Анисимов Андрей Евгеньевич
Старший преподаватель Удмуртского государственного университета, один из соавторов книги «Сборник задач по основаниям программирования», которая является практическим приложением к фундаментальному труду Н.Н. Непейводы и И.Н. Скопина «Основания программирования», а также к книге Н.Н. Непейводы «Стили и методы программирования».

Автоматное программирование. Часть 1

В жизни современного человека встречается масса самых разнообразных устройств – банкоматы, терминалы оплаты, мобильные телефоны, автоматы по продаже напитков, турникеты в метро и так далее. Все они работают по определённым алгоритмам, у каждого – свой. Однако в бескрайнем море автоматов существуют общие закономерности и правила их работы.

Часто приходится программировать нечто, напоминающее автоматический механизм. Программа при этом должна соответствовать природе этого «механизма». Такой подход ещё называют автоматным стилем программированием. Элементы этого стиля проявляются очень часто и в практическом программировании, и в жизни – при настройке и использовании разных устройств.

В статье даётся краткий обзор основной концепции автоматного программирования – понятия конечного автомата, из чего он состоит, как «работает» и как его запрограммировать. Тема эта важна и должна быть интересна для всех, кто увлекается программированием.

1. Автоматное программирование как один из стилей программирования

В теории и технологии создания программ выделены несколько так называемых стилей программирования – базовых, фундаментальных подходов к созданию программ. Каждый стиль обладает своими логическими и алгоритмическими концепциями, соглашениями. Кроме того, каждый стиль рассчитан на свою модель вычислителя – абстрактную конструкцию, способную выполнять действия и вычисления. В книге Н.Н. Непейводы \cite{SMP} «Стили и методы программирования» выделены несколько стилей – автоматное программирование (от состояний), струк-

турное, сентенциальное, функциональное, объектно-ориентированное и другие. Одним из них является стиль автоматного программирования (программирования от состояний). Главным теоретическим понятием этого подхода является конечный автомат – формальная конструкция, моделирующая процесс реальных вычислений. Это одно из важнейших понятий теории алгоритмов и теории формальных языков.

Целью этой статьи является познакомить читателя с понятием конечного автомата. В статье дан ряд примеров конечных автоматов с разбором их структуры и принципов работы. Примеры программ в этой статье приведены на языке программирования C/C++, однако это никак не ограничивает применение автоматов. Эти же задачи теми же приёмами можно с лёгкостью (порой – необыкновенной) решить на любом другом традиционном языке. Проделать такую работу на любимом языке читателя можно считать пожеланием автора в качестве занимательного упражнения.

2. Что такое конечный автомат

Рассмотрим, из чего состоит и как работает простой конечный автомат[1]. Удобно представить конечный автомат в виде считывающей головки (как головка магнитофона), способной распознавать со входной ленты записанные на неё символы (см. рисунок 1). Автомат не может изменить содержимое ленты (то есть его головка работает только на «воспроизведение»).

Кроме этого, автомат всегда находится ровно в одном из своих *состояний*. У каждого автомата конечное число различных состояний. Автомат может перейти из одного

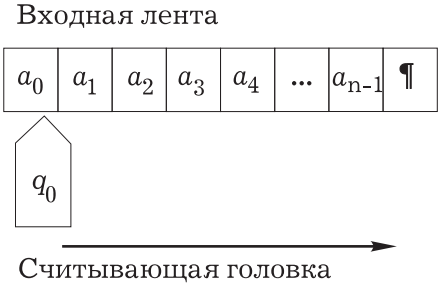


Рис. 1. Конечный автомат

[1] В данной статье ограничимся рассмотрением только одного частного случая автоматов – *детерминированных* конечных автоматов. Поэтому далее в тексте статьи термин «конечный автомат», строго говоря, надо понимать как «детерминированный конечный автомат».

состояния в другое, причём делает это мгновенно. Одно из состояний называется «начальным», в этом состоянии автомат находится в начале распознавания ленты. Также среди состояний выделены «заключительные», перейдя в которые процесс распознавания останавливается.

