В. Н. КАСАТКИН

СЕМЬ ЗАДАЧ ПО КИБЕРНЕТИКЕ

Издательское объединение «Вища школа»
Головное издательство
Киев — 1975

6Ф0.1
К28

УДК 62—50 (023.11)

Семь задач по кибернетике. Касаткин В. Н. Издательское объединение «Вища школа», 1975, 96 с.

С появлением ЭЦВМ возможности человека при постановке и решении задач возросли чрезвычайно. Их появление способствовало и способствует возникновению новых, в «домашинный период» совершенно неизвестных методов и алгоритмов решения задач. Применение ЭЦВМ требует высокой алгоритмической культуры, воспитание которой следует начинать еще на школьной скамье.

В книге рассказывается о некоторых, специфических для кибернетики, подходах к решению задач. Эти подходы разъясняются на задачах занимательного характера. Изложение ведется в форме, доступной для учащихся. Другими методами рассматриваемые задачи решать затруднительно — применение вводимых понятий делает их решение очень простым и изящным.

Книжка рассчитана на учащихся физико-математических школ. Будет полезна всем учащимся общеобразовательных школ, которые интересуются математикой и кибернетикой.

Табл. 11. Ил. 50. Библиогр. 9.

Редакция литературы по математике и физике
Зав. редакцией *А. С. Макуха*

К $\frac{30500 - 176}{M211\,(04) - 75}$ 423—75

ПРЕДИСЛОВИЕ

Читателю-школьнику предлагается познакомиться с семью занимательными задачами по кибернетике и затем попробовать свои силы в самостоятельном решении нескольких аналогичных задач.

Задачи предназначены для тех школьников, которые увлекаются математикой и хотят разобраться в том, как и какими своими разделами сотрудничают между собой математика и кибернетика.

В числе предложенных — задачи по теории автоматов, элементам теории алгоритмов и основам программирования. Все предлагаемые задачи были неоднократно разобраны на занятиях в школе юных кибернетиков Малой Академии наук школьников Крыма «Искатель» и оказались удачным средством первоначального ознакомления учащихся с основами кибернетики.

Занимательность изложения не должна, в глазах читателя, затушевать серьезность рассматриваемых вопросов и избрана автором лишь для того, чтобы образностью и занимательностью привлечь внимание и облегчить работу тем, для кого вводимые понятия являются совершенно новыми.

Приведенные образцы решения задач, предложенные для самостоятельного решения, не являются единственными, читатель, возможно,

найдет и другие, к поискам которых и следует стремиться.

Автор не претендует на полноту и завершенность рассмотрения затронутых в книге вопросов. Книгу следует рассматривать всего лишь как своеобразное путешествие в «предгорья» Большой кибернетики.

Пожелания относительно содержания книги просьба присылать по адресу: 252054, Киев, 54, ул. Гоголевская, 7, Головное издательство издательского объединения «Вища школа», редакция литературы по математике и физике.

Автор

Рассказ-задача 1

КАК РАСКРЫВАЮТ ТАЙНЫ ЧЕРНЫХ ЯЩИКОВ

Представьте себя на месте получателя следующего письма, которое приводит английский физиолог и кибернетик Уильям Росс Эшби в своей книге «Введение в кибернетику»[1].

«Замогилье»
Дом с привидениями

Дорогой друг!

Некоторое время назад я купил старый дом, но обнаружил, что он посещается двумя призрачными звуками: Пением и Смехом. В результате он мало подходит для жилья. Однако я не отчаиваюсь, ибо я установил путем практической проверки, что их поведение подчиняется определенным законам, непонятным, но непререкаемым, и что я могу воздействовать на них, играя на Органе или сжигая Ладан.

В течение каждой минуты каждый из этих звуков либо звучит, либо молчит; никаких переходов они не обнаруживают. Поведение же их в последующую минуту зависит только от событий предыдущей минуты, и эта зависимость такова:

Пение в последующую минуту ведет себя так же, как и в предыдущую (звучит или молчит), если только в эту предыдущую минуту не было игры на Органе при молчащем Смехе. В последнем случае оно меняет свое поведение на противоположное (звучание на молчание и наоборот).

Что касается Смеха, то если в предыдущую минуту горел Ладан, Смех будет звучать или молчать в зависимости от того, звучало или молчало Пение (так что Смех копирует Пение минутой позже).

Если, однако, Ладан не горел, Смех будет делать противоположное тому, что делало Пение.

В ту минуту, когда я пишу Вам это, Смех и Пение оба звучат. Прошу Вас сообщить мне, какие действия с Ладаном и Органом должен я совершить, чтобы установить и поддерживать тишину в доме?

---

[1] Росс Эшби У. Введение в кибернетику. М., Изд-во иностр. лит., 1959, с. 92—93.

Эта необычная на первый взгляд задача, если отбросить занимательную форму ее постановки, предстанет перед нами рядовым примером тех проблем, которые часто решают в кибернетике. Речь идет о задаче исследования черного ящика. Что значит «черный ящик»?

В кибернетике существуют рабочие понятия «белый ящик» и «черный ящик». Вот как определяет эти понятия Норберт Винер:

«Под черным ящиком я подразумеваю какое-либо устройство (например, четырехполюсник с двумя входными и двумя выходными полюсами), которое выполняет определенную операцию над настоящим и прошлым входного потенциала, но для которого мы не обязательно располагаем информацией о структуре, обеспечивающей выполнение этой операции.

С другой стороны, белый ящик есть аналогичная цепь, в которой для обеспечения заданной зависимости между входом и выходом мы связали входной и выходной потенциалы согласно определенному структурному плану» [1].

При решении задач в кибернетике, как и в математике, прибегают к абстракции. И черный ящик часто выступает абстрактным образом конкретного устройства.

Попробуем взглянуть на дом с привидениями из Замогилья, как на черный ящик, использовав при этом определение Норберта Винера.

У дома в Замогилье есть два выходных полюса — по одному из них транслируется Смех, по другому — Пение. Есть у нашего черного ящика и два входных полюса — по ним наш корреспондент вводил свои воздействия: игру на Органе и сжигание Ладана. И мы ничего не знаем о том, как «устроен» наш черный ящик, что у него внутри.

---

[1] Винер Н. Кибернетика или управление и связь в животном и машине. М., «Советское радио», 1968, с. 33.

Решение задачи мы уже начали, заменив таинственный дом с привидениями черным ящиком — этаким «музыкальным автоматом». Продолжим.

Введем понятие о состоянии автомата (или черного ящика). Состояние любого автомата определяется в любой момент времени тем, что удается обнаружить на его выходах. В данном случае может быть только четыре состояния. Вот они:

a_1 — первое состояние — нет Смеха и нет Пения,
a_2 — второе состояние — есть Смех и нет Пения,
a_3 — третье состояние — нет Смеха и есть Пение,
a_4 — четвертое состояние — есть Смех и есть Пение.

Воздействия, которые может организовать корреспондент из Замогилья, обозначим буквой b с индексами:

b_1 — нет игры на Органе и нет сжигания Ладана,
b_2 — нет игры на Органе и есть сжигание Ладана,
b_3 — есть игра на Органе и нет сжигания Ладана,
b_4 — есть игра на Органе и есть сжигание Ладана.

Используем еще один прием формализации, широко применяемый в кибернетике при изучении самых разнообразных черных ящиков, а именно: займемся вычерчиванием так называемых **граф-схем** черного ящика.

При вычерчивании граф-схемы мы будем использовать только что введенные понятия: состояние автомата и воздействие на автомат.

Состояние автомата на граф-схеме будем изображать кружочком, внутри которого будут вписываться обозначения состояния, например на рис. 1 дано изображение первого состояния на граф-схеме.

Из текста письма нам известно, что под влиянием сжигания Ладана в сочетании с игрой на Органе наш черный ящик изменяет свое поведение — переходит из одного состояния в другое. Переход из одного состояния в другое на схеме будем изображать стрелкой, соединяющей два состояния, а около стрелки будем выписывать обозначе-

ние того воздействия, под влиянием которого осуществляется данная смена состояний.

На рис. 2 изображена ситуация, когда под влиянием воздействия b_1 черный ящик сменил состояние a_1 на a_2. На рис. 3 отражена ситуация, состоящая в том, что воздействие b_2 не меняет состояния a_2 — воздействие b_2 есть, а смены состояния нет. Введением этих обозначений завершаем подготовку формализации условий задачи. Те-

Рис. 1

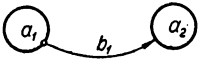

Рис. 2

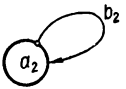

Рис. 3

перь нам предстоит вернуться к тексту письма и попробовать вычертить граф-схему.

Эта часть аналогична тому моменту решения алгебраических задач в школе, когда мы, введя буквенные обозначения всех используемых в задаче величин, приступаем, например, к составлению уравнения или системы уравнений. В результате все величины (искомые и данные) окажутся связанными и представленными в удобной форме — в виде одной или нескольких формул.

