Лабораторная работа № 4. Машина Тьюринга, как модель автоматных программ

ВВЕДЕНИЕ.

Программирование нуждается в новых универсальных алгоритмических моделях, а аппаратные средства реализуют алгоритмы не только в другой форме, но и на базе другой алгоритмической модели – автоматной. Заимствование технологии из сферы разработки аппаратных средств ключевая идея автоматного программирования. Однако синтез цифровых устройств отличается от программирования. Но, заимствуя модель, с одной стороны, ее не желательно существенно изменять, а, с другой стороны, нельзя не учитывать уже существующую теорию и практику программирования.

Далее мы рассмотрим SWITCH-технологию проектирования автоматных программ. С одной стороны, она так изменила модель конечного автомата, что фактически вывела ее за рамки теории автоматов. А, с другой стороны, вводит в программирование понятия, которые с трудом воспринимаются программистами, а, порой, просто являются лишними, т.к. существуют более привычные аналоги из теории программ и практики программирования.

1. Автоматизированные объекты, схемы программ

В лекции достижением автоматного программирования объявлено введение в него понятия автоматизированных объектов управления, заимствованное из теории автоматического управления (ТАУ). Но, напомним, что в ТАУ рассматривают не столько объекты, а системы, среди которых выделяют следующие [4]:

> ***Системы автоматического управления и автоматического контроля*** — это технические и природные системы, выполняющие свои функции автоматически, т. е. без осознанного участия человека.
>
> ***Автоматизированные системы*** — это системы, часть функций которых выполняется автоматически, а часть — оператором (организатором).

Исходя из этого, правильнее было бы говорить о системах автоматического управления (САУ). Теперь посмотрим на типовую функциональную схему САУ, показанную рис. 1. Если ленту машины Тьюринга считать объектом управления, то исполнительными устройствами (ИсУ) будут элементы МТ, реализующие изменение содержимого ленты и передвигающие головку, а измерительными устройствами (ИзУ) — элементы, читающие информацию с ленты.

- исполнительные устройства ИсУ;
- измерительные устройства ИзУ;
- устройства управления УУ.

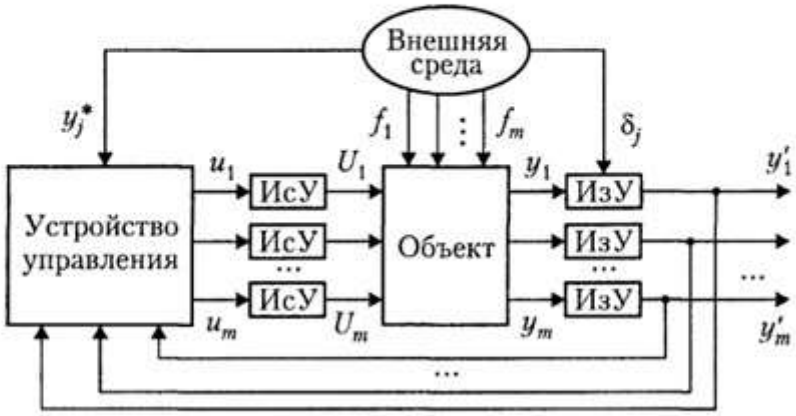

Рис.1. Функциональная схема САУ

Но зачем обращаться к ТАУ, если есть более близкая к программированию практика проектирования вычислительных систем, в которой операционные устройства (ОУ), к которым, безусловно, относится и МТ, рассматриваются в виде комбинации операционного (ОА) и управляющего (УА) автоматов. И это ближе к тому, к чему мы в конечном итоге стремимся – обоснованию мощности автоматного программирования. На рис. 2 представлен скрин текста из монографии Майорова С.А., Новикова Г.И. Структура электронных вычислительных машин [5], в которой вопросы проектирования ОУ рассмотрены весьма детально.

6.2. Концепция операционного и управляющего автоматов

В функциональном и структурном отношении операционное устройство разделяется на две части: операционный и управляющий автоматы (рис. 6.1). *Операционный автомат ОА* служит для хранения слов информации, выполнения набора микроопераций и вычисления значений логических условий, т. е. операционный автомат является структурой, организованной для выполнения действий над информацией. Микрооперации, реализуемые операционным автоматом, инициируются множеством *управляющих сигналов* $Y = \{y_1, \ldots, y_M\}$, с каждым из которых отождествляется определенная микрооперация. Значения логических условий, вычисляемые в операционном автомате, отображаются множеством *осведомительных сигналов* $X = \{x_1, \ldots, x_L\}$, каждый из которых отождествляется с определенным логическим условием. *Управляющий автомат УА* генерирует последовательность управляющих сигналов, предписанную микропрограммой и соответствующую значениям логических условий. Иначе говоря, управляющий автомат задает порядок выполнения действий в операционном автомате, вытекающий из алгоритма выполнения операций. Наименование операции, которую необходимо выполнить в устройстве, определяется кодом g операции. По отношению к управляющему автомату сигналы $g_1, \ldots, g_h$, посредством которых кодируется наименование операции, и осведомительные сигналы $x_1, \ldots, x_L$, формируемые в операционном автомате, играют одинаковую роль: они влияют на порядок выработки управляющих сигналов Y. Поэтому сигналы $g_1, \ldots, g_h$ и $x_1, \ldots, x_L$ относятся к одному классу — к классу осведомительных сигналов, поступающих на вход управляющего автомата.

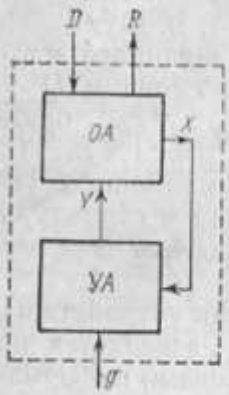

Рис. 6.1. Структура операционного устройства

Таким образом, любое операционное устройство — процессор, канал ввода—вывода, устройство управления внешним устройством — является композицией операционного и управляющего автоматов. Операционный автомат, реализуя действия над словами информации, является исполнительной частью устройства, работой которого управляет управляющий автомат, генерирующий необходимые последовательности управляющих сигналов.

Рис.2. Концепция управляющего и операционного автоматов

Но, если сравнивать теорию проектирования ЭВМ и теорию программ, то между ними прослеживается явная структурная аналогия. В теории программирования модель любой программы на структурном уровне можно представить, как схему программы $S = (M, A, C)$, где M – множество элементов памяти, A – множество операторов, C – управление. Следуя этому подходу, любую программу машины Тьюринга также можно определить, как схему программы, в которой множество M представлено ячейками ленты, множество операторов – действиями МТ, связанными 1) с анализом ячеек, 2) изменением символов в ячейках ленты и 3) перемещением головки.

Таким образом понятие схемы программы полностью аналогично рассмотренной концепции операционного и управляющего автоматов, где моделью УА является модель рассматриваемого далее структурного конечного автомата (СКА), а ОА «является структурой для выполнения действий над информацией». При этом ОА включает элементы хранения данных (выше – это память) и блоки для обработки информации, реализующих вычисление логических условий и реализацию тех или иных действий (выше – множество операторов).

Из сказанного можно понять, что лента лишь условно может считаться объектом управления для МТ. Хотя бы потому, что устройство управления машины Тьюринга не имеет к ней прямого доступа, т.к. все операции с ячейками реализуют опосредованно блоки ОА. Кроме того, как представляется, не очень привычно или, если не сказать, странно считать целью управления программы, как системы управления, объект, представляющий собой память (ленту).

Таким образом, для формального определения машины Тьюринга, а в ее контексте места для модели конечного автомата, достаточно понятий теории программ. Теперь, в отличие от весьма расплывчатого определения автоматных программ, данного в рамках SWITCH-технологии, можно сказать, что автоматной программой считается программа, имеющая управление в форме модели конечного автомата.

