaList)

{

if (delta.leftQ == Q && delta.leftT == a) return delta;

}

return null; // not find

}

//\*\*\* вспомогательные процедуры \*\*\*

//объединение множеств A or B

public ArrayList Unify(ArrayList A, ArrayList B)

{

ArrayList unify = A;

foreach (string s in B)

if (!A.Contains(s))

unify.Add(s);

return unify;

}

//преобразование элементов массива в строку

public string arrToStr(ArrayList array)

{

if (array.Equals(null)) return null;

else

{

string newLine = "";

foreach (string s in array)

newLine += s;

return newLine;

}

}

public string StackToString(Stack Z)

{

if (Z.Count == 0) return null;

else

{

string newLine = "";

Stack temp = new Stack();

for (int i = 0; i < Z.Count; i++)

{

temp.Push(Z.Pop());

newLine += Z.Peek();

}

for (int i = 0; i < temp.Count; i++)

Z.Push(temp.Pop());

return newLine;

}

}

// \*\* Debug \*\*

public string DebugStack(Stack s)

{ // печать текущего состояния магазина

string p = "|";

Stack s1 = new Stack();

while (s.Count != 0)

{

s1.Push(s.Pop());

p = p + s1.Peek().ToString();

}

while (s1.Count != 0) s.Push(s1.Pop());

return p;

}

public void debugDelta()

{

Console.WriteLine("Deltarules :");

if (this.DeltaList == null) Console.WriteLine("null");

else

foreach (DeltaQSigmaGamma d in this.DeltaList)

{ // тут

Console.Write("( " + d.leftQ + " , " + d.leftT + " , " + d.leftZ + " )");

Console.Write(" -> \n");

Console.WriteLine("[ { " + arrToStr(d.rightQ) + " } , { " + arrToStr(d.rightZ) + " } ]");

}

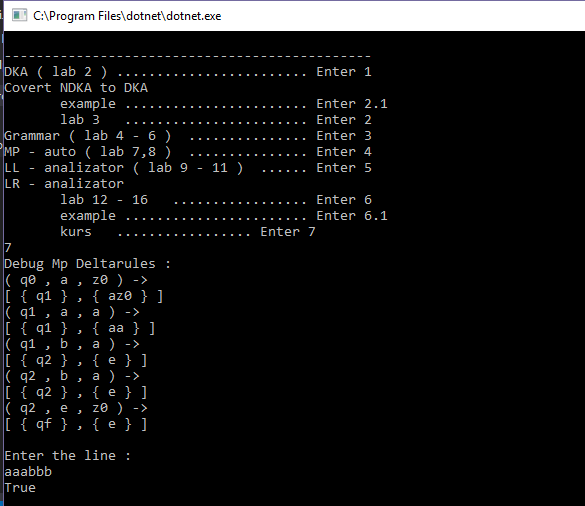
}

}

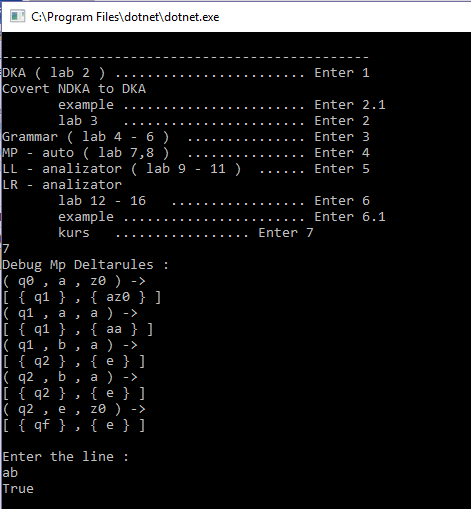
}

**Пример работы**

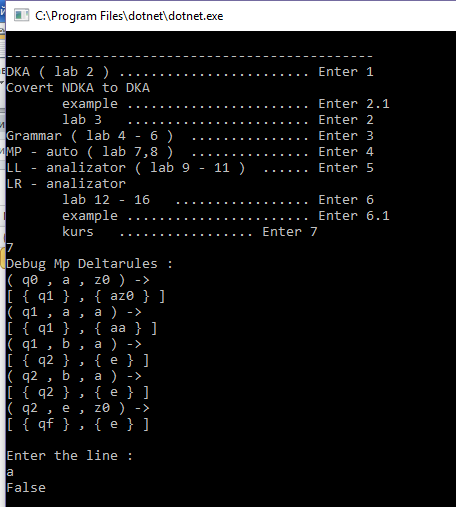
Входная строка: aaabbb – принадлежит языку L.



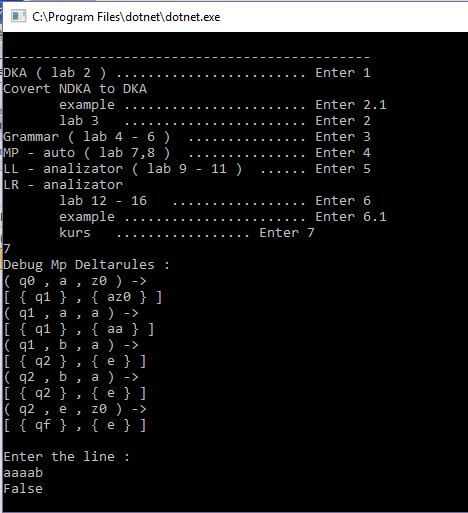
Входная строка: ab – принадлежит языку L.



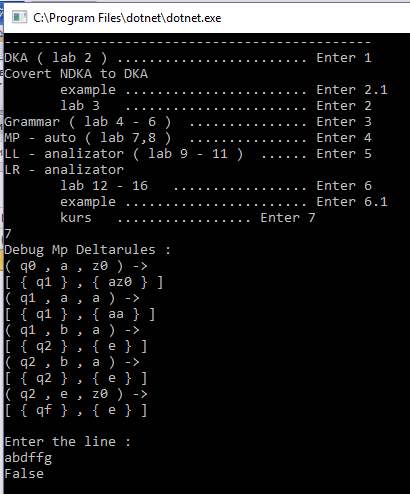
Входная строка: a – не принадлежит языку L.



Входная строка: aaaab – не принадлежит языку L.



Входная строка: abdffg – принадлежит языку L.



**Заключение**

Во время выполнения курсовой работы были отточены навыки работы с мп-автоматами. Это несомненно было полезно, так как поможет в дальнейшем обучении. Так же были приобретены знания в программировании на с# и повторены основные принципы ООП.

В процессе реализации построения мп-автомата был написан код, осуществляющий проверку принадлежности строки заданному языку.