В данном случае разыскиваемая нами граф-схема должна стать формой выражения сущности процесса смены одного состояния на другое под влиянием внешних воздействий. Мы намерены заняться графическим решением задачи об исследовании черного ящика. Поскольку дело это необычное, то решение мы будем вести весьма подробно.

Графическое решение начинаем с рассмотрения состояния a_1 (напомним, что в этом состоянии нет ни Смеха, ни Пения). Изобразим это состояние не кружочком, а прямоугольником (делается это исключительно ради удобства изложения):

a_1	нет Пения нет Смеха

Читаем текст письма (4-й абзац); там раскрывается влияние Ладана на Смех: если Ладан в данную минуту горит, то Смех в последующую минуту будет делать то, что в данную минуту делало Пение.

Среди наших четырех воздействий только два включают сжигание Ладана. Эти соображения позволяют заполнить следующую часть разыскиваемой граф-схемы (рис. 4).

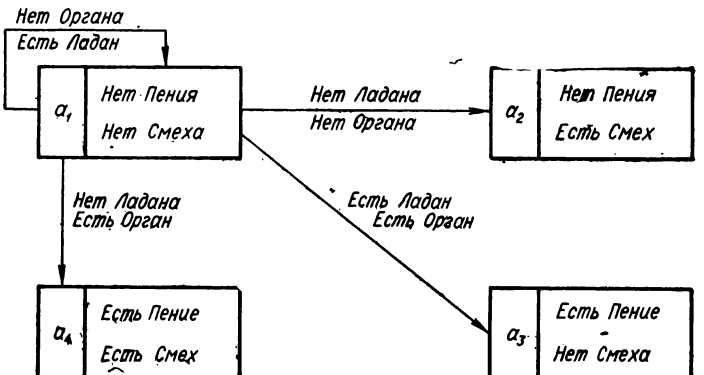

Рис. 4

Рассмотрите внимательно схему и вы убедитесь, что при отсутствии Ладана в составе нашего воздействия Смех в последующую минуту обязательно есть, а именно это и требовалось по условиям четвертого абзаца письма. Наличие Ладана в составе воздействия приводит к возникновению состояний, в которых нет Смеха.

Заметим, что использование одного только цитированного условия из письма приводит к неоднозначному решению. Мы на всех возможных случаях останавливаться не будем.

Обращаемся к тексту третьего абзаца письма, в котором явно указана зависимость Пения от воздействия игрой на Органе при молчащем Смехе.

Дополним ранее вычерченную часть граф-схемы новыми (двойными) стрелками (рис. 5).

Рассмотрение граф-схемы начнем с того, что сначала убедимся в том, что ранее вычерченная часть согласуется с условием третьего абзаца. Действительно, в состоянии a_1 нет Смеха; взяв воздействие b_2 (нет Органа, есть Ладан),

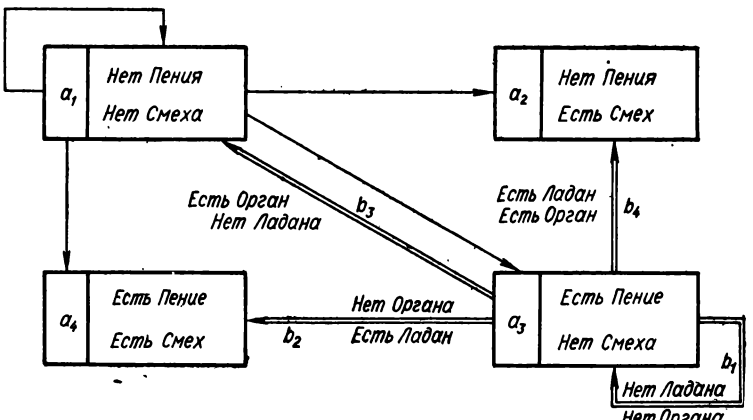

Рис. 5

мы замечаем, что наш автомат остается в том же состоянии, что означает отсутствие Пения в последующую минуту. Если же мы рассмотрим воздействие b_1 (нет ни Ладана, ни Органа), то увидим, что автомат перейдет в состояние a_2, в котором опять-таки нет Пения.

Сопоставление граф-схемы (части, вычерченной ранее) с условием третьего абзаца показало, что граф-схема отражает действительное поведение автомата. Продолжим разбор, обратившись к состоянию a_3, так как в этом состоянии нет Смеха, и используем условия третьего абзаца. Ясно, что если на автомат воздействовать без игры на Органе, то он должен перейти лишь в те состояния, в ко-

торых Пение есть. Если же в состав воздействия ввести игру на Органе, то автомат должен переходить только в те состояния, в которых нет Пения.

Обратившись к другим состояниям и используя только тексты двух тех же самых абзацев письма, легко дополнить граф-схему до окончательного вида. Полная граф-схема (рис. 6) показывает, как изменяется любое из состояний под влиянием любого из воздействий.

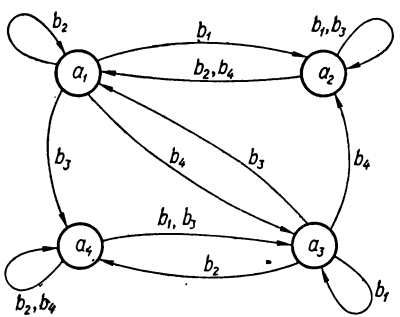

Рис. 6

Располагая полной граф-схемой, мы можем обратиться к проблеме, стоящей в письме, и решить ее. В письме спрашивается, как сменить состояние a_4 (Смех и Пение оба звучат) на состояние a_1 (Смех и Пение оба молчат).

На граф-схеме легко увидеть путь, который ведет из состояния a_4 в состояние a_1. Этот путь представляет из себя последовательность воздействий:

1-я минута — воздействие b_1 (прекратить игру на Органе и сжигание Ладана);

2-я минута — воздействие b_3 (не зажигая Ладана, играть минуту на Органе);

3-я минута — воздействие b_2 (прекратить игру на Органе и зажечь Ладан).

Если Ладан затем все время будет гореть, то в доме установится требуемая тишина. Более того, располагая граф-схемой, мы можем теперь осуществить такие воздействия, при которых в доме, например, будет непрерывно звучать Смех или непрерывно звучать (если найдутся желающие послушать) Пение. Подобрать системы воздействия для обеспечения обеих ситуаций, пользуясь

граф-схемой, мы предоставляем читателю (в ответах, приведенных в конце книги, обе системы рассмотрены).

Итак, мы ознакомились на примере с методом исследования черного ящика. Ничего не зная о его структуре и не интересуясь этим, мы сумели получить полное представление о его деятельности. Решение задачи мы получили в форме граф-схемы, которая оказалась очень удобной для описания деятельности дома с привидениями. Граф-схема, полученная нами, позволила не только решить поставленную задачу, но и явилась инструментом для решения других задач, связанных с появлением Смеха и Пения, и управления их поведением.

Замечательным является то обстоятельство, что решение в форме граф-схемы весьма убедительно. В частности, нам ясно, что теперь о поведении дома с привидениями мы знаем все, что на граф-схеме отражены все состояния и указаны все возможные воздействия.

Использование граф-схемы при решении задач мы покажем еще на одном занимательном примере (рассказ-задача 2). Речь идет об использовании граф-схемы как способе задания стратегии в некоторой игре.

Задачи для самостоятельного решения

1. Укажите последовательность воздействий, которые следует осуществить, чтобы в доме все время звучало Пение.
2. Укажите последовательность воздействий, которые следует осуществить, чтобы в доме все время звучал Смех.

Исходное состояние Дома в обеих задачах: Смех и Пение молчат (состояние — a_1).

Рассказ-задача 2

АВТОМАТ СЛАВЫ СТРЕЛЬЦОВА

Представьте себя участником игры с предметами, например с камешками. Условия игры следующие. Соперников двое. Ходят по очереди. За каждый свой ход каж-

дый из играющих может взять себе из общей группы в 25 предметов 1, 2, 3 или 4 предмета. Победителем считается тот, кто сумеет из общей группы предметов взять в свою столько, что по окончании игры их число у него будет четным.

Попробуем разработать беспроигрышную стратегию, пользуясь которой, всегда можно победить. Очевидно, необходимо иметь ответы на два вопроса:

1. Каким по-очереди вступать в игру? Первым или вторым? Или, иначе говоря, что лучше — начинать игру или ждать первого хода соперника?

2. Сколько предметов следует брать при своем ходе и чем при этом руководствоваться?

Занимательным в этой истории является то, что один из вариантов безпроигрышной стратегии был открыт пятнадцатилетним школьником, и то, что эта стратегия была стратегией черного ящика. Иначе говоря, Слава Стрельцов (так звали этого школьника) не только догадался, как безошибочно следует играть, а сумел составить граф-схему автомата (черного ящика), который может выступить в игре вместо него.

