Московский энергетический институт (ТУ)

# Программное моделирование работы

### МАШИНЫ ТЬЮРИНГА

Предмет: Теория алгоритмов.

Преподаватель: Старичкова Юлия Викторовна.

Работу выполнили: Ахремцев Ярослав, Захаров Данила, Ковех Ольга.

2009

# Задание

Необходимо создать модель машины Тьюринга. На входе программы должны быть два файла – первый, список правил, представленный в любой удобной форме, и, второй, лента входных данных. (При выполнении данной задачи входная информация находилась в одном файле с расширение .txt). Программа должна вести лог своих действий, сообщать о завершении работы, предусмотреть пошаговый режим выполнения с подтверждением пользователем перехода к следующему действию и распечаткой состояния на каждом шаге.

Перед тем, как начать описание выполненной работы, приведём список правил поведения машины Тьюринга, ориентируясь на которые была создана программа.

# Описание машины Тьюринга

Машина Тьюринга имеет бесконечную в обе стороны ленту, разделенную на квадратики (ячейки). В каждой ячейке может быть записан некоторый символ из фиксированного (для данной машины) конечного множества, называемого алфавитом данной машины. Один из символов алфавита выделен и называется "пробелом", предполагается, что изначально вся лента пуста, то есть заполнена пробелами.

Итак, машина Тьюринга может менять содержимое ленты с помощью специальной читающей и пишущей головки, которая движется вдоль ленты. В каждый момент головка находится в одной из ячеек. Машина Тьюринга получает от головки информацию о том, какой символ та видит, и в зависимости от этого (и от своего внутреннего состояния) решает, что делать, то есть какой символ записать в текущей ячейке и куда сдвинуться после этого (налево, направо или остаться на месте). При этом также меняется внутреннее состояние машины (мы предполагаем, что машина, не считая ленты, имеет конечную память, то есть конечное число внутренних состояний). Еще необходимо условиться, с чего мы начинаем и когда кончаем работу. Таким образом, чтобы задать машину Тьюринга, надо указать следующие объекты:

1. произвольное конечное множество А (алфавит);
2. некоторый выделенный символ а, который принадлежит алфавиту А (пробел, или пустой символ);
3. конечное множество S, называемое множеством состояний;
4. некоторое выделенное состояние s, называемое начальным;
5. таблицу переходов, которая определяет поведение машины в зависимости от состояния и текущего символа;
6. некоторое подмножество F, вкладываемое в S, элементы которого называются заключительными состояниями (попав в такое состояние, машина останавливается).

Таблица переходов устроена следующим образом: для каждой пары (текущее состояние, текущий символ) указана тройка (новое состояние, новый символ, новый сдвиг). Здесь сдвиг одно из чисел -1 (влево), 0 (на месте) и 1 (направо). Таким образом, таблица переходов есть функция типа S x A => S x A x {-1, 0, 1}, определенная на тех парах, в которых состояние не является заключительным.

Остается описать поведение машины Тьюринга. В каждый момент имеется некоторая конфигурация, складывающаяся из содержимого ленты, текущей позиции головки (некоторое целое число) и текущего состояния машины (элемент S). Преобразование конфигурации в следующую происходит по естественным правилам: мы смотрим в таблице, что надо делать для данного состояния и для данного символа, то есть выясняем новое состояние машины, меняем символ на указанный и после этого сдвигаем головку влево, вправо или оставляем на месте. При этом, если новое состояние является одним из заключительных, работа машины заканчивается.

# Выполнение работы

1. Входные, выходные данные.
2. Описание работы программы.(КАК?)+вклад двух супер деятелей Оли и Данилы.??????Оо

# Текст программы

## Прорисовка и вывод строки:

#include "stdafx.h"

#include "MT(win32)a.h"

#include "MT(win32)aDlg.h"

#include <string>

#include "MT.h"

#ifdef \_DEBUG

#define new DEBUG\_NEW

#endif

// CMTwin32aDlg dialog

CDC\* pDC;

MT\* mt;

CMTwin32aDlg::CMTwin32aDlg(CWnd\* pParent /\*=NULL\*/)

: CDialog(CMTwin32aDlg::IDD, pParent)

, m\_edit(\_T(""))

, m\_sost(\_T(""))

, m\_kurs(\_T(""))

, m\_sost\_(\_T(""))

, m\_last\_sost(\_T(""))

, m\_line(\_T(""))

{

m\_hIcon = AfxGetApp()->LoadIcon(IDR\_MAINFRAME);