Алфавит символов (его ещё называют входным алфавитом), которые записаны на входной ленте, состоит из конечного числа знаков. Количество различных состояний, в которых может находиться автомат, тоже конечно. На входной ленте записана конечная последовательность символов алфавита. По этим причинам автомат называется *конечным*.

В каждом состоянии автомат «видит» на ленте определённый символ. И, в зависимости от текущего состояния и обозреваемого символа, автомат должен переключиться в другое состояние. В какое именно состояние переходит автомат, определяется правилами, которые называются *функцией перехода*. После перехода в новое состояние автомат сдвигается по входной ленте на одну позицию *вправо*, т.е. переходит к следующему символу.

Таким образом, автомат, для которого определены входной алфавит, множество состояний, начальное состояние, заключительные состояния и функция перехода, начинает считывать последовательно символы из входной ленты, переходя из одного состояния в другое. После каждого такого перехода считывающая головка перемещается на один символ вправо. Останавливается процесс тогда, когда автомат оказывается в одном из *заключительных состояний*, либо произойдёт ошибка.

Пример 1. «Сконструируем» конечный автомат A_1. Пусть у этого автомата будут три состояния, обозначим их q_0, q_1 и q_2. Начальным состоянием будем считать q_0, заключительным – q_2. Входной алфавит будет состоять из трёх символов – $\{a, b, c\}$. Правила перехода такие:

а) если автомат находится в состоянии q_1 и обозревает символ a, то переход в состояние q_1;

б) если автомат находится в состоянии q_1 и обозревает символ b, то переход в состояние q_1;

в) если автомат находится в состоянии q_1 и обозревает символ c, то переход в состояние q_2.

Представим те же правила перехода в виде *таблицы переходов автомата* A_1 (см. таблицу 1):

Таблица 1

Состояние	Условие	Переход
q_0	a	q_1
q_1	b	q_1
	c	q_2
q_2		

Первая колонка «Состояние» содержит текущее состояние автомата. Во второй колонке «Условие» даны символы, обозреваемые в данный момент автоматом. Третья колонка «Переход» указывает, в какое новое состояние переходит автомат, находящийся в данном состоянии и обозревающий данный символ.

Ещё более наглядно построенный автомат A_1 можно представить в виде *графа состояний* (см. рис. 2). На графе все состояния представлены кружочками, причём договоримся, что все заключительные состояния будем обозначать кружочками с жирной границей (здесь это состояние q_2). Стрелки обозначают переходы из одного состояния в другое, каждая стрелка по-

мечается символом входного алфавита. Для простоты здесь и в последующем будем считать, что начальное состояние автомата – это q_0.

Заметим, что переход из состояния q_1 по символу b происходит в то же самое состояние, поэтому соответствующая стрелка образует петлю.

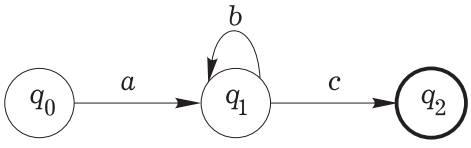


Рис. 2. Граф состояний автомата A_1

Таким образом, граф состояний и таблица переходов – это два *эквивалентных* способа описания поведения конечного автомата при распознавании входной цепочки. Иногда таблицу перехода называют «табличным представлением графа состояний».

Автомат построен. Проверим, как он «работает», т.е. распознаёт входные строки. Для этого надо подать на входной ленте автомата некоторую цепочку символов. Попробуем испытать автомат несколько раз.

а) Входная цепочка «*ac*». Автомат находится в начальном состоянии q_0 и «видит» на ленте первый символ *a*. По правилам перехода автомат пере ключается в состояние q_1 и перемещается к следующему символу.

Теперь в состоянии q_1 автомат читает символ *c*, значит, происходит переход в состояние q_2. Так как q_2 – заключительное состояние, на этом работа автомата закончена, и входная цепочка символов «*ac*» считается разобранной или, говорят, *допущенной* автоматом. Последовательность состояний, в которых побывал автомат при разборе этой строки символов, получилась такой: q_0, q_1, q_2.