Какая при этом будет сама программа – с простым или сложным поведением, какова ее «разновидность» – с логическим управлением, «с явным выделением состояний» и т.д. и т.п. не имеет абсолютно значения. Главное – вид управления. Остальные же элементы программы могут определяться в широких пределах – от простейших, как, например, у машины Тьюринга, до самых сложных – любых форм операторов, функций и структур данных языков программирования – ассемблера, языка высокого уровня и т.п.

Можно также вспомнить, что машину Тьюринга уже давно принято считать автоматом или уж, в крайнем случае, его простым расширением. Но необходимо понимать, что это за автомат, что это за расширение, и эквивалентны ли они моделям классических конечных автоматов. Попробуем это как раз и уточнить.

2. Тьюрингово программирование в среде автоматного программирования

На рис. 3 показан автомат для МТ функции инкремента из монографии [8]. По форме это явно не программа для МТ, но уже и не классический конечный автомат. На

рис. 4 приведен граф классического структурного конечного автомата (СКА) и его реализация в среде ВКПа (среда визуально-компонентного автоматного программирования на языке C++ в рамках библиотеки Qt и среды Qt Creator), реализующего тот же самый алгоритм блока управления (БУ) МТ.

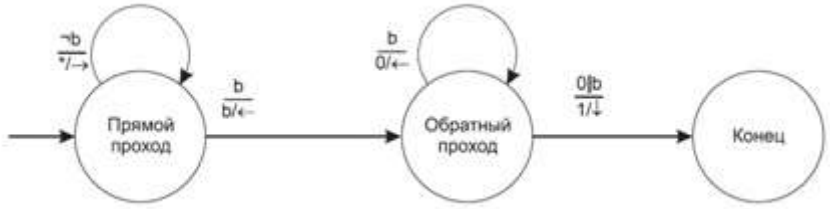

Рис.3. Увеличение числа на единицу с помощью машины Тьюринга

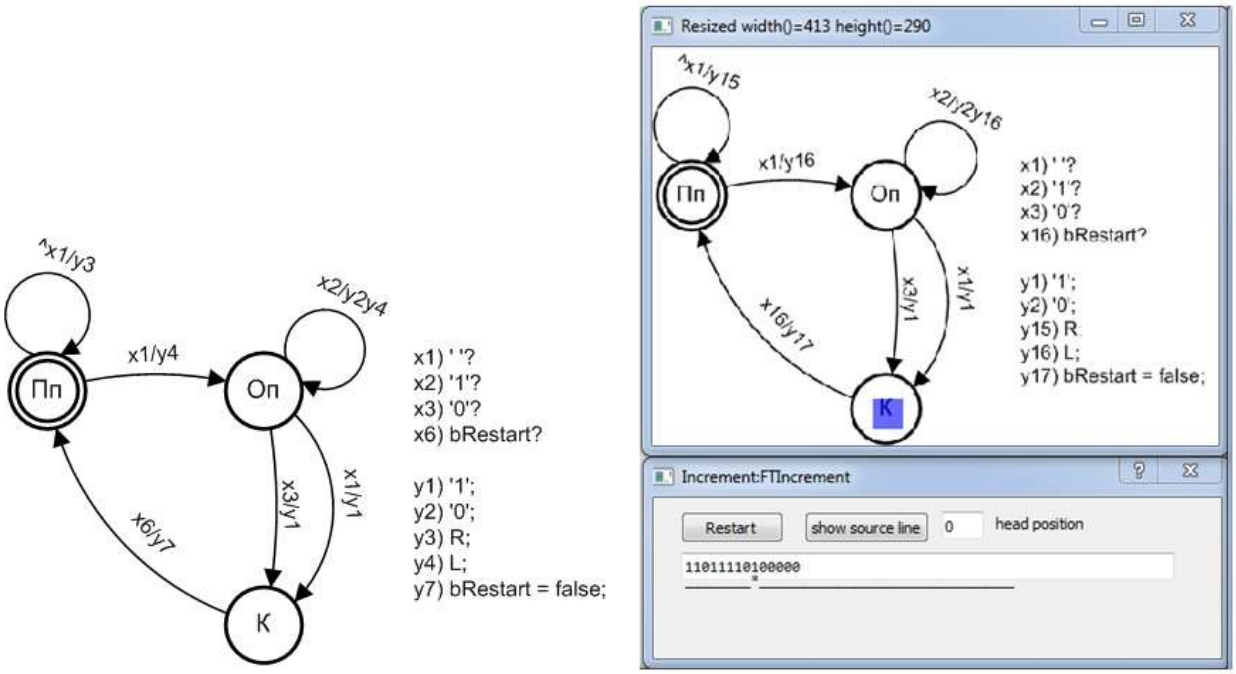

Рис.4 Модель программы инкремента для МТ в форме СКА

Можно видеть, что структурный автомат имеет четыре входных и пять выходных каналов. Каждый из этих каналов связан с одноименной ему программной функцией – предикатом или действием. Здесь предикаты – функции без параметров, возвращающие булевское значение в зависимости от значения обозреваемой ими ячейки ленты, а действия – функции без параметров, выполняющие то или иной действие по изменению ячейки ленты и передвижению головки машины Тьюринга.

Данный СКА имеет то же множество состояний, что и автомат на рис.3. При этом кроме собственно автоматного отображения, представленного СКА, он реализует еще два отображения – отображение множества предикатов (x_1, …, x_M) на множество одноименных входных каналов автомата, и множество выходных каналов автомата на

множество одноименных действий – y_1, …, y_N. Например, предикат x_3 вернет значение true (значение 1 для одноименного входного сигнала), если в текущей ячейке будет символ 1, а действие y_4, которое будет запущено, когда одноименный выходной сигнал автомата примет значение 1, будет соответствовать перемещение головки влево (L) и т.д. и т.п.

Обратим внимание, что СКА не управляет напрямую лентой, а реализует [дополнительные] отображения, связывая сигналы автомата с функциями, определяющими множество операций машины Тьюринга. Это еще раз убеждает, что нет необходимости вводить понятие автоматизированного объекта управления в ситуации, когда достаточно «старомодного», но математически строго понятия отображение.

Сравнивая автоматы на рис. 3 и рис. 4, можно видеть, что СКА не использует команду «*» (см. рис. 1). В подобной ситуации ему достаточно не выдавать сигнал, связанный с данной командой. Кроме того, два и более сигналов (как входных, так и выходных) на одном переходе параллельны. Поэтому, когда возникает конфликт доступа к общим объектам (например, необходимо изменить ячейку и переместить головку) используется соглашение: действия на одном переходе исполняются последовательно в порядке следования их номеров, т.е. действие с большим номером выполняется после действия с меньшим номером. Это соглашение не распространяется на предикаты, т.к. они не изменяют ленту. Так мы делаем автомат более компактным и наглядным (не надо вводить промежуточные состояния).

В процессе тестирования программы инкремента были выявлены ситуации, когда в процессе работы МТ могут возникнуть проблемы. Во-первых, реальная лента не бесконечна и выход за ее пределы может вызвать «крах» программы. Во-вторых, нужно обязательно указывать начальное положение головки. Без этого, если, например, число находится в произвольном месте ленты, а начальное состояние головки слева от числа и напротив пробела, то головка сразу начнет движение влево. Далее она может выйти за границы ленты, вызвав «крах» программы, или, переместившись, на шаг влево запишет в ячейку 1 и, зависнув, завершит «успешно» работу. Или, если число содержит во всех разрядах 1 и записано с начала ленты, то заключительная попытка переноса 1 в старший разряд вызовет тот же самый «крах».

2.1. Объектная реализация МТ на языке С++

Рассмотрим объектную программную реализацию машины Тьюринга на языке С++ в среде ВКПа, реализующую любую программу для МТ, в том числе и программу вычисления инкремента.

С этой целью создан базовый класс, представляющий любую машину Тьюринга, который и наследуют программные объекты, реализующие ту или иную программу МТ. Данный базовый класс представлен на листинге 1, а программа, реализующая задачу инкремента, на листинге 2.