}

void CMTwin32aDlg::DoDataExchange(CDataExchange\* pDX)

{

CDialog::DoDataExchange(pDX);

DDX\_Control(pDX, IDC\_BUTTON1, button1);

DDX\_Text(pDX, IDC\_EDIT1, m\_edit);

DDX\_Control(pDX, IDC\_BUTTON2, bt2\_control);

DDX\_Text(pDX, IDC\_STATIC1, m\_kurs);

DDX\_Text(pDX, IDC\_STATIC2, m\_sost\_);

DDX\_Text(pDX, IDC\_STATIC3, m\_last\_sost);

DDX\_Text(pDX, IDC\_STATIC4, m\_line);

}

BEGIN\_MESSAGE\_MAP(CMTwin32aDlg, CDialog)

ON\_WM\_PAINT()

ON\_WM\_QUERYDRAGICON()

//}}AFX\_MSG\_MAP

ON\_BN\_CLICKED(IDC\_BUTTON1, &CMTwin32aDlg::OnBnClickedButton1)

ON\_WM\_CREATE()

ON\_BN\_CLICKED(IDC\_BUTTON2, &CMTwin32aDlg::OnBnClickedButton2)

ON\_WM\_CLOSE()

ON\_BN\_CLICKED(IDC\_BUTTON3, &CMTwin32aDlg::OnBnClickedButton3)

END\_MESSAGE\_MAP()

BOOL CMTwin32aDlg::OnInitDialog()

{

CDialog::OnInitDialog();

//

SetIcon(m\_hIcon, TRUE); // Set big icon

SetIcon(m\_hIcon, FALSE); // Set small icon

InitCommonControls();

return TRUE;

}

void CMTwin32aDlg::OnPaint()

{

PAINTSTRUCT pn;

CDC\* pDC1 = CMTwin32aDlg::BeginPaint(&pn);

pDC1->SetMapMode(MM\_ANISOTROPIC);

painting\_lines(pDC1);

pDC = pDC1;

}

HCURSOR CMTwin32aDlg::OnQueryDragIcon()

{

return static\_cast<HCURSOR>(m\_hIcon);

}

void CMTwin32aDlg::OnBnClickedButton1()

{

painting\_lines(pDC);

if (mt)

{

mt->analyze\_line();

std::string str1 = "asnbdsdbfhjsdfbsjdhbf";

pDC->SetMapMode(MM\_ANISOTROPIC);

print\_string(mt->get\_line(), pDC,mt->get('k'));

bt2\_control.EnableWindow(false);

UpdateData(true);

print\_sost();

UpdateData(false);

}

}

void CMTwin32aDlg::painting\_lines(CDC\* pDC)

{

CRect rectClient;

GetClientRect(rectClient);

CBrush brush, brush1;

CPen pen;

pDC->SetMapMode(MM\_ANISOTROPIC);

brush.CreateSolidBrush(RGB(0, 255, 0));

brush1.CreateSolidBrush(RGB(192, 192, 192));

pDC->SelectObject(brush1);

for(int i = 0; i <= rectClient.right; i = i + rectClient.right/20)

{

pDC->Rectangle(i,130,i + rectClient.right/20, 170);

if (i/10 == rectClient.right/20)

{

pDC->SelectObject(brush);

pDC->Rectangle(i,130,i + rectClient.right/20, 170);

pDC->SelectObject(brush1);

}

}

}

void CMTwin32aDlg::print\_string(std::string str, CDC\* pDC, int position)

{

CRect rectClient;

GetClientRect(rectClient);

CString s;

int step = rectClient.right/20;

int length = str.length();

int part\_0 = position, part\_1 = str.length() - position - 1, i = 0;

int x = 10\*step + 20 - step\*position;

while(length-- && str[i] != '\_')

{

pDC->SetBkColor(RGB(192, 192, 192));

if (i == position)

{

pDC->SetBkColor(RGB(0, 255, 0));

}

s = str[i];

pDC->TextOut(x,150,s);

x += step;

i++;

}

}

int CMTwin32aDlg::OnCreate(LPCREATESTRUCT lpCreateStruct)

{

if (CDialog::OnCreate(lpCreateStruct) == -1)

return -1;

return 0;

}

void CMTwin32aDlg::OnBnClickedButton2()

{

painting\_lines(pDC);

if (mt)

{

if (mt->get('c') < mt->get('l'))

{

mt->next\_step();

print\_string(mt->get\_line(), pDC,mt->get('k'));