Заметим, что открыть секрет успеха в этой игре — отнюдь не простое дело. Лучше всего убедиться в этом, если попробовать сделать это самим. Вы убедитесь как сложно найти простые рекомендации к любому ходу. Один из вариантов стратегии приводит Б. А. Кордемский в своей книге «Математическая смекалка» (текст его алгоритма безошибочной игры мы приводим в разделе «Ответы и решения»). Кстати, очень интересно сравнить текст Б. А. Кордемского и граф-схему Славы Стрельцова. О том, как пришел Слава к своему результату, мы рассказать не можем. (Он говорит: «Думал, думал и придумал».) и поэтому только прокомментируем его решение.

Начнем с того, что покажем граф-схему его мудрого беспроигрышного автомата (рис. 7). Объясним условные обозначения:

4(2) — эта запись говорит: если партнер взял себе 4 предмета — бери при своем ходе 2.

Часть схемы, изображенную на рис. 8, следует понимать так: находясь в состоянии a_2, автомат обнаружил, что соперник-человек взял себе 2 предмета, и принимает свое решение — решение автомата: «Беру 3 предмета и перехожу из состояния a_2 в состояние a_3».

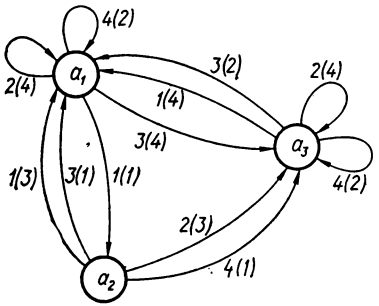

Рис. 7

Заметим, что a_1 — это состояние, находясь в котором автомат вступает в игру. В этом состоянии автомат в общей группе видит 25 предметов и ждет первого хода человека. В ходе игры автомат, переходя из состояния в состояние, может не раз переходить в состояние a_1.

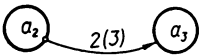

Рис. 8

Чтобы понять красоту, изящество работы автомата, мы просто сыграем один раз с этим автоматом.

Итак, «Мы» — человек.

1-й ход человека: «Беру себе из общей группы 3 предмета».

Ответный ход автомата. Автомат рассуждает так: «Я, автомат, нахожусь в состоянии a_1 и поэтому на ход человека «Беру 3 предмета» должен взять 4 предмета и перейти в состояние a_3.

После первого хода имеем:

Ход	Человек	Автомат
1-й	3	4

2-й ход человека: «Беру себе 4 предмета».

Ответный ход автомата (смотри граф-схему): «Так как я нахожусь в состоянии a_3, то в ответ на ход человека «Беру 4» сам себе возьму 2 и останусь в состоянии a_3».

После второго хода имеем:

Ход	Человек	Автомат
1-й	3	4
2-й	4	2

3-й ход человека: «Беру себе 1 предмет».

Ответный ход автомата: «Беру себе 4 предмета и перехожу в состояние a_1».

Ситуация после третьего хода:

Ход	Человек	Автомат
1-й	3	4
2-й	4	2
3-й	1	4

Четвертый ход человека: «Беру себе 4 предмета».

Ответный ход автомата: «Я помню, что нахожусь в состоянии a_1, и поэтому беру 2, оставаясь в состоянии a_1».

Приводим состояние игры после четырех ходов:

Ход	Человек	Автомат
1-й	3	4
2-й	4	2
3-й	1	4
4-й	4	2
	12	12

Ясно, что оставшийся один единственный предмет достался человеку: ведь еще можно сделать ход. После этого у автомата всего 12 предметов и он победил, ибо 12 — число четное.

Не правда ли красиво играет автомат, руководствуясь граф-схемой, которая очень просто «подсказывает», что делать?

Приведенная граф-схема является полезным инструментом для ведения игры, однако, как правило, последний ход человек определяет без использования схемы, ибо при малом количестве оставшихся предметов формальное использование схемы невозможно, а «увидеть» хороший ход легко.

Завершая рассказ об одном из методов формализации, часто используемом в кибернетике, мы приводим текст задачи об одном практически важном (уже не в занимательном смысле) автомате и надеемся, что читатели сочтут интересным составить граф-схему его работы. Полное решение этой задачи мы приводим в разделе «Ответы и решения».

АВТОМАТ НАВОДИТ ПОРЯДОК

На заводе имеется два цеха — цех *A* и цех *B*. Детали, изготовляемые в каждом из цехов, поступают на общий конвейер и используются при сборке узлов некоего устройства. Условимся детали, изготовляемые в цехе *A*, обозначать буквой *A*, а детали из цеха *B* — буквой *B*. На конвейере детали при поступлении из цехов образуют беспорядочную последовательность, например: *AABBBBAABBBBAAABBAAAAABBBB*. Для последующего использования детали следует рассортировать так, чтобы они на конвейере образовывали тройки:

ABA ABA ABA

Требуется сконструировать автомат (для нас это означает, что следует вычертить его граф-схему), который сумеет создать из деталей, образующих сначала беспоря-

дочную последовательность, новую последовательность, состоящую из упорядоченных троек.

Рекомендуется использовать следующие соображения: пусть справа от нашего автомата располагается беспорядочная последовательность, а слева — последовательность из троек *ABA*. Рис. 9 показывает суть предложения. Чтобы обеспечить требуемую деятельность автомата,

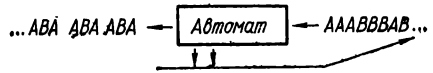

Рис 9

поручим ему сбрасывать ненужные детали на другой вспомогательный конвейер, который возвратит детали на главный. Автомат должен ждать первой в тройке детали *A* и затем, запомнив это, выбирает из тех деталей, которые подносит ему конвейер, деталь *B*. Дождавшись деталь *B*, он должен помнить, что две из трех деталей тройки он уже получил и теперь вновь ждет деталь *A* (ненужные детали *B* он сбрасывает при этом на вспомогательный конвейер). После того, как третья деталь будет пропущена автоматом в линию, он должен начинать все сначала.

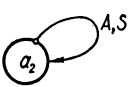

Рис. 10

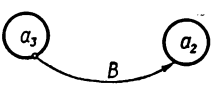

Рис. 11

Условимся об обозначении, которое потребуется при вычерчивании граф-схемы. Рис. 10 обозначает, что в состоянии a_2 ненужной является деталь *A* и автомат ее сбрасывает, а сам остается в состоянии a_2. Рис. 11 обозначает, что автомат в состоянии a_3 увидел *B*, пропустил ее мимо себя, не сбрасывая, а сам перешел в состояние a_2.

Если вы сами составите граф-схему этого автомата, то увидите замечательные возможности формализаций, которые предлагает кибернетика.

Разумеется, такой способ описания деятельности автоматов не является единственным в кибернетике.

Рассказ-задача 3

НОВЫЙ СЕКРЕТ АЛИ-БАБЫ

Известно, что, по сказке, Али-Баба попадал в заколдованную пещеру разбойников, используя волшебную фразу «Сезам откройся». Стоило Али-Бабе забыть волшебную фразу — и в пещеру он бы не попал. Все зависело от его памяти. Сегодня это кажется очень простым испытанием, и перед Али-Бабой, желающим попасть в пещеру, ставится более сложная задача, решение которой требует сообразительности.

Вот что происходит у пещеры «в наши дни». Появившись перед пещерой, Али-Баба обнаруживает бочонок, который может вращаться вокруг вертикальной оси. В днище бочонка имеются четыре совершенно одинаковых и симметричных отверстия, в любое из которых можно опустить руку. В бочонке под каждым из отверстий помещен сосуд. На ощупь можно определить положение любого сосуда (находится он кверху горлышком или днищем). Что-либо увидеть через отверстие невозможно.

Правила обращения с бочонком следующие. В бочонок можно одновременно опустить только две руки. Опущенными в отверстия руками можно при желании перевернуть один (любой) или оба сосуда. (Можно и не переворачивать). После действия с сосудами руки из бочонка вынимаются и он сам начинает вращаться. Затем бочонок останавливается. Узнать те отверстия, в которые перед вращением опускались руки, невозможно: бочонок после вращения занимает случайное положение, а отверстия по внешнему виду неразличимы. Затем вновь можно в любые два отверстия опустить руки и манипулировать с сосудами. Делается это для того, чтобы добиться одинакового расположения сосудов в бочонке — все горлышком вверх или вниз. После этого дверь откроется.

Это — пример задачи, за решение которой охотно возьмутся кибернетики. Речь идет о том, чтобы, пользуясь

разрешенными действиями, найти такую их последовательность, при которой дверь всегда откроется.

Какие действия нам разрешены?

Первое действие — распознавание положения двух любых сосудов. Это делается руками на ощупь. Второе действие — переворачивание одного или двух сосудов (или непереворачивание обоих).

Подчеркнем, что суть первого действия состоит в р а с п о з н а в а н и и того, каким же образом расположены сосуды под каждой рукой. Второе действие является более активным. С его помощью мы оперативно вмешиваемся в ситуацию. Первое действие будем называть **распознавателем**, а второе — **оператором**. Последовательность только таких действий (ведь никакие другие не разрешены) должна образовывать беспроигрышную процедуру открытия двери. При этом факт случайного расположения отверстий после вращения никак не должен отражаться на успешности нашей работы.