б) Теперь пусть входная цепочка такая – «*abbbc*» (см. рис. 3). Аналогично, начинаем с начального состояния q_0, переходя от символа к символу и переключаясь с одного состояния на другое. Получим такую последовательность состояний: q_0, q_1, q_1, q_1, q_1, q_2. И опять мы добрались до заключительного q_2, а значит, и эта цепочка допущена. Обратим внимание, что на ленте (рис. 3) находится ещё один специальный символ «¶». Это так называемый *маркер конца строки*, который будет являться признаком того, что символы строки закончились [1].

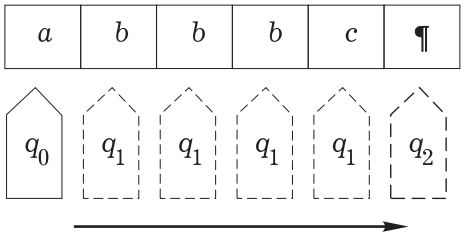


Рис. 3. Разбор автоматом A_1 строки «*abbbc*»

в) Подадим на вход такую строку: «*abb*». Ясно, что при разборе этой строки автомат побывает в таких состояниях: q_0, q_1, q_1, q_1. Но цепочка кончилась, а автомат остался в состоянии q_1, которое не является

[1] Кстати, работу автомата легче всего проследить по графу состояний, «передвигаясь» последовательно от начального состояния по стрелкам, помеченным соответствующими символами входной строки. Если мы доходим до кружка с жирной границей, то цепочка успешно допускается, если нет – то нет. Это очень похоже на детскую игру, в которой надо бросать кости и перемещать фишки по полю от клетки к клетке по стрелкам. Ещё это похоже на знаменитую игру «Монополия».

заключительным! Такая ситуация является *ошибкой*, и цепочка считается *не допущенной автоматом*.

г) Пусть теперь на входе строка «*aac*». Автомат вначале находится в q_0, «видит» *a*. Переключается в q_1 и переходит к следующему символу (вторая *a*). Но для состояния q_1 и символа *a* в наших правилах перехода не указано, в какое состояние нужно дальше перейти! Это снова ошибка, строка «*aac*» не допущена.

Эти примеры работы автомата показывают, что *не любые* последовательности символов алфавита (входные строки) допускаются автоматом. В этом примере конечный автомат A_1 допускает только такие цепочки строк, каждая из которых начинается символом *a*, заканчивается символом *c*, между которыми может находится нуль или более символов *b*. Любая другая цепочка не будет допущена A_1.

Формально это множество строк можно записать так: $\{ab^n c \mid n \geq 0\}$. Тут всё ясно без комментариев.

Конец примера 1.

Пример 2. Определим конечный автомат A_2. Множество состояний – $\{q_0, q_1, q_2, q_3\}$, из них q_0 – начальное, q_3 – заключительное. Алфавит такой же, как и выше – $\{a, b, c\}$. Переходы определяются таблицей переходов (см. таблицу 2) или графом состояний (см. рис. 4).

Таблица 2

Состояние	Условие	Переход
q_0	*a*	q_1
q_1	*b*	q_2
q_2	*b*	q_2
	c	q_3
q_3		

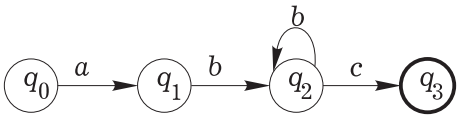


Рис. 4. Граф состояний автомата A_2

Ясно, что автомат A_2 допустит такие строки, как «*abc*», «*abbbc*», «*abbbbbbbbbbbbbbc*», но не допустит «*ac*», «*aac*», «*ab*». Убедитесь в этом самостоятельно.

Таким образом, множество строк, допускаемых автоматом A_2, есть $\{ab^n c \mid n \geq 1\}$.

Конец примера 2.

3. Более формальное определение конечного автомата

Разобравшись в предыдущей главе с понятием конечного автомата содержательно, приведём здесь его более формальное определение.