}

else

{

print\_string(mt->get\_line(), pDC,mt->get('k'));

bt2\_control.EnableWindow(false);

}

UpdateData(true);

print\_sost();

UpdateData(false);

}

}

void CMTwin32aDlg::OnClose()

{

delete mt;

CDialog::OnClose();

}

void CMTwin32aDlg::OnBnClickedButton3()

{

painting\_lines(pDC);

UpdateData(true);

mt = new MT(m\_edit);

print\_string(mt->get\_line(), pDC,mt->get('k'));

bt2\_control.EnableWindow(true);

print\_sost();

UpdateData(false);

}

void CMTwin32aDlg::print\_sost(void)

{

CString num\_kurs, num\_sost, num\_last\_sost, line;

num\_kurs.Format("%d",mt->get('k') + 1);

num\_sost.Format("%d",mt->get('c'));

num\_last\_sost.Format("%d",mt->get('l'));

line = ConvertToCString(mt->get\_line());

m\_kurs = "Курсор: " + num\_kurs + "\n";

m\_sost\_ = "Состояние: " + num\_sost + "\n";

m\_last\_sost = "Последнее состояние: " + num\_last\_sost;

m\_line = "Лента: " + line;

}

CString CMTwin32aDlg::ConvertToCString(string s)

{

CString str = "";

for(int i = 0; i < s.size() && s[i] != '\_'; ++i)

str += s[i];

return str;

}

# Моделирование машины Тьюринга:

#include "StdAfx.h"

#include "MT.h"

#include <iostream>

#include <vector>

#include <string>

#include <string.h>

#include <fstream>

#include <stdlib.h>

using namespace std;

MT::MT(CString txt)

{

condition = kursor = last\_condition = 0;

try

{

int i = 0, count = 0;

ifstream ftxt(txt);

if (ftxt)

{

ftxt >> line;

for(int j = 0; j < 92 - line.size(); j++)

line += "\_";

string str, res;

int number\_0 = 0, number\_1 = 0;

while (!ftxt.eof())

{

number\_0 = 0;

number\_1 = 0;

//count = 0;

char input[50];

char\* buffer\_0 = new char[5];

char\* buffer\_1 = new char[5];

char symbol\_0 = ' ', symbol\_1 = ' ', symbol\_2 = ' ';

i = 1;

do //пропускаем пробелы

{

ftxt.getline(input, 50);

count++;

}

while(strlen(input) == 0);

count--;

str = input;

if (str.length() < 12 || str[i] < 48 || str[i] > 57)

throw count;

while (str[i] >= 48 && str[i] <= 57)

number\_0 = 10 \* number\_0 + atoi(&str[i++]);

symbol\_0 = str[++i];

i++;

while (str[i] < 48 || str[i] > 57)

i++;

while (str[i] >= 48 && str[i] <= 57)

number\_1 = 10 \* number\_1 + atoi(&str[i++]);

if (last\_condition < number\_1)

last\_condition = number\_1;

symbol\_1 = str[++i];

if (symbol\_1 == 'R' || symbol\_1 == 'L' || symbol\_1 == 'H')

{

symbol\_2 = symbol\_1;

symbol\_1 = '\_';

}

else

symbol\_2 = str[++i];

if (number\_0 >= static\_cast<int>(m.size()))

m.resize(number\_0 + 1);

if (number\_1 >= static\_cast<int>(m[number\_0].size()))

m[number\_0].resize(number\_1 + 1);

sprintf(buffer\_0, "%d", number\_0);

sprintf(buffer\_1, "%d", number\_1);

m[number\_0][number\_1] += (string)buffer\_0 + '\_' + symbol\_0 + '\_' + (string)buffer\_1 + '\_' + symbol\_1 + '\_' + symbol\_2;

}

}

else

{

cout << "!Error opening file!" << endl;

}

}

catch(int error)

{

cout << "!Error in line " << error << " !" << endl;

}

}

void MT::print()

{

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

{

while (m[i].size() == 0)

i++;

for(int j = 0; j < m[i].size(); ++j)

cout << m[i][j] << " ||| ";

cout << endl;

}

}

string MT::analyze\_string(const string& s)

{

int position = 2;

string res = "";

while (position < s.length())

{

if (line[kursor] == s[position])

{

for(int i = position - 2; i <= position + 6; ++i)

res += s[i];

return res;

}

position += 9;

}

return res;

}

void MT::analyze\_line()

{

string str;

while (condition != last\_condition)

{

str = "";

next\_step();