Необходимо, как говорят кибернетики, сформулировать такой алгоритм открытия двери, что при любом начальном расположении сосудов и при любых вариантах остановки после вращения бочонка через некоторое, заранее известное, число ходов дверь будет открыта.

Мы ввели часто используемый в кибернетике (а с легкой руки кибернетиков и в самых разнообразных отраслях деятельности человека) термин — **алгоритм**.

Алгоритм — одно из основных первичных понятий не только кибернетики, но и человеческого знания вообще. Человек имеет дело с алгоритмами решения самых разнообразных задач: это алгоритм решения систем линейных уравнений, алгоритм извлечения квадратного корня из многозначного числа, алгоритм деления отрезка произвольной длины на n равных частей, алгоритм обработки деталей на станке, алгоритм химического процесса, алгоритм игры королем и ладьей против короля и многое-многое другое.

Если мы знаем алгоритм, то значит можем решить систему линейных уравнений с любыми допустимыми коэффициентами, извлечь корень из любого числа (если это вообще возможно), поставить мат королю, располагая королем и ладьей, независимо от того, какая позиция была начальной. Словом, алгоритм — это секрет успеха в решении целого класса задач.

Используемые алгоритмы задаются в самых разнообразных формах. Мы выделяем алгоритмы-формулы, например формула вычисления среднего арифметического n чисел:

$$a = \frac{a_1 + a_2 + a_3 + \cdots + a_n}{n}$$

или какая-нибудь другая. Часто встречаются алгоритмы-инструкции, например алгоритм извлечения квадратного корня. Очень удобно задавать алгоритмы графически. Графический способ задания алгоритмов особенно распространен в кибернетике.

Однако, прежде чем продвигаться дальше, а наша цель — сформулировать (в какой-то форме) алгоритм открытия двери, — мы приведем весьма подходящее для наших целей описание алгоритма, принадлежащее академику А. А. Ляпунову:

«Алгоритмом для решения предложенной задачи называется объединение элементарных актов и проверяемых условий, которые обеспечивают такой порядок работы (т. е. проверки условий и выполнения элементарных актов), который при любых начальных данных, т. е. исходной информации, приводит к правильному ответу»[1].

Узнаете знакомые действия? «Проверка условий», по Ляпунову, — это распознаватель. «Выполнение элементарного акта» — это оператор.

---

[1] Ляпунов А. А. О некоторых общих вопросах кибернетики.— В кн.: Проблемы кибернетики, вып. I, 1958.

Приведенное описание алгоритма не является единственным. Имеется много других, ему эквивалентных. Для нас, однако, это описание удобно тем, что Ляпунов очень четко подчеркнул структуру алгоритма — всякий алгоритм есть совокупность распознавателей и операторов.

Алгоритм открытия двери в пещеру мы и намерены представить в виде совокупности распознавателей-операторов.

На будущей граф-схеме алгоритма распознаватели будут обозначаться кружками, а операторы прямоугольниками. (Такие обозначения являются общепринятыми, предложены они профессором Киевского университета Львом Аркадьевичем Калужниным).

Рис. 12

Рис. 13

Возвратимся к основной задаче — будем искать алгоритм в виде последовательности распознавателей и операторов.

Легко понять, как за два хода можно сделать так, что три из четырех сосудов будут обращены горлышком вверх. Осуществим первое распознавание по диагонали — распознаватель I (рис. 12 — черточкой соединены те отверстия, в которые опускаются руки). Тогда возможны три исхода: либо один сосуд расположен горлышком вверх, либо оба, либо оба горлышком вниз. Если один горлышком вверх, то второй сосуд переворачиваем и он тоже становится горлышком вверх, если оба расположены горлышком вниз — переворачиваем оба.

Если первое распознавание показало, что оба сосуда расположены горлышком вверх, то вынимаем руки из отверстия, не изменяя положения сосудов.

После этого бочонок вращается и занимает после вращения некое случайное положение. Второе распознавание проводим так, как показано на рис. 13 — распознаватель II. Если при этом обнаруживается, что сосуды расположены различно, то нам следует расположить их оба горлышком

вверх и затем переходить к следующему такту работы. Если оба сосуда, расположенных под руками, находятся горлышком вверх, то с сосудами ничего не надо делать. Следует вынуть руки из бочонка и ждать результата вращения.

Ясно, что после второго такта работы три из четырех сосудов расположены горлышком вверх: два по одной из диагоналей и два по какой-то из сторон.

После вращения бочонка выполняется третий такт работы. Предлагается применить распознаватель I. На рис. 14 показано, что после применения распознавателя I мы

Рис. 14

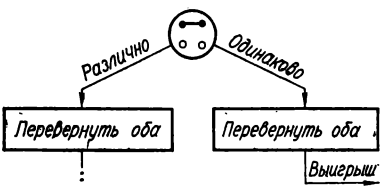

Рис. 15

будем знать, что либо оба сосуда расположены горлышком вверх, либо один из сосудов расположен горлышком вверх. Что делать в последнем случае ясно — тот сосуд, который расположен горлышком вниз, следует перевернуть, и задача будет решена. Более интересной является ситуация вторая — оба сосуда расположены горлышком вверх. Что делать в этом случае? Предлагается неожиданное и очень оригинальное решение — следует применить оператор: левый верхний сосуд перевернуть.

После этого вдоль одной из сторон какие-то два сосуда будут расположены горлышком вверх, а вдоль другой — горлышком вниз.

После очередного вращения бочонка и его остановки в случайном положении выполняется четвертый такт работы. Применяется распознаватель II, и при этом могут иметь место два случая (рис. 15). Рекомендуемые операторы,

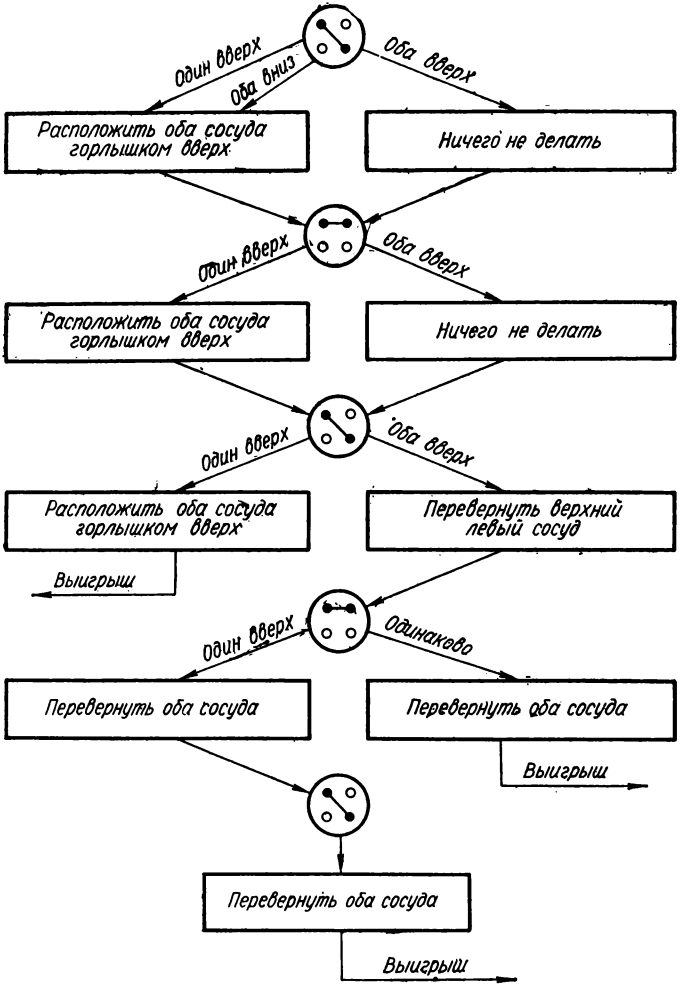
один вверх
оба вниз
Оба вверх
Расположить оба сосуда горлышком вверх
Ничего не делать
один вверх
Оба вверх
Расположить оба сосуда горлышком вверх
Ничего не делать
один вверх
Оба вверх
Расположить оба сосуда горлышком вверх
Перевернуть верхний левый сосуд
Выигрыш
один вверх
Одинаково
Перевернуть оба сосуда
Перевернуть оба сосуда
Выигрыш
Перевернуть оба сосуда
Выигрыш

Рис. 16

которые следует применять на этом такте, приведены в квадратах. Интересно, что оба оператора совпали.

На пятом такте работы применяется распознаватель I и независимо от результата распознавания применяется оператор «перевернуть оба».

После этого четыре сосуда оказываются в одинаковом положении. Все наши рассуждения сведены в одну граф-схему, изображенную на рис. 16.

Итак, современный Али-Баба получил в свое распоряжение волшебный алгоритм, алгоритм, который всегда выходит победителем в борьбе со случаем. Действительно, располагая граф-схемой алгоритма, Али-Баба всегда сумеет расположить четыре сосуда либо горлышком вверх, либо — вниз. И то обстоятельство, что бочонок после каждого вращения занимает случайное положение, для алгоритма не страшно.