Листинг 1. Программная реализация базового класса МТ

```
#include "lfsaappl.h"
class FTuringMashine : public LFsaAppl
{
public:
    FTuringMashine(string strNam, CVarFsaLibrary *pCVFL, LArc* pTBL);
protected:
    int x15(); int x16();
    void y14(); void y15(); void y16(); void y17();
    QString strSrc;             // исходное состояние ленты
    QString strTape;            // лента
    QString strHead;            // головка
    int nIndexHead{0};          // индекс головки
    bool bRestart{false};       // признак рестарта
    int nHeadPosition{0};       // текущее положение головки
};

#include "stdafx.h"
#include "FTuringMashine.h"
FTuringMashine::FTuringMashine(string strNam, CVarFsaLibrary *pCVFL, LArc*
pTBL):
    LFsaAppl(pTBL, strNam, nullptr, pCVFL)
{
    nHeadPosition = 0;
    strHead = "_____________________________________________";
    nIndexHead = nHeadPosition;
}
//============================================================
// ПРЕДИКАТЫ
// пустой символ?
int FTuringMashine::x15() { return strTape[nIndexHead] == '#'; }
// рестарт?
int FTuringMashine::x16() { return bRestart; }
//============================================================
```

```
// ДЕЙСТВИЯ
// записать пустой символ на ленту
void FTuringMashine::y14() { strTape[nIndexHead] = '#'; }
// головка сместить вправо (нарастить индекс)
void FTuringMashine::y15() { nIndexHead++; }
// головка сместить влево (уменьшить индекс)
void FTuringMashine::y16() { nIndexHead--; }
// установить в исходное состояние
void FTuringMashine::y17() {
    strTape = strSrc; nIndexHead = 0;
    bRestart = false;
    nIndexHead = nHeadPosition;
}
```

Листинг 2. Программа инкремента для машины Тьюринга

```
#include "FTuringMashine.h"
class FTIncrement : public FTuringMashine
{
public:
    LFsaAppl* Create(CVarFSA *pCVF) {
       Q_UNUSED(pCVF)return new FTIncrement(nameFsa, pCVarFsaLibrary);
    }
    FTIncrement(string strNam, CVarFsaLibrary *pCVFL);
protected:
    int x1(); int x2(); int x3();
    void y1(); void y2();
};

#include "FTIncrement.h"
static LArc TBL_TIncrement[] = {
// см. Н. Поликарпова, А. Шалыто Автоматное программирование, 2-е издание, 2011
г.,
// стр.17-18
//===== функция переходов МТ (см. Ю.Г. Карпов Теория автоматов, - СПб.: Питер,
2003. - 208 с.) ==============
//     f(Пп,^` `) = (Пп,`*`,R)
//     f(Пп,` `)  = (Оп,` `,L)
//     f(Оп,`1`)  = (Оп,`0`,L)
//     f(Оп,` `)  = (К,`1`,R)
//     f(Оп,`0`)  = (К,`1`,R)
//=======================================
       LArc("Прямой проход",   "Прямой проход",   "^x1", "y15"),
       LArc("Прямой проход",        "Обратный проход", "x1",  "y16"),
       LArc("Обратный проход", "Обратный проход", "x2",  "y2y16"),
       LArc("Обратный проход", "Конец",           "x1",  "y1"),
```

```
        LArc("Обратный проход", "Конец",           "x3",  "y1"),
        LArc("Конец",           "Прямой проход",   "x16", "y17"),
        LArc()
};

FTIncrement::FTIncrement(string strNam, CVarFsaLibrary *pCVFL):
    FTuringMashine(strNam, pCVFL, TBL_TIncrement)
{
    strSrc = "11011110011111 ";
    strTape = strSrc;
}
// ПРЕДИКАТЫ
int FTIncrement::x1() { return strTape[nIndexHead] == ' '; }
int FTIncrement::x2() { return strTape[nIndexHead] == '1'; }
int FTIncrement::x3() { return strTape[nIndexHead] == '0'; }
// ДЕЙСТВИЯ
void FTIncrement::y1() { strTape[nIndexHead] = '1'; }
void FTIncrement::y2() { strTape[nIndexHead] = '0'; }
```

2.2. Примеры программ для МТ с реализацией на C++

Рассмотрим пример программы для МТ, которая «выступает как акцептор языка, т.е. она может распознавать язык» из [9]. Ее функция переходов представлена на рис. 5, а эквивалентный ей автомат в форме СКА на рис. 6.

δ(1, a) = (2, x, R)	δ(1, y) = (4, y, R)
δ(2, a) = (2, a, R)	δ(2, y) = (2, y, R)
δ(2, b) = (3, y, L)	δ(3, y) = (3, y, L)
δ(3, a) = (3, a, R)	δ(3, x) = (1, x, R)
δ(4, y) = (4, a, R)	δ(4, #) = (F, #, L)

Рис. 5. Функция переходов машины Тьюринга, распознающая язык {anbn: n≥1}

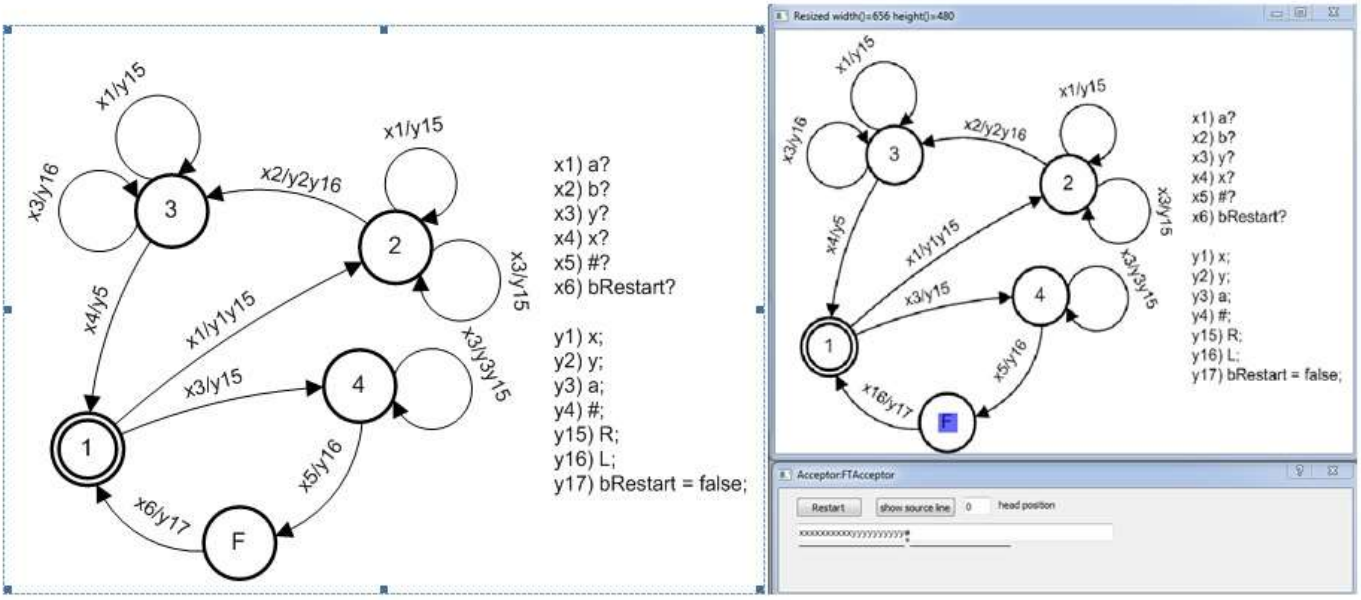

Рис. 6. Граф СКА машины Тьюринга, распознающей язык {anbn: n≥1)

Блок управления МТ в форме СКА имеет 6 входных и 7 выходных каналов. Программа акцептора включает также соответствующее им число предикатов и действий, которые представлены на рисунке справа от графа автомата. Реализация программы на C++ в среде ВКПа представлена на листинге 3.