}

}

string MT::get\_line(void)

{

return line;

}

int MT::get(char a)

{

switch(a)

{

case 'k': return kursor;

case 'c': return condition;

case 'l': return last\_condition;

}

return -1;

}

void MT::next\_step()

{

string str = "";

for(int i = 0; i < m[condition].size() && str == ""; ++i)

if (m[condition][i] != "")

str = analyze\_string(m[condition][i]);

if (str != "")

{

line[kursor] = str[6];

if (str[8] == 'R')

kursor++;

if (str[8] == 'L')

kursor--;

if (kursor > line.size())

line.resize(kursor);

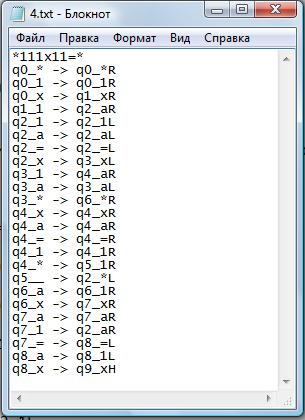
condition = atoi(&str[4]);

}

}

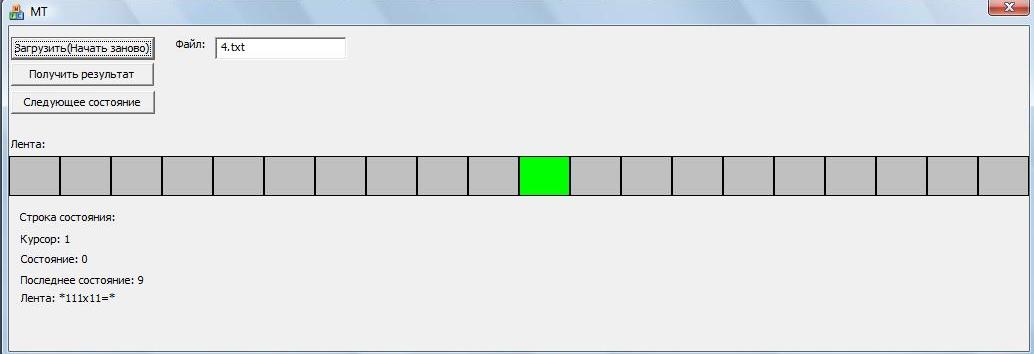
# Тестирование

Для тестирования прилагается файл 4.txt, который содержит в себе унарное умножение. Это входной файл, в котором находится список правил, и лента входных данных.



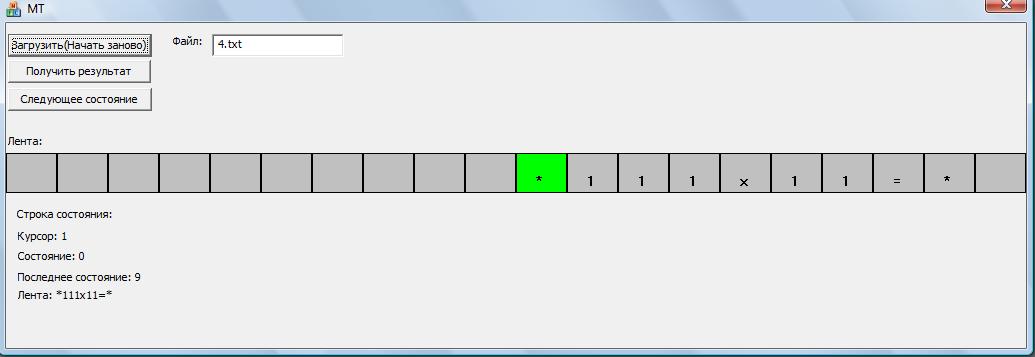
Исходя из этого теста, были получены результаты:

* На панели расположены три кнопки, лента и информационная зона, из которой мы узнаём положение курсора, состояние, последнее состояние.