Алгоритм закрепил нашу смекалку в очень удобной для пользования форме, в форме граф-схемы. С представлением алгоритма в форме граф-схемы в кибернетике, и в частности при программировании, приходится встречаться на каждом шагу.

Рассказ-задача 4

ЭМИЛЬ ПОСТ И ЕГО «МАШИНА»

Слово «машина» взято в кавычки не случайно. История термина «машина Поста» заслуживает отдельного рассказа. История эта началась в 1967 году, когда в статье для журнала «Математика в школе» профессор Московского университета Владимир Андреевич Успенский использовал этот им же придуманный термин для того, чтобы образнее рассказать об алгоритмической системе американского математика и логика Эмиля Поста.

Профессор Успенский полагал, что рассказ о воображаемой машине Поста и рассказ о том, как ею управлять поможет читателям разобраться в алгоритмической системе Поста. Сам же Эмиль Пост никогда в своих работах термина «машина» не употреблял и даже не подозревал, что его идеи можно интерпретировать так, как это сделал В. А. Успенский.

Машина Поста, придуманная Успенским, это машина, которая работает по программам, составленным для нее человеком. Программирование для этой воображаемой машины Успенский рассматривал как одно из хороших средств введения в программирование вообще. Научившись управлять машиной Поста, легко перейти к программированию для любой цифровой вычислительной машины и с программным управлением. В упомянутой статье Успенский высказал убеждение, что изучение основ программирования обязательно следует начинать в школе, а учить программированию для машины Поста он считал возможным в четвертом классе.

Мы дважды подчеркнули, что Успенский рассматривал машину Поста, как воображаемую, мысленную конструкцию. Он акцентировал внимание читателя на том, как она работает, как ею управлять, и не ставил перед собой задачи рассказать о конструкции такого рода машин, т. е. использовал понятие «машина» как программист, а не как конструктор.

Идея Успенского оказалась очень привлекательной для педагогов. Предложение Успенского начинать ознакомление учащихся с программированием через изучение машины Поста было проверено на многих экспериментальных занятиях в первой в нашей стране школе юных кибернетиков. Такая школа, в которой занимаются ребята 7—9 классов, работает в Крыму. Опыты оказались успешными, школьники седьмых и даже шестых классов с увлечением занимались программированием для воображаемой машины Поста и своими успехами подтвердили педагогическую

гипотезу Успенского. Однако программирование на бумаге, без возможности увидеть работу «живой» действующей машины, хотелось бы закрепить программированием «за пультом». Для этого оставалось создать машину Поста в металле.

И такая машина была разработана. Первый экземпляр машины Поста был изготовлен в Симферопольском государственном университете в 1970 г. В скором времени эти простейшие ЦВМ с программным управлением будут выпускать на Украине.

От истории машины Поста пора перейти к рассказу о программировании по Посту.

Эмиль Пост еще в 1936 году разработал простейшую из вообще возможных систем обработки информации и показал, что предлагаемая им система обладает весьма важным свойством алгоритмической полноты.

В чем же суть его системы?

Во-первых, Пост предложил всякую информацию, подлежащую обработке по существу, предварительно преобразовывать по форме. Он требует каждое слово, данное для обработки, предварительно переписать с привычного алфавита на алфавит двоичный, то есть на двубуквенный алфавит. С подобной формальной обработкой текста мы встречаемся при обращении к телеграфу. Телеграфист, не меняя смысла, содержания текста, заменяет его форму. При этом телеграфист использует двубуквенный алфавит — его буквы: «точка» и «тире».

Еще раз подчеркнем, что «по Посту» сначала всякая информация непременно должна быть записана в виде слов двубуквенного алфавита.

Второй важной частью его системы является предложение обрабатывать каждое данное слово (уже записанное в двоичном алфавите) побуквенно: букву за буквой. Это приводит к очень простой системе элементарных действий, используя которые, можно, следуя Посту, осущест-

вить любое преобразование двоичного слова, если это вообще возможно.

Как же устроена машина Поста?

Сначала мы рассмотрим, как описал ее Успенский. После того как определенная информация переписана в двоичном алфавите, она буква за буквой заносится на информационную ленту машины. Информационная лента разбита на одинаковые секции (рис. 17). Число секций на ленте бесконечно. Это значит, что по мере необходимости мы можем подклеивать к информационной ленте справа или слева нужное число секций.

В качестве одной из букв двоичного алфавита, придуманного Успенским, будем использовать **метку** (V), которая может быть помещена в какую-либо секцию ленты,

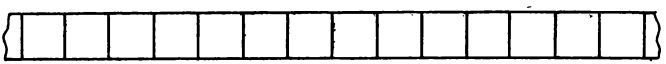

Рис. 17

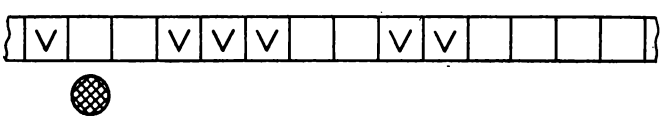

Рис. 18

второй буквой будет «пустота» в секции. Если секция содержит метку, то она называется **отмеченной**, в противном случае она содержит «пустоту» и называется **неотмеченной**. Никаких других знаков кроме «метки» и «пусто» в секциях информационной ленты не могут помещать в процессе работы ни человек, ни машина.

В показанной на рис. 18 ленте часть секций отмечена. Распределение отмеченных и неотмеченных секций определяет **состояние** ленты. Вдоль информационной ленты движется **каретка** (заштрихованный кружок). Она может двигаться только скачками, каждый раз либо точно на одну секцию

вправо, либо — влево. На рис. 18 каретка расположена под неотмеченной секцией.

Конкретное состояние ленты с указанием, где расположена каретка, определяет **состояние машины.** С помощью каретки машина может **различать, распознавать,** является ли конкретная секция ленты, под которой расположена каретка, отмеченной или неотмеченной. Каретка может стереть метку, если она имеется в секции, может поместить метку в пустую секцию. С помощью каретки осуществляется побуквенное преобразование конкретного двоичного слова.

Задачей машины Поста является *преобразование состояния информационной ленты.* Преобразование это машина осуществляет, руководствуясь алгоритмом, разработанным человеком. Поскольку машина Поста есть программно-управляемая машина, то алгоритм ее работы оформляется в виде программы.

Система команд машины Поста содержит всего шесть команд:

$a. \rightarrow b$ — команда, по которой машина сдвинет каретку вправо и перейдет к выполнению команды с номером b (здесь a — номер команды, выполнив которую, машина сдвинется на одну секцию вправо; b — номер следующей команды);

$a. \leftarrow b$ — команда сдвига каретки влево;

$a. \vee b$ — команда, по которой машина отметит пустую секцию;

$a. \updownarrow b$ — команда «стереть метку»;

$a!$ — команда «стоп»;

$a.? \begin{matrix} \nearrow b \\ \searrow c \end{matrix}$ — команда передачи управления по содержимому обозреваемой секции. Если в обозреваемой кареткой секции имеется метка, то в качестве следующей команды машина выбирает в программе команду с номером c, если секция не была отмечена, то в качестве следующей выбирается команда с номером b.

Рассмотрим пример использования команд для решения простой задачи. Располагая возможностями машины, нужно составить программу, работая по которой машина (рис.19) сотрет левую метку (под ней находится каретка) и присоединит ее к меткам, расположенным на информационной ленте где-то на конечном расстоянии справа.

Приводим **текст программы:**

1. ↕ 2 — выполнив команду с номером 1, машина сотрет метку и перейдет к выполнению команды с номером 2.

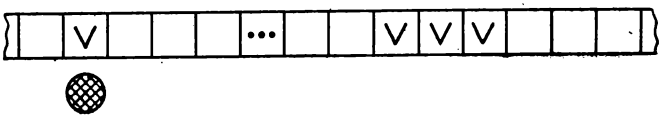

Рис. 19

2. → 3 — сдвиг вправо и переход к 3-й команде.

3.? $\begin{matrix}2\\4\end{matrix}$ — принятие решения. Если обозреваемая секция пуста, то — переход к команде второй, в противном случае — к четвертой.

4. ← 5 — сдвиг влево.

5. ∨ 6 — выполняется команда 5 — машина ставит метку в пустую секцию.

6! — остановка машины.

С целью привлечь внимание читателя к удивительным возможностям машины Поста расскажем о том, как машина Поста может выступить соперником человека в настольной игре. Оказывается простота системы команд (очень уж элементарны команды) не ограничивает возможностей машины в целом.

Играть будем в один из вариантов игры Баше. В игре участвуют двое (в нашем случае: человек и машина Поста), ходят поочередно. За каждый свой ход каждый из играющих может взять из общей группы предметов (число предметов 21) 1, 2, 3 или 4 предмета. Победителем считается

тот, кто предоставляет сопернику взять последний предмет.