Определение 1. *Детерминированный конечный автомат* состоит из следующих элементов:

1) множество состояний

$$Q = \{q_0, q_1, \ldots, q_n\},$$

в котором выделяются:

а) одно начальное состояние q_0,

б) одно или несколько конечных состояний;

2) входной алфавит – множество входных символов, которые автомат способен прочитать на входной ленте; будем обозначать алфавит буквой Σ;

3) правила (функция) перехода; каждое правило зависит от двух аргументов – текущего состояния и текущего символа – и определяет, в какое состояние должен перейти автомат при этом условии.

Конец определения 1.

Так как все множества, входящие в определение автомата, являются конечными, поэтому он называется конечным. В дальнейшем мы будем рассматривать только детерминированные[1] конечные автоматы, поэтому слово «детерминированный», как правило, будем опускать.

Каждое правило, сопоставляющее состоянию q_i и текущему входному символу a новое состояние q_{i+1}, удобно представлять в виде стрелки на графе состояний (рис. 5) или отдельной строки в таблице переходов (таблица 3).

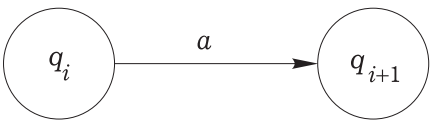


Рис. 5. Стрелка означает переход от состояния q_i по символу a к состоянию q_{i+1}

Таблица 3

Состояние	Условие	Переход
	...	
q_i	a	q_{i+1}
	...	

Также заметим, что не для всех пар (q_k, a) может существовать правило перехода в новое состояние.

Определение 2. Цепочка входных символов (входная строка, входной поток) $a_0a_1 \ldots a_{n-1}$ называется *допустимой конечным автоматом* тогда и только тогда, когда существует последовательность состояний

$$q_0, q_1, \ldots, q_n,$$

где q_0 – начальное состояние, а каждое последующее состояние q_{i+1} получено по правилу перехода от предыдущего состояния q_i и очередного символа входной цепочки a_i, при этом последнее состояние в данной последовательности q_n – заключительное.

Конец определения 2.

Пример 3. Рассмотрим, почему строка «*abbbc*» из примера 1 (случай *б*, рисунок 3) допускается конечным автоматом A_1. Начальное состояние автомата q_0. Первый входной символ цепочки – *a*. В таблице переходов и на графе состояний есть правило перехода, по которому из состояния q_0 при символе *a* конечный автомат должен переключаться в состояние q_1, то есть $(q_0, a) \to q_1$.

Следующим входным символом становится *b* (головка автомата сдвинулась вправо). Далее повторяется тот же алгоритм:

$$(q_1, b) \to q_1;\ (q_1, b) \to q_1;$$
$$(q_1, b) \to q_1;\ (q_1, c) \to q_2.$$

Автомат останавливается, так как теперь он находится в заключительном состоянии q_2. Строка «*abbbc*» допущена.

Конец примера 3.

Определение 3. Множество всех цепочек, допускаемых конечным автоматом *A*, называется *регулярным множеством* конечного автомата *A*.

Конец определения 3.

Регулярное множество конечного автомата может быть определено спомощью *регулярного выраже-*

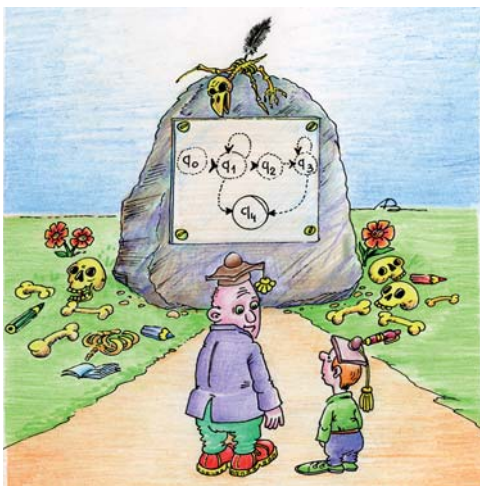
[1] Наверное, читатель уже понял, что кроме *детерминированных*, есть ещё недетерминированные конечные автоматы. Обсуждение разницы в этой статье проводить не будем – для этого можно посмотреть литературу. Лишь заметим, что детерминированный – это частный случай недетерминированного конечного автомата.