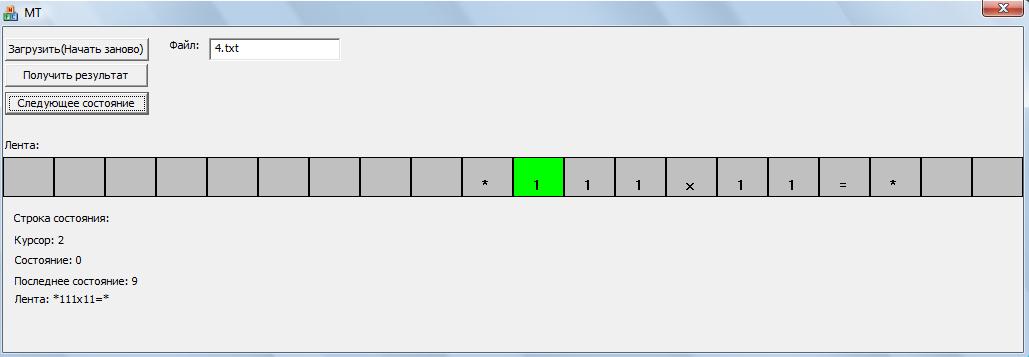


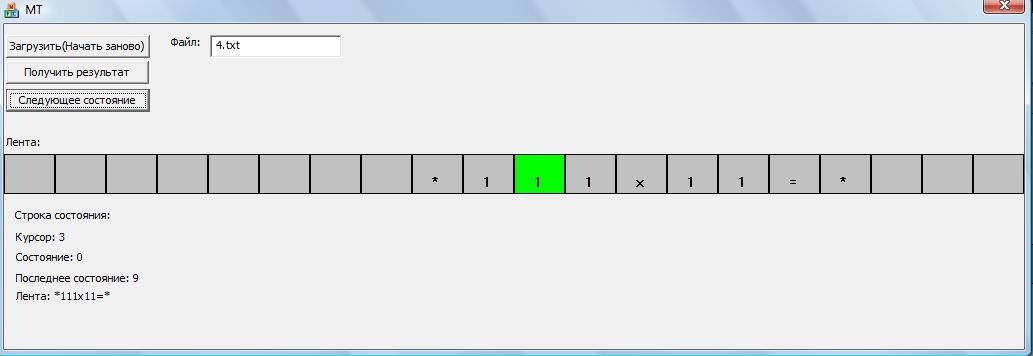
* Далее мы можем либо сразу нажать на вторую кнопку и получить искомый результат, либо отследить всю работу машины Тьюринга, при помощи кнопки «Следующее состояние».

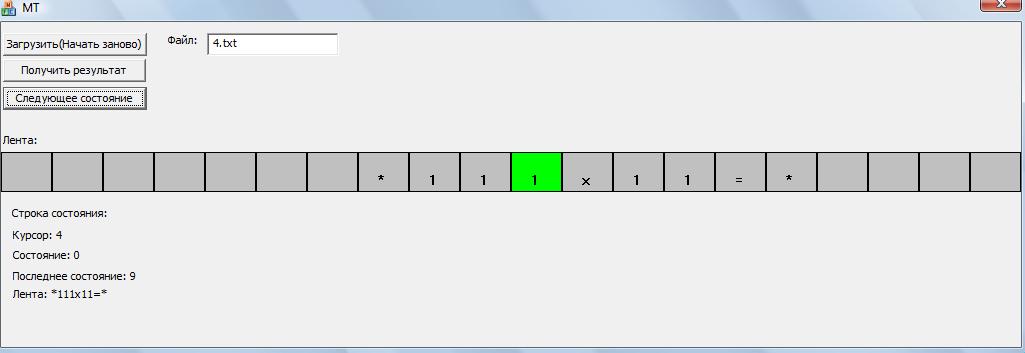
Приведённый ниже рисунок, характеризует начальное состояние машины Тьюринга.

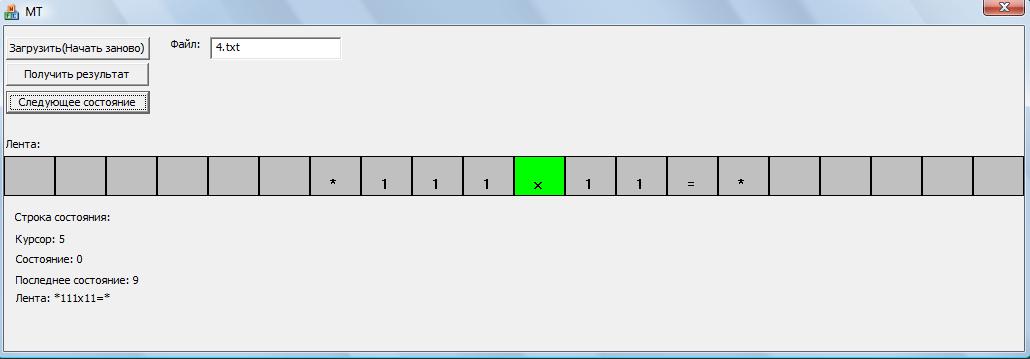


* Теперь просмотрим несколько состояний машины Тьюринга.