Вместо 21 предмета используем информационную ленту машины, на которой отметим массив из 21 метки (рис. 20). Каретку поместим под самой левой меткой. Потребуем от человека, чтобы при его ходе он сначала решил, сколько меток он намерен стереть, и лишь после этого стирал

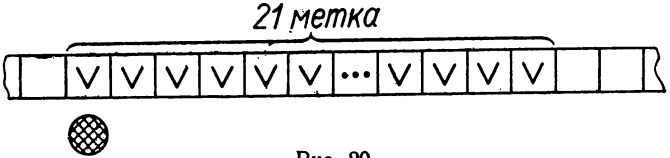

Рис. 20

метки в секциях так, как это показано на рис. 21. В игру человек вступает всегда первым.

Наша задача заключается в том, чтобы найти алгоритм игры «за человека» и затем его «омашинить» — представить в виде текста программы для машины Поста.

Что касается идеи алгоритма, то она очень проста. После хода соперника следует стирать столько меток, чтобы обоими соперниками вместе было стерто пять меток.

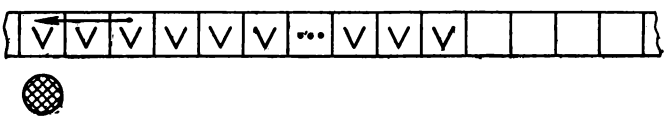

Рис. 21

Тогда при очередном своем ходе соперник будет вынужден начать стирать метки в следующей пятерке и, следовательно, будет стирать и последнюю, 21-ю метку.

Итак: стирай столько меток, чтобы в сумме со стертыми противником за ход их было пять.

Приводим текст программы, которая и выразит эту стратегию. Комментарии помогут понять ход игры.

1. ?(2, 1) — очень интересно начало программы. Машина ждет появления неотмеченной секции над кареткой, это для нее будет обозначать, что человек завершил свой ход и очередь за ней;

2. → 3
3. → 4
4. → 5
5. → 6 — выполнив эти команды, машина сдвинет каретку под пятую метку. Так как человек за один свой ход не может стереть более 4 меток, то пятая метка всегда есть;

6. ↕ 7 — стирается метка в секции;

7. ← 8
8. ?(9, 6) — каретка движется влево и стирает все нестертые человеком метки до того момента, пока не встретит пустую секцию;

9. → 10
10. ?(9, 1) — каретка сдвигается вправо под отмеченную секцию и «ждет» очередного хода человека.

Две приведенные выше программы помогли нам показать, в чем суть **программирования по Посту**. Программируя по Посту, мы сначала д о г а д а л и с ь об идее решения задачи, при этом мы все время «помнили», что идею придется в о п л о щ а т ь в текст программы, а программа это ни что иное, как упорядоченная разумным образом последовательность разрешенных программисту для использования команд.

Программирование по Посту (а лучше сказать по Успенскому, ибо ему принадлежит основная педагогическая идея такого введения в программирование) содержит в себе все основные детали профессионального программирования. Использование машины Поста человеком в принципе ничем не отличается от использования человеком таких машин, как «Минск-32», «БЭСМ-6» или каких-либо других ЭЦВМ серийного производства.

В самом деле, после разработки идеи решения (иногда говорят — **проекта алгоритма**) необходимо составить программу, что требует, с одной стороны, знания системы команд, с другой — известного воображения, которое поможет проект алгоритма превратить в текст программы. Затем следует составленную программу о т л а д и т ь, т. е. убедиться в ее безошибочности, что делается путем проверки на частном случае. Иногда отладка является делом нелегким — нужно, например, придумать простой, но достаточно общий частный случай и терпеливо вручную «прогнать программу» на этом частном случае.

Программирование для абстрактной машины Поста предъявляет к занимающемуся такие же требования, как и программирование для любых других машин или в любых иных алгоритмических системах.

Обратим внимание еще на одну аналогию программирования по Посту, в этот раз — с решением шахматных задач. Шахматист, рассматривающий шахматный этюд, располагает некоторой информацией о начальном состоянии этюда. Эту информацию он черпает, рассматривая «информационное поле» — шахматную доску с конкретным расположением фигур. В распоряжении шахматиста четко очерченные возможности вмешательства в расположение фигур — это правила движения фигурами. Прежде чем сделать ход, шахматист предварительно прогнозирует возможные ситуации. И лишь убедившись в безошибочности системы ходов, выписывает текст «окончания».

Еще раз подчеркнем, что шахматист «разыгрывая» окончания на стандартном информационном поле, располагает четкими и ограниченными возможностями и обязан проявить смекалку, вдумчивость и воображение. Результатом его мыслительной деятельности должна стать «программа шахматных мероприятий», которая всегда приведет к успеху. При этом программа записывается в общепринятых условных обозначениях (использовать можно только предусмотренные шахматным кодексом правила шахматной

нотации — правила записи отдельных ходов и всей партии в целом).

При программировании по Посту (или по Успенскому) мы также располагаем «полем для игры» — это информационная лента, на которой стоят «фигуры» (метки). Программирование также требует сообразительности и завершается текстом программы в принятых условных обозначениях.

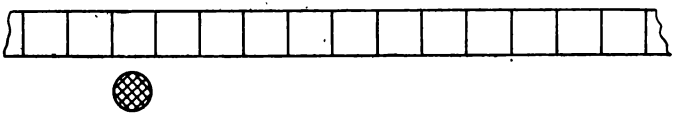

Рис. 22

Задание на программирование можно формулировать так, как задают «этюд» в шахматах.

Пример. Задача В. Успенского.

На информационной ленте (рис. 22) либо вправо, либо влево от секции, под которой расположена каретка, находится массив меток. Расстояние до массива выражается конечным числом. Необходимо составить программу, работая по которой, машина Поста найдет этот массив и присоединит к нему метку. Программист начинает и выигрывает.

Попробуйте составить текст программы. Решение, предложенное Успенским, мы приводим в разделе «Ответы и решения».

Все вышесказанное свидетельствует о том, что программирование по Посту — Успенскому — это искусство придумывания программ.

Для школьников очень интересно разобраться и в том, как же работает машина Поста, руководствуясь текстом программы. Это позволит ответить на вопрос, где же рождается автоматизм в работе машин с программным управлением.

Рассказ о действующей модели мы рассматриваем, как самостоятельную тему. Описание конструкции машины дано в приложении. Это уже задача по технической кибернетике.

Рассказ-задача 5

МАШИНА ТЬЮРИНГА И НЕОБЫЧНЫЕ АРИФМЕТИКИ

Задача, которую предстоит решить, заключается в следующем. Пусть даны два целых положительных числа в различных системах счисления, например одно число в троичной системе счисления, а другое — восьмиричной. Необходимо отыскать алгоритм вычисления суммы этих чисел. При этом сумма чисел должна быть записана в виде числа в любой, наперед заданной, системе счисления. В частности, сумма приведенных в примере чисел может быть (если мы этого потребуем) числом троичной, восьмиричной или какой-либо иной системы счисления.

Алгоритм будем разыскивать в виде машины Тьюринга. Иначе говоря, мы собираемся «сконструировать» специальную машину, машину, предназначенную исключительно для решения поставленной выше задачи.

Машина Тьюринга — это еще одна форма существования алгоритма, еще один способ фиксирования и «изображения» алгоритма. Наименование «машина Тьюринга» для очень интересной формы существования алгоритма взято в честь английского математика Аллана Мэтисона Тьюринга.

Если, по Посту, решить задачу — это значит составить текст программы, по Ляпунову, решить задачу — значит вычертить граф-схему, в которой разумным образом чередуются распознаватели и операторы, то, по Тьюрингу, решить задачу — это значит, руководствуясь идеей задачи, сконструировать подходящую для данного случая машину.

Что значит сконструировать? Какой смысл вкладывается в этот термин в данном случае?

Мы уже говорили о том, что профессор В. Успенский, разрабатывая машину Поста, не ставил перед собой инженерной задачи. Его описание машины с инженерной точки зрения было далеко неполным. Он ограничился

рассказом только о тех деталях ее конструкции, которые потребуются пользователю машиной — программисту. В данном случае ситуация аналогична. Описывая машину Тьюринга, мы ограничимся принципами работы машины и научимся «конструировать на бумаге».

Машин Тьюринга можно сконструировать много. Поскольку каждая из них есть машина Тьюринга, то она будет иметь много общего с другими. С другой стороны, каждая машина Тьюринга есть специализированная машина. Это значит, что она чем-то отличается от других. Заметим, что описание машины Тьюринга напомнит вам описание машины Поста. У них есть общие черты и есть отличия. В дальнейшем мы покажем, что машину Поста можно рассматривать как частный случай машины Тьюринга.

До 1973 года никто не задумывался о создании действующих машин Тьюринга, не абстрактных, воображаемых машин, а конструкций в металле. В Малой Академии наук школьников Крыма «Искатель» специально для педагогических экспериментов по программированию с 1973 г. используется первая в мире действующая модель машины Тьюринга, разработанная в Симферопольском университете. Опыты по применению модели при изучении основ теории алгоритмов и программирования оказались очень интересными.