Листинг 3. Программа для машины Тьюринга, распознающей язык {anbn: n≥1}

```
#include "FTuringMashine.h"

extern LArc TBL_TAcceptor[];
class FTAcceptor : public FTuringMashine
{
public:
    LFsaAppl* Create(CVarFSA *pCVF) { Q_UNUSED(pCVF)return new
FTAcceptor(nameFsa, pCVarFsaLibrary); }
    FTAcceptor(string strNam, CVarFsaLibrary *pCVFL, LArc* pTB =
TBL_TAcceptor);
protected:
    int x1(); int x2(); int x3(); int x4();
    void y1(); void y2(); void y3(); void y18();
    int nState{1};
friend class CDlgTAcceptor;
};

#include "stdafx.h"
#include "FTAcceptor.h"

LArc TBL_TAcceptor[] = {
// см. Дж.Макконнелла Анализ алгоритмов. Активный обучающий подход. 2013 г.,
// Машина Тьюринга как акцептор языка, стр.304
```

```
//===== функция переходов МТ ==============
//     f(1,a) = (2,x,R)   f(1,y) = (4,y,R)
//     f(2,a) = (2,x,R)   f(2,y) = (2,y,R)
//     f(2,b) = (2,x,R)   f(3,y) = (3,y,L)
//     f(3,a) = (3,a,R)   f(3,x) = (1,x,R)
//     f(4,y) = (4,a,R)   f(4,#) = (F,#,L)
//=========================================
    LArc("1",         "2","x1",  "y1y15"),  // 1,a,2,x,R
    LArc("1",         "4","x3",  "y15"),          // 1,y,4,R
    LArc("2",         "2","x1",  "y15"),          // 2,a,2,R
    LArc("2",         "3","x2",  "y2y16"),  // 2,b,3,y,L
    LArc("2",         "2","x3",  "y15"),          // 2,y,2,R
    LArc("3",         "3","x1",  "y16"),          // 3,a,3,L
    LArc("3",         "3","x3",  "y16"),          // 3,y,3,L
    LArc("3",         "1","x4",  "y15"),          // 3,x,1,R
    LArc("4",         "4","x3",  "y2y15"),  // 4,y,4,a,R
    LArc("4",         "F","x15", "-"),      // 4,#,F,-,-
    LArc("F",         "1","x16", "y17"),      //
    LArc("1",         "1","x16", "y17"),      //
    LArc("2",         "1","x16", "y17"),      //
    LArc("3",         "1","x16", "y17"),      //
    LArc("4",         "1","x16", "y17"),      //
//    LArc("1",       "1","--",  "y18"),         //
    LArc()
};

FTAcceptor::FTAcceptor(string strNam, CVarFsaLibrary *pCVFL, LArc* pTB):
    FTuringMashine(strNam, pCVFL, pTB)
{
    strSrc = "aaaaaaaaaabbbbbbbbbb#";
    strTape = strSrc;
}

int FTAcceptor::x1() { return strTape[nIndexHead] == 'a'; }
int FTAcceptor::x2() { return strTape[nIndexHead] == 'b'; }
int FTAcceptor::x3() { return strTape[nIndexHead] == 'y'; }
int FTAcceptor::x4() { return strTape[nIndexHead] == 'x'; }

void FTAcceptor::y1() { strTape[nIndexHead] = 'x'; }
void FTAcceptor::y2() { strTape[nIndexHead] = 'y'; }
void FTAcceptor::y3() { strTape[nIndexHead] = 'a'; }
void FTAcceptor::y18() {
    switch(nState) {
    case 1:
        if (x1()) { nState = 2; y1(); y5(); break; }
        if (x3()) { nState = 4; y5();  break; }
        break;
```

```
    case 2:
        if (x1()) { nState = 2; y5(); break; }
        if (x2()) { nState = 3; y2();y6(); break; }
        if (x3()) { nState = 2; y5(); break; }
        break;
    case 3:
        if (x1()) { nState = 3; y6(); break; }
        if (x3()) { nState = 3; y6(); break; }
        if (x4()) { nState = 1; y5(); break; }
        break;
    case 4:
        if (x3()) { nState = 4; y2(); y5(); break; }
        if (x5()) { nState = 5; break; }
        break;
    case 5:
        if (x6()) { y7(); nState = 1; break; }
        break;
    }
    }
```

На листинге 3 действие y_{18} представляет вариант программы для МТ в соответствии с подходом SWITCH-технологии. В рамках реализации автоматного программирования среды ВКПа в этом случае вместо автомата на рис. 6 необходимо будет реализовать автомат с одним состоянием, который выдает в цикле сигнал y18. Ему соответствует закомментированная строка таблицы переходов на листинге 3. Для работы автомата по типу SWICH необходимо снять комментарий с данной строки, а остальные строки закомментировать.

Рассмотрим еще один пример программы для машины Тьюринга, где МТ определяется, как «весьма простое расширение модели конечного автомата». В этом случае программа для машины Тьюринга представляет собой конечный список пятерок частично-определенной функции переходов и выходов δ: $S\times X\rightarrow S\times X\times \Gamma$.

Программа для МТ, находящая наибольший общий делитель (НОД) двух чисел, показана на рис. 7. Эквивалентный ей граф СКА представлен на рис. 8. Заметим, что и здесь не используются команда перезаписи символов. Реализация на C++ представлен листингом 4.

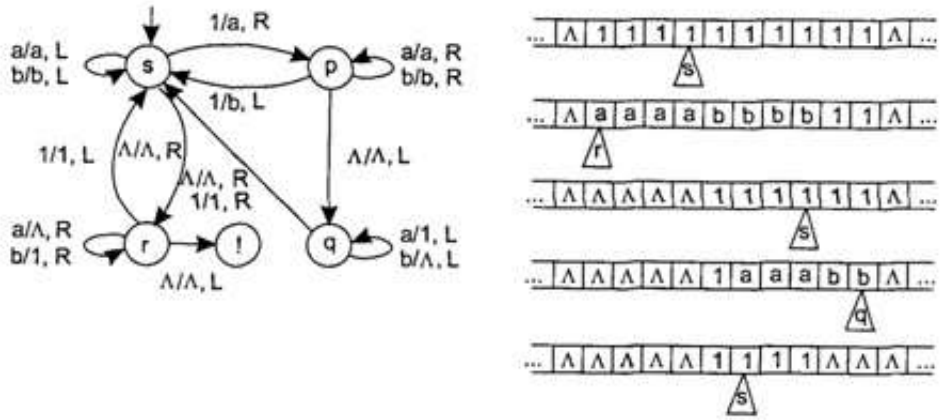

Рис. 7. Граф переходов машины Тьюринга, вычисляющей НОД двух чисел, и несколько ее конфигураций при обработке пары чисел <4, 6>