ния, как это было сделано в примерах 1 и 2.

Пример 4. Определим конечный автомат A_3. Множество состояний $Q=\{q_0,q_1,q_2,q_3,q_4\}$, начальное состояние q_0, заключительное состояние q_4, множество входных символов

$$\Sigma=\{'.', 0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}.$$

Граф состояний конечного автомата A_3 приведён на рисунке 6 , а его таблица переходов – в таблице 4.



Здесь использован символ *digit*, означающий *любую из десяти цифр* конечного алфавита. Это сделано лишь для сокращения записи правил перехода (а то пришлось бы писать правило для каждой цифры в отдельности). Кроме этого напоминаем, что символ ¶ – это маркер конца строки, который подразумевается после последнего символа входной строки.

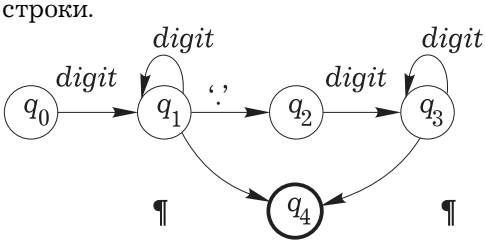


Рис. 6. Граф состояний автомата A_3

Таблица 4

Состояние	Условие	Переход
q_0	*digit*	q_1
q_1	*digit*	q_1
	'.'	q_2
	¶	q_4
q_2	*digit*	q_3
q_3	*digit*	q_3
	¶	q_4
q_4		

Ясно, что такие цепочки, как «123», «3.1415926», «0.00001», «0» будут допущены конечным автоматом A_3, а цепочки «1.», «.12», «1..10» – не будут.

Понятно также, что регулярное множество таких автоматов есть

$$\left\{digit^n\left[.digit^m\right]\middle|n\geq 1, m\geq 1\right\},$$

где квадратные скобки [ ] означают необязательность того, что внутри них находится.

Итак, конечный автомат A_3 распознаёт целые или дробные десятичные числа.

Конец примера 4.

Пример 5. Рассмотрим конечный автомат A_4. Его граф состояний дан на рисунке 7.

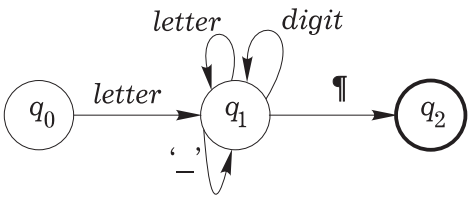


Рис. 7. Граф состояний автомата A_4

Здесь символ *letter* обозначает любую букву латинского алфавита $a-z$ или $A-Z$. Самостоятельно постройте табличное представление переходов этого автомата.

Автомат A_4 допустит такие цепочки, как «*abcd*», «*File_Save*», «*x*1*y*1», «*E_r_r_o_r_*404», но не допустит цепочки «1*_liter*», «123», «*_No_Sir*». При анализе становится ясно, что этот конечный автомат – не что иное, как распознаватель *идентификаторов*, как их принято записывать в языках программирования.

Конец примера 5.

Упражнения

Здесь даны несколько полезных упражнений для практического закрепления материала статьи. В упражнениях использованы следующие условные обозначения: *digit* – это любая из десятичных цифр $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$, *letter* – любая из букв латинского алфавита $\{a...z, A...Z\}$, *sym* – любая цифра *digit* или любая буква *letter*.

Упражнение 1. Постройте граф состояний по табличному заданию переходов конечного автомата:

а) A_{y1} (см. таб. 5), где q_0 – начальное, а q_3 – заключительное состояния;

б) A_{y2} (см. таб. 6), где q_0 – начальное, а q_6 – заключительное состояния;

в) A_{y3} (см. таб. 7), где q_0 – начальное, а q_5 – заключительное состояния.