Итак, что общего в различных машинах Тьюринга?

Каждая машина имеет **информационную ленту.** Лента бесконечна и разбита на одинаковые **секции.** Вдоль информационной ленты во время работы движется **каретка.** Движение каретки аналогично движению каретки в машине Поста — она движется скачками; каждый раз либо точно на одну секцию вправо, либо точно на одну секцию влево.

В отличие от машины Поста каретка машины Тьюринга может находиться в процессе работы машины в р а з л и ч н ы х с о с т о я н и я х, меняя их скачкообразно (смысл термина «состояние» будет подробно раскрыт ниже).

Для обозначения конкретного состояния мы будем использовать символы так называемого **внутреннего алфавита** машины:

$$Q = \{q_1, q_2, q_3, \ldots, q_n\}.$$

Внутренний алфавит конкретной машины Тьюринга есть конечный набор символов.

Машины Тьюринга могут отличаться друг от друга числом символов, входящих во внутренний алфавит. Это значит, что число состояний этих машин различно.

Другим важным отличием машин Тьюринга от машин Поста является то, что для каждой конкретной машины Тьюринга проектируют ее собственный **внешний алфавит.** Символы этого внешнего алфавита могут р а с п о з н а в а т ь с я и в ы п и с ы в а т ь с я кареткой. Пост использует единый стандартный для любой задачи двоичный алфавит. Тьюринг для каждой машины определяет алфавит заново. Одна машина Тьюринга может отличаться от другой внешним алфавитом.

Алфавит может быть таким:

$$A_1 = \{0, 1, 2, \ldots, 8, 9, \Delta\}$$

или таким:

$$A_2 = \{\alpha, \beta, 5, \Delta\}.$$

Конструирование машины сводится к разработке так называемой **функциональной схемы** машины. Коль скоро функциональная схема вычерчена, машина сконструирована.

Приведем несложный пример конструирования машины Тьюринга.

На информационной ленте машины Тьюринга (рис. 23) имеется два массива палочек (I), разделенных звездочкой (*). Необходимо разработать функциональную схему машины Тьюринга, которая крайнюю справа палочку правого массива сотрет и запишет ее вместо звездочки. Каретка в состоянии q_0 обозревает крайнюю справа палочку. Символом Δ обозначены пустые секции ленты.

Идея решения данной задачи может быть такой: стереть палочку и затем, продвинувшись влево под звездочку, ее стереть и вместо нее вписать палочку.

Помним о том, что наши возможности ограничены: мы можем использовать умение каретки сдвигаться скачками, умение ее распознавать символы внешнего алфавита в обозреваемой секции информационной ленты, умение заменять один из символов внешнего алфавита на любой

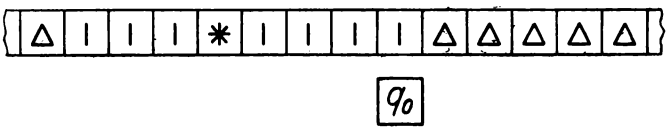

Рис. 23

другой из этого же алфавита. Наконец, мы можем использовать возможность смены состояния каретки. Конечно мы можем и остановить работу в необходимый момент.

Машина Тьюринга работает тактами, на каждом она выполняет одну комплексную команду. Структура команды имеет вид:

Указание о смене символа	Указание о сдвиге каретки	Указание о смене внутреннего состояния

Пример. $\boxed{\alpha П q_7}$ — выполняя эту команду, машина должна в обозреваемую секцию поместить символ α, затем сдвинуться на один шаг вправо и сменить свое предшествующее состояние на состояние q_7.

Еще пример. $\boxed{5 Л q_0}$ — в обозреваемую секцию вписывается «5», делается шаг влево и состояние меняется на q_0.

Совокупность таких команд и образует функциональную схему машины. Все схемы имеют единую стандартную форму (табл. 1).

Таблица 1

	q_0	q_1
|	$\Delta Л q_1$	$Л$
*		| |
Δ		

В самом левом вертикальном столбце выписаны символы внешнего алфавита. В нашем примере это алфавит:

$$A = \{|, *, \Delta\}.$$

В верхней горизонтальной строке — символы внутреннего алфавита:

$$Q = \{q_0, q_1\}.$$

Внутренние клеточки схемы заполняются командами. Выше приведена функциональная схема машины для решения нашей задачи.

Как же работает машина, имеющая такую функциональную схему?

В «поле зрения» машины находится в начальный момент секция с помещенной в ней палочкой (рис. 24). Иначе

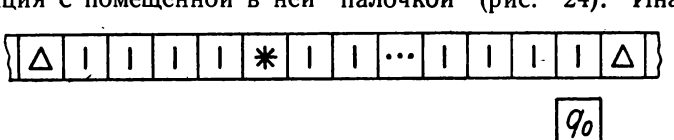

Рис. 24

говоря, машина в состоянии q_0 обозревает |. В схеме на пересечении строки с символом | и столбца с символом q_0 мы обнаруживаем команду $\Delta Л q_1$. Выполняя эту команду,

машина сотрет палочку (заменит ее символом Δ (пусто)), каретка сдвинется влево и сменит состояние q_0 на q_1.

На втором такте работы (рис. 25) каретка в состоянии q_1 обозревает палочку |. Команду, которую машине придется исполнять, находим на пересечении первой строки и второго столбца схемы. Эта команда представляет собой указание *Л* — сдвинься влево. Указания о смене обозреваемого символа нет, нет и указания о смене состояния, а это значит, что ничего менять не следует.

Сдвинувшись влево в том же состоянии q_1, каретка вновь будет обозревать секцию, содержащую символ-

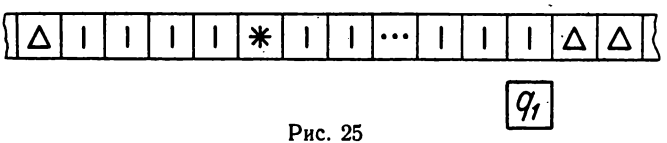

Рис. 25

палочку | и вновь ей придется исполнять ту же самую команду — сдвинься влево. Так будет продолжаться до тех пор, пока, сдвигаясь влево, каретка в состоянии q_1 не окажется под секцией со звездочкой *.

На следующем такте каретка выполнит команду, помещенную во второй строке на пересечении со вторым столбцом: |! Эта команда расшифровывается так: замени * на | и остановись.

Не правда ли, конструирование машины Тьюринга напоминает программирование? Мы располагаем некоторыми возможными действиями машины, умеем организовать отдельно взятые команды (они состоят из трех отдельных указаний), знаем, как из отдельных команд составить функциональную схему. Схему можно рассматривать, как специфический вид программы, в которой команды, в отличие от команд для машины Поста, не имеют номеров. Однако это обстоятельство не мешает машине использовать одну и ту же команду столько раз, сколько необходимо.

Рассказанного достаточно, чтобы приступить к решению задачи, с которой мы начали рассказ о машинах Тьюринга.

Еще раз вернемся к условию задачи. Пусть даны числа: одно троичной системы счисления, другое — восьмиричной. Сумму будем вычислять в восьмиричной системе счисления.

Проектирование машины начнем с определения нужного нам внешнего алфавита. Так как данное число и сумма есть числа восьмиричной системы счисления, то нам потребуются цифры:

0, 1, 2, 3, 4, 5, 6, 7.

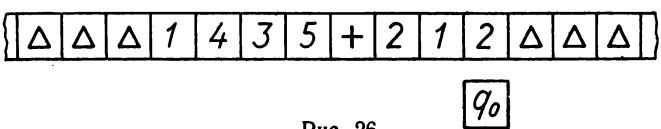

Рис. 26

Одно число от другого будем отделять знаком «+», пустую секцию — обозначать символом Δ.

Алфавит, следовательно, такой:

$$A = \{0, 1, 2, 3, 4, 5, 6, 7, +, \Delta\}.$$

Изобразим условно данные задачи на ленте (рис. 26). Справа число троичной системы счисления, слева — число восьмиричное. Каретка в состоянии q_0 обозревает цифру младшего разряда правого числа.

Идея решения задачи может быть такой: от числа, расположенного справа, отнимаем единицу (действуем, конечно, по правилам троичной арифметики), затем продвигаемся влево и к восьмиричному числу прибавим единицу (действуя уже по правилам восьмиричной арифметики). После этого вернемся к правому числу и повторим первый цикл. Работа будет продолжаться до тех пор, пока число, расположенное справа, не будет исчерпано. После того, как к левому числу прибавим единицу в последний раз, мы сдвинемся вправо, сотрем знак «+» и остановимся.

Ниже приводится готовая функциональная схема искомой машины (табл. 2).

Не правда ли, оригинальное решение? Как естественно выглядит процедура отыскивания суммы чисел. То обстоятельство, что числа заданы в различных системах счисления, оказывается не столь уже существенным.