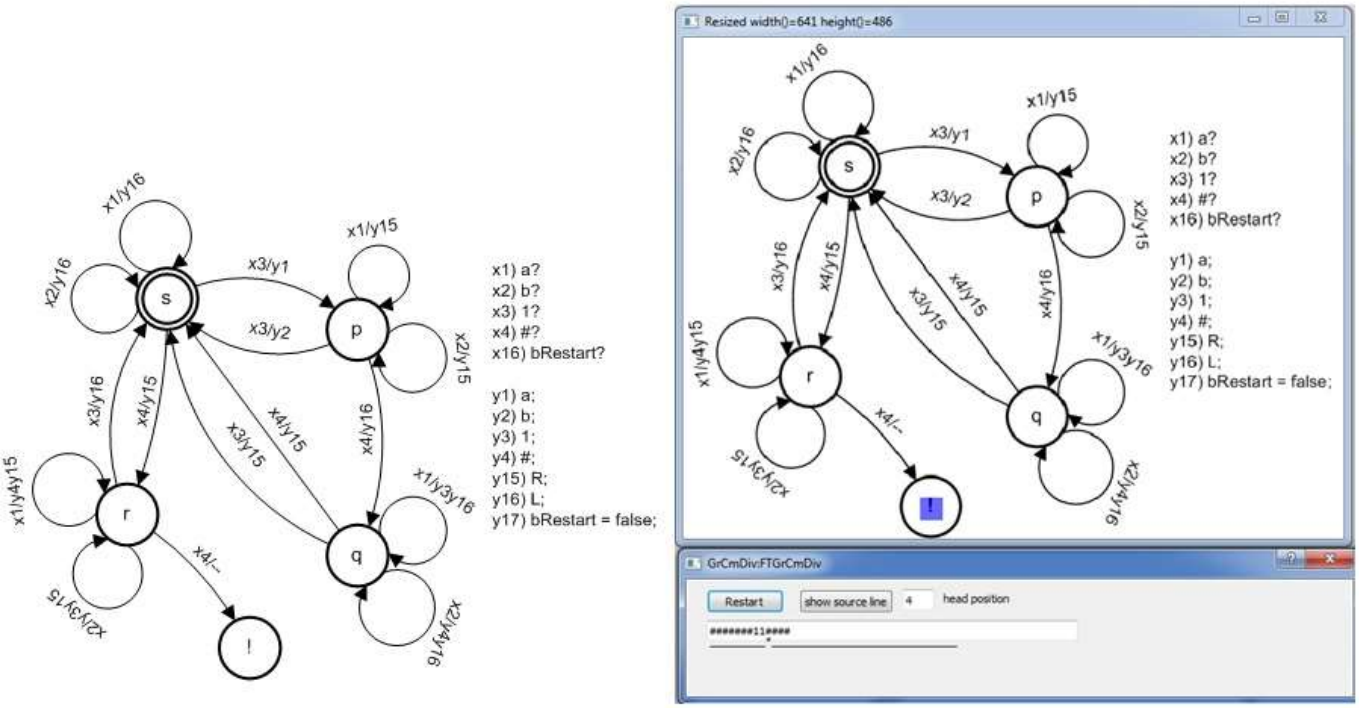

Рис. 8. Граф СКА, эквивалентный графу на рис. 7

Листинг 4. Программа для машины Тьюринга нахождения НОД двух чисел

```
#include "FTuringMashine.h"

class FTGrCmDiv: public FTuringMashine
{
public:
    LFsaAppl* Create(CVarFSA *pCVF) { Q_UNUSED(pCVF)return new
FTGrCmDiv(nameFsa, pCVarFsaLibrary); }
    FTGrCmDiv(string strNam, CVarFsaLibrary *pCVFL);
protected:
    int x1(); int x2(); int x3(); int x4();
    void y1(); void y2(); void y3(); void y17();
};
```

```
#include "FTGrCmDiv.h"
static LArc TBL_TGrCmDiv[] = {
//===== программа МТ нахождения НОД (Greatest Common Divider) ==============
// см. Ю.Г. Карпов Теория автоматов, - СПб.: Питер, 2003. - 208 с.
// стр.194
// см. Трахтенброт Б.А. Алгоритмы и вычислительные автоматы. М.: Советское
радио, 1974, - 200с.
// стр.76, 84-87
    LArc("s","s","x1",     "y16"),          //
    LArc("s","s","x2",     "y16"),          //
    LArc("s","p","x3",     "y1"),           //
    LArc("s","r","x15",    "y15"),          //
    LArc("p","p","x1",     "y15"),          //
    LArc("p","p","x2",     "y15"),          //
    LArc("p","s","x3",     "y2"),           //
    LArc("p","q","x15",    "y16"),          //
    LArc("q","q","x1",     "y3y16"),  //
    LArc("q","q","x2",     "y14y16"), //
    LArc("q","s","x3",     "y15"),          //
    LArc("q","s","x15",    "y15"),          //
    LArc("r","r","x1",     "y14y15"), //
    LArc("r","r","x2",     "y3y15"),  //
    LArc("r","s","x3",     "y16"),          //
    LArc("r","!","x15",    "--"),           //
    LArc("!","s","x16",    "y17"),          //
    LArc()
};

FTGrCmDiv::FTGrCmDiv(string strNam, CVarFsaLibrary *pCVFL):
    FTuringMashine(strNam, pCVFL, TBL_TGrCmDiv)
{
    nHeadPosition = 4;
    strSrc = "#1111111111## ";
    strTape = strSrc;
    nIndexHead = nHeadPosition;
}

int FTGrCmDiv::x1() { return strTape[nIndexHead] == 'a'; }
int FTGrCmDiv::x2() { return strTape[nIndexHead] == 'b'; }
int FTGrCmDiv::x3() { return strTape[nIndexHead] == '1'; }
int FTGrCmDiv::x4() { return strTape[nIndexHead] == '#'; }

void FTGrCmDiv::y1() { strTape[nIndexHead] = 'a'; }
void FTGrCmDiv::y2() { strTape[nIndexHead] = 'b'; }
void FTGrCmDiv::y3() { strTape[nIndexHead] = '1'; }
void FTGrCmDiv::y17() {
```

```
    strTape = strSrc; nIndexHead = 4;
    bRestart = false;
    nIndexHead = nHeadPosition;
    }
```

И еще одна программа для МТ от разработчиков SWITH-технологии, для реализации задачи распознавание скобок в двух варианта. Один в форме автомата Мили, второй – смешанный автомат (соответственно на рис. 9 и рис. 11). Соответствующие им структурные автоматы приведены на рис. 10 и рис. 12. Реализацию программы на C++ демонстрирует листинг 5.