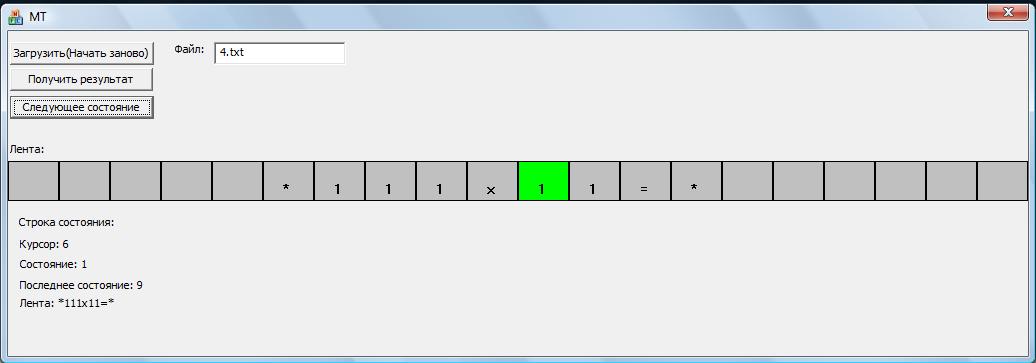




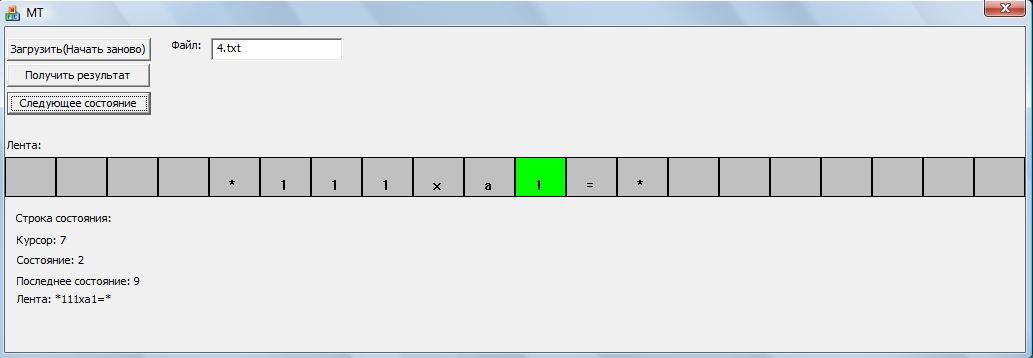




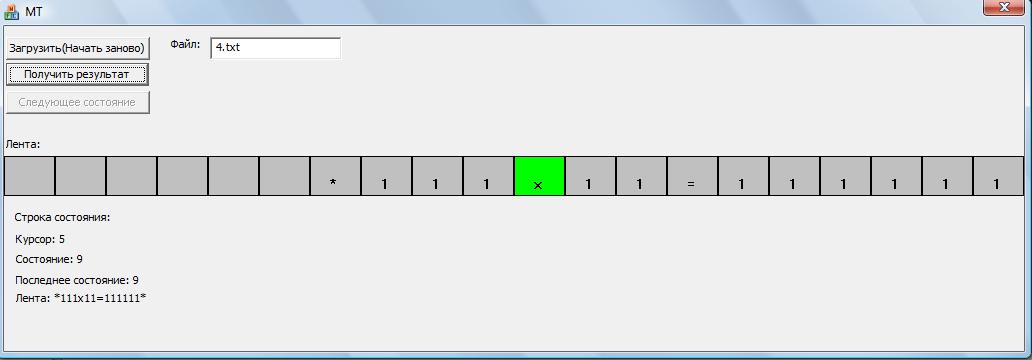
* Приведём пример, когда мы достигнем первого состояния.



Теперь второго:



* Всего будет 9 состояний, искомый результат – девятое состояние. Получаем его при помощи кнопки «Получить результат».



Тестирование подтвердило ожидаемые результаты, программа работает верно.

# Список литературы

1. Панюшкин Михаил «Изобретения Алана Тьюринга», 1997 г., 236 с.
2. Джон Хопкрофт, Раджив Мотвани, Джеффри Ульма «Введение в теорию машин Тьюринга», «Introduction to Automata Theory», 2002 г., 578с.
3. Карпов Юрий «Теория автоматов», 2001 г., 428 с.
4. Эббинхаус, Якобс, Ман «Машины Тьюринга и рекурсивные функции», 2003 г., 474 с.