Таблица 5

Состояние	Условие	Переход
q_0	a	q_1
	b	q_2
q_1	a	q_1
	b	q_2
q_2	b	q_2
	c	q_3
q_3		

Таблица 6

Состояние	Условие	Переход
q_0	*letter*	q_1
q_1	*sym*	q_1
	:	q_2
q_2	=	q_3
q_3	*digit*	q_4
	letter	q_5
q_4	*digit*	q_4
	¶	q_6
q_5	*sym*	q_5
	¶	q_6
q_6		

Упражнение 2. Постройте таблицу переходов по графу состояний конечного автомата:

а) A_{x1} (см. рис. 8);

б) A_{x2} (см. рис. 9).

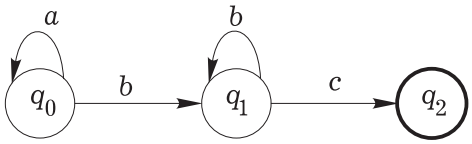


Рис. 8. Граф состояний автомата A_{x1}

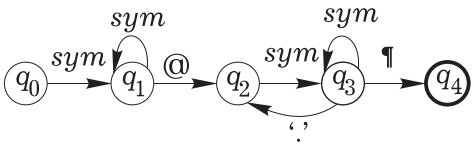


Рис. 9. Граф состояний автомата A_{x2}

Упражнение 3. Смоделируйте процесс работы конечного автомата[1]:

а) A_{y1} (см. таб. 5) на строках «*abc*», «*aabbbc*», «*bbc*», «*ac*».

б) A_{y2} (см. таб. 6) на строках «$x := 1$», «$Count := Count1$», «$Sum1 := x1y2$», «$\sin := 911$», «$1 := a$»;

[1] Внимание, не все строки в примерах допускаются автоматом!

в) A_{y3} (см. таб. 7) на строках «*a*[1]», «*abs*[10][15][20]», «*array*1[1][2]», «*s*[*i*][*j*]», «[123]»;

г) A_{x1} (см. рис. 8) на строках «*abc*», «*aabbbc*», «*bbc*», «*ac*»;

д) A_{x2} (см. рис. 9) на строках «*user*@*mail.ru*», «*vasya*1989@*x.y.z*», «*aae*@*uni.udm.ru*», «123@*list.ru*», «*www.microsoft.com*».

Упражнение 4. Определите неформально (словесно) и, если получится, формально (в виде регулярного выражения) множество строк, допускаемых каждым из конечных автоматов из упражнений 1 и 2 (подсказка – в упражнении 3).

Таблица 7

Состояние	Условие	Переход
q_0	*letter*	q_1
q_1	*sym*	q_1
	[	q_2
q_2	*digit*	q_3
q_3	*digit*	q_3
	]	q_4
q_4	[	q_2
	¶	q_5
q_5		

Упражнение 5. Определите конечный автомат в виде графа состояний и таблицы переходов для такого вида строк:

а) обыкновенные дроби (например, «1/2», «13/11», »200/100», «0/1234567»);

б) сумма, разность целых неотрицательных чисел (например, «1+988–2007+112»);

в) арифметический калькулятор (например, «3+125*21-12/5»);

г) даты (например «11.09.2001», «0.0.0», «29.02.1900»);

д) цена в рублях и копейках (например, «4р12к», «100р0к», «0р99к»).

В следующем номере «Потенциала» будет опубликовано продолжение данной статьи. Будет рассмотрено, как дополнить конечный автомат способностью производить вычисления и как писать программу для компьютера, реализующего такой автомат.

Список рекомендуемой литературы

1. Непейвода Н.Н., Скопин И.Н. Основания программирования. Москва-Ижевск: РХД, 2003. 880 с.

2. Непейвода Н.Н. Стили и методы программирования. – М.: ИНТУИТ. РУ. «Интернет-Университет Информационных Технологий», 2005. 320 с.

3. Сборник заданий по основаниям программирования: Учебное пособие/ Анисимов А.Е., Пупышев В.В. – М.: ИНТУИТ.РУ «Интернет-Университет Информационных технологий»; БИНОМ. Лаборатория знаний,2006. 348 с.

4. http://is.ifmo.ru: сайт по автоматному программированию и мотивации к творчеству // Кафедра «Технологии программирования» Санкт-Петербургского государственного университета информационных технологий, механики и оптики.