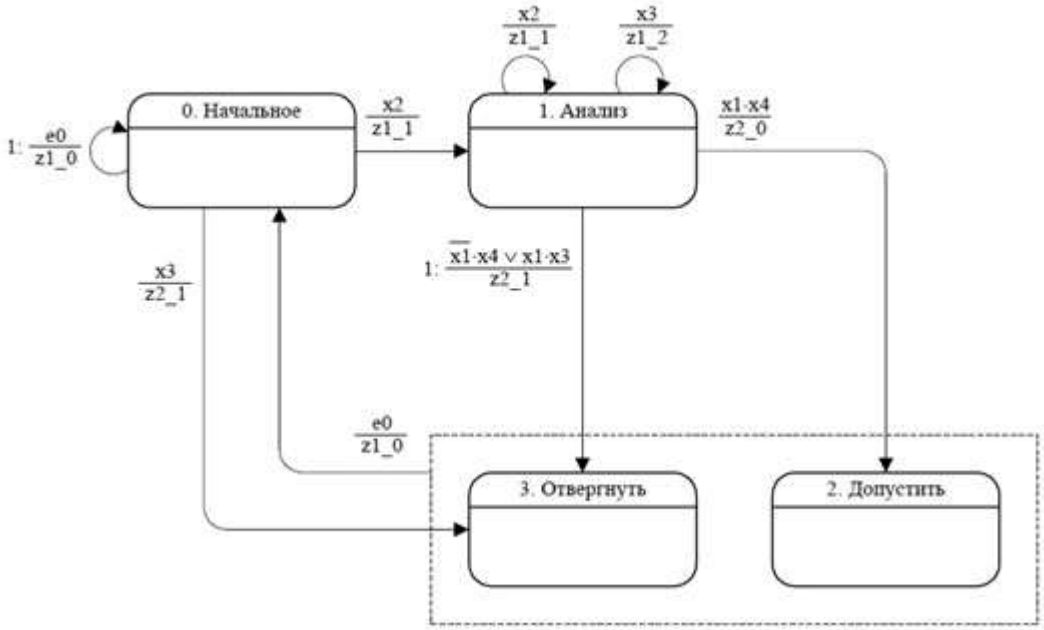

Рис. 9. Распознавание скобок произвольной глубины. Граф переходов Мили

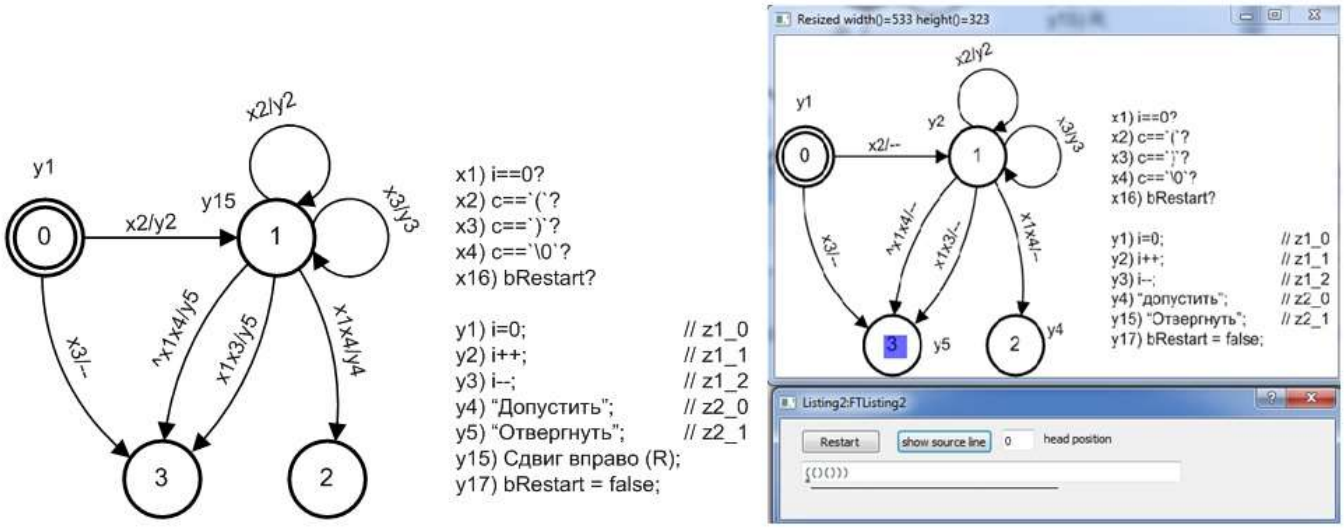

Рис. 10. Распознавание скобок произвольной глубины. Граф СКА Мили

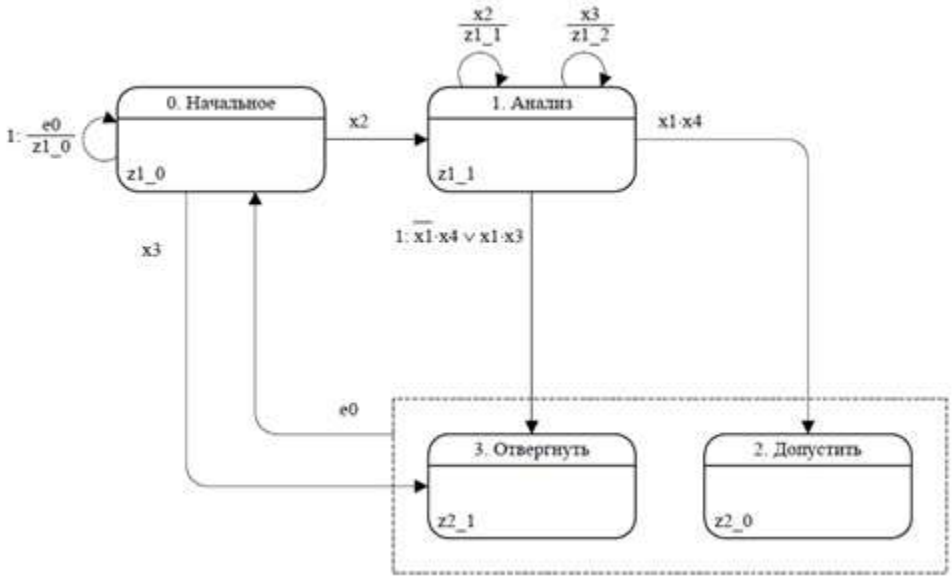

Рис. 11. Распознавание скобок произвольной глубины. Граф переходов смешанного автомата

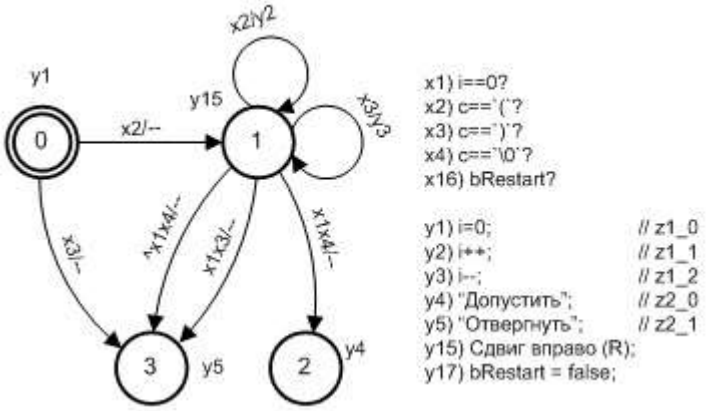

Рис. 12. Распознавание скобок произвольной глубины. Граф СКА переходов смешанного автомата

Листинг 5. Программа для машины Тьюринга распознавания вложенности скобок

```
#include "../FTuringMashine.h"

class FTListing2 : public FTuringMashine
{
public:
    void MooreAction();
    LFsaAppl* Create(CVarFSA *pCVF) { Q_UNUSED(pCVF)return new
FTListing2(nameFsa, pCVarFsaLibrary); }
    FTListing2(string strNam, CVarFsaLibrary *pCVFL);
protected:
    int x1(); int x2(); int x3(); int x4();
    void y1(); void y2(); void y3(); void y4(); void y5();
    int i{0};
};

#include "FTListing2.h"
```

```
static LArc TBL_TListing2[] = {
// см. Туккель Н.И., Шалыто А.А. От Тьюрингова программирования к автоматному, МирПК, №2, С.144-149
//===== функция переходов МТ (см. Ю.Г. Карпов Теория автоматов, - СПб.: Питер, 2003. - 208 с.) ==============
//     f(Пп,^` `) = (Пп,`*`,R)
//     f(Пп,` `)  = (Оп,` `,L)
//     f(Оп,`1`)  = (Оп,`0`,L)
//     f(Оп,` `)  = (К,`1`,R)
//     f(Оп,`0`)  = (К,`1`,R)
//=========================================
/*
// Мили
    LArc("0",         "1",    "x2",    "y2"),     // '(';движение вправо
    LArc("0",         "3",    "x3",    "--"),     // '(';
    LArc("1",         "1",  "x2",      "y2"),     // '(';движение вправо
    LArc("1",         "1",  "x3",      "y3"),     // ')';движение влево
    LArc("1",         "3",  "^x1x4",   "--"),     // i!=0;' '; Отвергнуть
    LArc("1",         "3",  "x1x3",    "--"),     // i==0;')'; Отвергнуть
    LArc("1",         "2",  "x1x4",    "--"),     // i==0;' '; Допустить
    LArc("2",         "0",  "x16",     "y17"),    // bRestart; перезапуск
    LArc("3",         "0",  "x16",     "y17"),    // bRestart; перезапуск
*/
//*
// Смешанный автомат Мили-Мура
    LArc("0",         "1",  "x2",      "y2"),          // '('
    LArc("0",         "3",  "x3",      "--"),          // ')'
    LArc("1",         "1",  "x2",      "y2"),          //'(';движение вправо
    LArc("1",         "1",  "x3",      "y3"),          // ')';движение влево
    LArc("1",         "2",  "x1x4",    "--"),      // i==0;' ';
    LArc("1",         "3",  "^x1x4",   "--"),          // i!=0;' ';
    LArc("1",         "3",  "x1x3",    "--"),          // i==0;')';
    LArc("2",         "0",  "x16",     "y17"),     // bRestart; перезапуск
    LArc("3",         "0",  "x16",     "y17"),     // bRestart; перезапуск
//*/
    LArc()
};

FTListing2::FTListing2(string strNam, CVarFsaLibrary *pCVFL):
    FTuringMashine(strNam, pCVFL, TBL_TListing2)
{
    strSrc = "(()()) ";
    strTape = strSrc;
}
// ПРЕДИКАТЫ
int FTListing2::x1() { return i == 0; }
int FTListing2::x2() { return strTape[nIndexHead] == '('; }     //
```

```
int FTListing2::x3() { return strTape[nIndexHead] == ')'; }     //
int FTListing2::x4() { return strTape[nIndexHead] == ' '; }     //
// ДЕЙСТВИЯ
void FTListing2::y1() { i = 0; }                    // z1_0
void FTListing2::y2() { i++; }                      // z1_1
void FTListing2::y3() { i--; }                      // z1_2
void FTListing2::y4() { strTape = "допустить"; }    // z2_0
void FTListing2::y5() { strTape = "отвергнуть"; }   // z2_1

void FTListing2::MooreAction()
{
    string strState = FGetState();
    if (strState=="0")     { y1(); }           // сброс счетчика
    else if (strState=="1")     { y15(); }      // сдвиг головки влево
    else if (strState=="2")     { y4(); }       // Допустить
    else if (strState=="3")     { y5(); }       // Отвергнуть
    }
```

Поскольку автомат на рис. 12 отказался работать, то было решено перейти к автомату на рис. 9. Эквивалентный ему автомат в форме СКА, показан на рис. 10. Правда, формально это тоже смешанный автомат, у которого от первой реализации (рис. 12) был оставлен сигнал при состоянии «0» и сигнал y15 при состоянии «1». Первый необходим при начальной установке, а сигнал y15 реализует смещение головки вправо в целях чтения очередного символа ленты. В остальном СКА соответствует автомату Мили на рис. 9.

После того, как автомат на рис. 10 был успешно протестирован, вернулись к автомату на рис. 11. И стало понятно, что у него лишним является сигнал z1_1 при состоянии «1» (у автомата на рис. 12 это сигнал y_2). Проблема в том, что он, обнаружив «левую скобку», наращивает счетчик на две единица, а при обнаружении «левой скобки» не изменяет его совсем. Так, при обнаружении «левой скобки» он вызывается дважды – один раз на петле, помеченной x_2/y_2, а второй раз при входе в состояние. А при обнаружении «правой скобки» счетчик сначала на петле уменьшается, а затем при входе в состояние увеличивается.

Причина такой работы управления МТ, в неверной трактовке авторами функционирования автомата типа Мура. Видимо, они полагают, что сигнал при состоянии у автомата Мура исполняется только при входе в это состояние (см. переход из состояния «0» в «1»), а на самом деле он выдается всякий раз при входе в это состояние. В том числе и при переходе по петле. Таким образом, мы имеет дело не с

ошибкой (кто не ошибался?), а с более серьезной проблемой – неверной трактовкой в рамках SWITH-технологии функционирования автоматов типа Мура. Тестирование эквивалентной модели это и показало.

Часть 1. Выполнение арифметических операций на машине Тьюринга

Машина Тьюринга (МТ) – это универсальная учебная машина, созданная для уточнения понятия «алгоритм». Первым идею универсальной машины предложил Алан Тьюринг в 1936 г. Для уточнения понятия алгоритма им был разработан абстрактный универсальный исполнитель, представляющий собой логическую вычислительную конструкцию, но не реальную вычислительную машину. Впоследствии придуманная Тьюрингом вычислительная конструкция была названа машиной Тьюринга. Машина Тьюринга (рис. 13) состоит из следующих элементов:

– бесконечная в обе стороны лента, разделенная на ячейки;

– каретка, содержащая читающую и записывающую головки (Г);

– программируемый автомат – управляющее устройство (УУ).

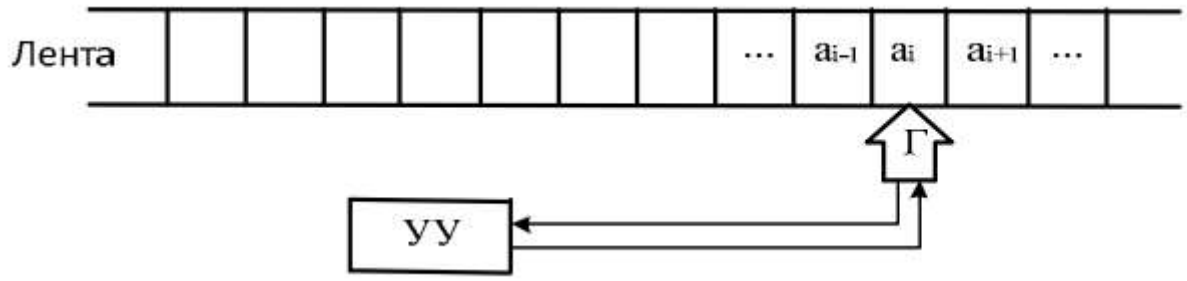

Рис.13 Структура машины Тьюринг

Управляющее устройство МТ может перемещаться влево и вправо по ленте, читать и записывать в ячейки ленты символы некоторого конечного алфавита. В процессе своей работы УУ управляется программой, во время каждого шага которой выполняются последовательно следующие действия: – записывать символ внешнего алфавита в ячейку (в том числе и пустой), заменяя находившийся в ней символ (в том числе и пустой); – передвигаться на одну ячейку влево или вправо; – менять свое внутреннее состояние. Поэтому при составлении программы для каждой пары «символ, состояние» нужно определить три параметра: символ a_i из выбранного алфавита А, направление перемещения каретки («←» – влево, «→» – вправо, «точка» – нет перемещения) и новое состояние автомата q_k. Например, команда $1 \leftarrow q_2$ обозначает «заменить символ на 1, переместить каретку влево на одну ячейку и перейти в состояние

q_2». Способы представления машины Тьюринга. Существует три способа представления машины Тьюринга: совокупностью команд, в виде графа, в виде таблицы соответствия.

Порядок выполнения работы.

Выполнить задание, согласно варианту.

1 На ленте МТ содержится последовательность символов «+». Напишите программу, которая каждый второй символ «+» заменит на «–». Замена начинать с правого конца последовательности. В состоянии q1 обозревается один из символов заданной последовательности

2 Дано число n в восьмеричной системе счисления. Разработать МТ, которая увеличивала бы заданное число n на 1. В состоянии q1 обозревается некая цифра входного слова

3 Дано десятичное натуральное число n. Разработать МТ, которая уменьшала бы это число на 1. В состоянии q1 обозревается правая цифра числа

4 Дано натуральное число n. Разработать МТ, которая уменьшала бы заданное число на 1, при этом в выходном слове старшая цифра не должна быть 0. Например, если входное слово 100, то выходным должно быть 99, но не 099. В состоянии q1 обозревается правая цифра числа

5 Дана последовательность открывающих и закрывающих скобок. Построить МТ, которая удаляла бы пары взаимных скобок, т. е. расположенных подряд ( ). Например, дано) (( ) (( ), надо получить)) . . . (. В состоянии q1 обозревается крайний левый символ строки

6 Дана строка из букв a и b. Разработать МТ, которая переместит все буквы a в левую, а буквы b – в правую часть строки. В состоянии q1 обозревается крайний левый символ строки

7 На ленте МТ записано десятичное число. Умножить это число на 2. В состоянии q1 обозревается крайняя левая цифра числа

8 Даны два натуральных числа m и n в унарной системе счисления, разделенные пустой ячейкой. В состоянии q1 обозревается самый правый символ входной последовательности. Разработать МТ, которая на ленте оставит сумму чисел m и n

9 Даны два натуральных числа m и n в унарной системе счисления, разделенные пустой ячейкой. В состоянии q1 обозревает самый правый символ входной последовательности. Разработать МТ, которая на ленте оставит разность чисел m и n.

Известно, что m > n

10 На ленте МТ находится десятичное число. Определить, делится ли это число на 5 без остатка. Если делится, то записать справа от числа слово «да», иначе – «нет». В начальном состоянии обозревается некоторая цифра входного числа

11 Дано число n в четверичной системе счисления. Разработать МТ, которая увеличивала бы заданное число n на 1. В состоянии q1 обозревается некая цифра входного слова

12 Дано десятичное натуральное число n. Разработать МТ, которая уменьшала бы это число на 1. В состоянии q1 обозревается некая цифра числа

Часть 2. Программирование машины Тьюринга

Машина Тьюринга представляет собой простейшую вычислительную машину с линейной памятью, которая согласно формальным правилам преобразует входные данные с помощью последовательности элементарных действий. Элементарность действия заключается в том, что оно меняет лишь небольшой кусочек данных – лишь одну ячейку, а число возможных действий конечно.

Несмотря на простоту МТ, на ней можно вычислить все, что можно вычислить на любой другой машине, осуществляющей вычисления с помощью последовательности элементарных действий. Это свойство МТ называется полнотой. На машине Тьюринга с помощью задания правил перехода можно имитировать всех других исполнителей, реализующих процесс пошагового вычисления, в котором каждый шаг вычисления достаточно элементарен. Существуют программы для обычных компьютеров, имитирующие работу машины Тьюринга. Но следует отметить, что данная имитация неполная, т. к. в машине Тьюринга присутствует абстрактная бесконечная лента.

Бесконечную ленту с данными невозможно в полной мере имитировать на компьютере с конечной оперативной памятью: памятью жёстких дисков и различных внешних носителей, регистров и кэш-процессора и др., которая может быть очень большой, но, тем не менее, всегда конечной. Одним из вариантов программы, имитирующей работу МТ, является детерминированная машина Тьюринга, интерфейс которой представлен на рис. 14. Интерфейс программы содержит следующие элементы:

«Лента»; текстовый блок «Состояние» с текстовым полем и кнопкой «Установить»; блок «Множество состояний»; текстовое поле «Конфигурация» с кноп□кой «Установить»; блок «Алфавит»; блок «Команды» и блок «Краткое руководство». Элемент «Лента» содержит движок, позволяющий перемещать содержимое ленты вправо-влево. Блок «Множество состояний» содержит 10 флажков для выбора состояний МТ и текстовые поля для ввода значений состояний, а также кнопку «Еще» для добавления текстовых полей состояний. Блок «Алфавит» содержит флажки «Цифры», «Буквы», «Символы»; 23 10 флажков для выбора цифр; 26 флажков для выбора латинских букв; кнопки «B», «», «=», «+», «-», «*», «/», «^», «%», а также 14 текстовых полей для ввода других символов.

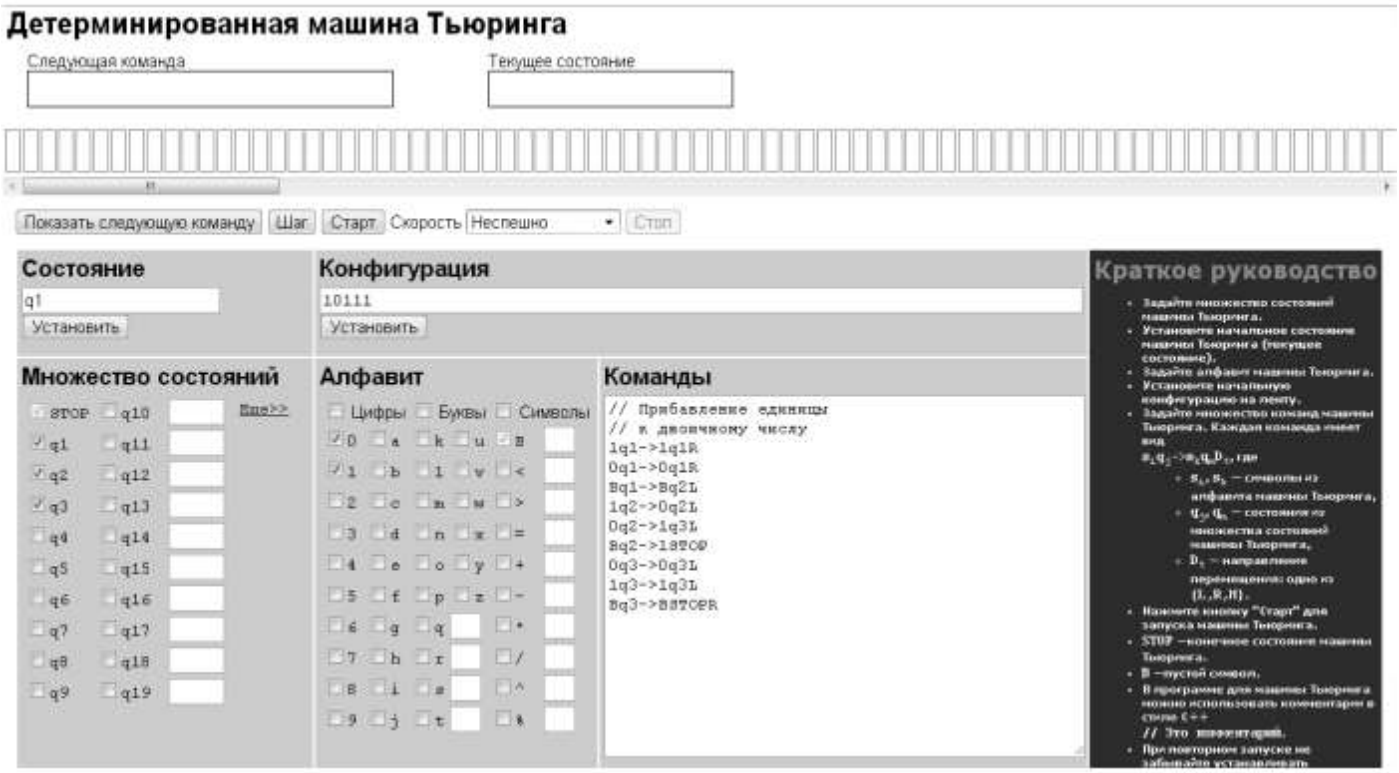

Рис. 14 Детерминированная машина Тьюринга

Блок «Команды» содержит окно кода для ввода команд программы МТ. Блок «Краткое руководство» содержит краткую инструкцию по работе с детерминированной машиной Тьюринга. Кнопка «Показать следующую команду» используется для просмотра следующей после только что исполненной команды. Кнопка «Шаг» предназначена для покомандного выполнения программы, а «Старт» – для запуска программы на выполнение в автоматическом режиме.

Поле со списком «скорость» позволяет выбрать скорость выполнения программы: мгновенно, очень быстро, быстро, неспешно, медленно или очень медленно. Кнопка

«Стоп» служит для останова программы. Перед запуском программы необходимо отметить необходимо для работы программы начальное состояние, выбрать символы алфавита, используемые в программе и ввести в окно кода заранее составленную программу.

После этого в блоке «Состояние» интерфейса МТ нажать кнопку «Установить», чтобы установить ее начальное состояние, а в блоке «Конфигурация» нажать кнопку «Установить», чтобы установить начальную конфигурацию (начальное слово) на ленту. После успешного выполнения перечисленных действий, используя кнопку «Шаг», можно проанализировать исполнение команд программы и выполнить ее отладку.

Порядок выполнения работы

Разработать программу для МТ, реализующую задание (таблица)

Вариант	Представить МТ таблицей соответствия для выполнения арифметической
1	111111+1 в унарной системе счисления
2	101110+1 в бинарной системе счисления
3	11111+11 в унарной системе счисления
4	10111011+11 в бинарной системе счисления
5	1001011+10 в бинарной системе счисления
6	10111011+11 в бинарной системе счисления
7	1111011+11 в унарной системе счисления
8	10100110+1 в бинарной системе счисления
9	1111+11 в унарной системе счисления
10	101011011+1 в бинарной системе счисления
11	1111011+11 в бинарной системе счисления
12	111001011+11 в бинарной системе счисления