Таблица 2

	q_0	q_1	q_2	q_3	q_4
0	$2Лq_0$	$Л$	$1Пq_3$	П	
1	$0Лq_1$	$Л$	$2Пq_3$	П	
2	$1Лq_1$	$Л$	$3Пq_3$	П	$\Delta Пq_4$
3			$4Пq_3$	П	
4			$5Пq_3$	П	
5			$6Пq_3$	П	
6			$7Пq_3$	П	
7			$0Лq_2$	П	
+	$\Delta Пq_4$	$Лq_2$		П	
Δ			$1Пq_3$	$Лq_0$	1

Приведенный пример можно интерпретировать и таким образом: пусть необходимо сконструировать машину Тьюринга, которая сможет выступить в роли троично-восьмиричного дешифратора. Сконструированная выше машина сможет это сделать — достаточно левое слагаемое считать равным нулю. Тогда, уменьшая правое троичное число

единица за единицей, мы построим слева от знака «+» новое число, равное уничтоженному, но записанное уже в восьмиричной системе.

Алгоритмы в форме машин Тьюринга интересуют математиков и специалистов по кибернетике. Многие свойства конкретных алгоритмов изучать удобнее, если эти алгоритмы представить в виде машины Тьюринга.

Все вышерассказанное может послужить всего лишь небольшим введением в знакомство с машинами Тьюринга. Те читатели, которые намерены продолжить знакомство с машинами Тьюринга, могут обратиться к книгам, перечень которых дан в конце книги. А для желающих попробовать свои силы уже сейчас мы предлагаем задачу:

На информационной ленте Тьюринга в трех секциях, в произвольном порядке записаны три различных буквы: *а*, *б* и *с*. Каретка в состоянии q_0 обозревает букву, расположенную справа (крайнюю справа). Необходимо составить функциональную схему машины Тьюринга, которая сумеет поменять местами крайние буквы.

Рассказ-задача 6

АЛГОРИТМИЧЕСКАЯ СИСТЕМА АНДРЕЯ МАРКОВА

Задача, решать которую мы будем, используя марковские алгоритмы, будет приведена в заключение. Рассказ же мы начнем с основных понятий алгоритмической системы известного советского математика Андрея Андреевича Маркова. Рассмотрим еще один способ записи алгоритмов, еще одну форму существования алгоритмов и продемонстрируем на примере достоинства и особенности такой формы алгоритмов.

Алгоритмы Маркова — это алгоритмы преобразования информации. Всякая информация, подлежащая преобразованию, должна быть записана в виде слов некоторого

конечного алфавита:

$$A = \{\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_n\}.$$

Из букв, входящих в алфавит, образуют слова. Рассматривается и **пустое слово.**

В отличие от способов преобразования слов, предложенных Э. Постом и А. Тьюрингом, А. Марков не требует п о б у к в е н н о г о преобразования данного слова. Преобразование по Маркову состоит в том, что одна часть слова (в частности это может быть и буква) заменяется другим словом. Может случиться и так, что слово из нескольких букв заменяется одной буквой.

Какими же средствами мы будем располагать, работая по Маркову? Ответ удивительный: в нашем распоряжении — один распознаватель и два оператора! Да, всего один распознаватель и два оператора. Простота системы Маркова не ограничивает ее возможностей. Можно показать, что всякий алгоритм по Посту, Тьюрингу или по Маркову взаимозаменяем. Иначе говоря, всякая задача, решенная на машинах Поста или Тьюринга, будет решена и в системе Маркова, и наоборот: всякая задача, решенная в марковской системе, может быть решена на рассмотренных нами машинах. Речь может идти не об ограничениях в принципиальных возможностях, а лишь о некоторых преимуществах решения задач в той или иной системе.

Итак, какой же распознаватель разрешен в системе Маркова? Это — **распознаватель вхождения.** Что значит вхождение и что значит распознаватель вхождения? Смысл этих терминов поясним на примерах.

П р и м е р. Пусть дан алфавит, состоящий из двух букв: *a* и *b*:

$$A = \{a, b\}.$$

В данном алфавите рассматривается конкретное слово:

aabbab.

Требуется выяснить, имеет ли место вхождение слов *ba* и *ab* в данное слово.

Ответ дать легко. В данном слове *aabbab* содержится слово *ba* и при этом точно один раз:

$$aabbab.$$

Слово *ab* в данное слово входит два раза:

$$aabbab.$$

Распознавание вхождения Марков считает операцией элементарной, нерасчленяемой и необъясняемой через какие-то более простые. Среди вхождений мы будем различать первое слева вхождение, второе слева вхождение и т. д. Распознавание вхождений ведется слева направо.

Какие же операторы можно использовать в системе А. Маркова? Первый из двух операторов называется **подстановка**. Указание о выполнении подстановки записывается так:

$$q_1 \to q_2.$$

Выполнить этот оператор (в дальнейшем будем говорить «применить подстановку») — это значит в данное слово, вместо первого вхождения слова q_1, записать слово q_2.

Пример. Дано слово

$$abba$$

и подстановка

$$a \to bb.$$

Для того чтобы применить данную подстановку, выясняем, имеет ли место вхождение слова *a* в данное слово. В данном случае имеет и поэтому вместо буквы *a* (самой левой в слове *abba*) мы записываем слово *bb* и получаем: *bbbba*.

Важное значение имеют два частных вида подстановок:

$$\to q_2$$

и

$$q_1 \to$$

В подстановке $\rightarrow q_2$ заменяемое слово есть пустое слово, а в подстановке $q_1 \rightarrow$, наоборот, — вписываемое есть пустое слово.

Пример. Дано слово в алфавите

$$A = \{1, 2, 3, 4, 5\}:$$

$$54331.$$

Требуется сначала применить подстановку $\rightarrow 2$, а затем $\rightarrow 1$.

Применяя первую подстановку ($\rightarrow 2$), получим:

$$254331$$

(вместо пустого слова слева от данного вписано слово 2). К полученному слову применим подстановку ($1 \rightarrow$) и получим:

$$25433$$

(вместо первого слева вхождения слова 1 вписано пустое слово).

Все рассмотренные подстановки называются **обычными подстановками.**

Второй оператор, используемый Марковым, называется **заключительной подстановкой.**

Указание о выполнении заключительной подстановки отличается от указания о выполнении обычной подстановки только тем, что возле стрелки ставится точка:

$$q_1 \rightarrow . \; q_2$$

Роль заключительных подстановок выяснится позже, а способ их выполнения совпадает со способом выполнения обычных подстановок.

Мы рассмотрели отдельно каждую из элементарных операций, разрешенных для использования, в системе А. Маркова. Подобно тому, как из отдельных команд мы составляли по четким правилам программы для машины Поста и группировали команды в функциональной схеме машины Тьюринга, так и в системе А. Маркова есть

правила конструирования марковских или (как он их сам называл) **нормальных алгоритмов.**

Сконструировать нормальный алгоритм — это значит выписать в определенном порядке одну подстановку за другой.

Применить нормальный алгоритм к данному слову — значит:

1. Применить первую применимую подстановку из числа входящих в алгоритм.

2. Процесс применения подстановок вести до тех пор, пока имеет место применение хотя бы одной из обычных подстановок или до тех пор, пока не будет применена первый и единственный раз заключительная подстановка.

Следующий пример позволит понять детали.

Пример. Пусть дано слово в алфавите

$$A = \{a, b, c\}:$$

$$abbacb$$

и дан нормальный алгоритм, представляющий из себя упорядоченную последовательность следующих подстановок:

$$ab \to c,$$

$$cb \to a,$$

$$a \to .$$

Требуется выяснить, какое слово будет результатом применения этого алгоритма к данному слову.

В соответствии с пунктом 1 вышеприведенной инструкции по применению алгоритма, пробуем применить первую из подстановок. Это нам удается и мы получаем слово

$$cbacb.$$

К полученному слову вновь пытаемся применить первую подстановку алгоритма. Для этого разыскиваем первое слева вхождение слова ab в данное слово, и так как такого вхождения нет, то первая подстановка оказывает-

ся неприменимой. После этого к слову пытаемся применить вторую подстановку, входящую в алгоритм, а именно:

$$cb \to a.$$

Это сделать удается, и мы имеем слово

$$aacb.$$

К полученному слову опять пытаемся применить первую подстановку алгоритма ($ab \to c$), но так как она оказывается неприменимой, то пробуем применить вторую, что нам удается. После ее применения имеем:

$$aaa.$$

Ясно, что к полученному слову нельзя применить ни первую, ни вторую подстановку и поэтому пробуем применить третью. Она оказывается применимой, после чего имеем:

$$aa.$$

Работа алгоритма (применение его) на этом заканчивается, так как третья подстановка является заключительной и должна применяться единственный раз.

Результатом применения данного алгоритма к слову

$$abbacb$$

явилось слово

$$aa.$$

Одно слово заменили другим.

Иногда **алгоритм неприменим** к данному слову. Алгоритм называется неприменимым к данному слову, если:

1) ни одна из подстановок алгоритма не применима ни одного раза;

2) процесс применения его продолжается бесконечно.

На рис. 27 приведена комментированная граф-схема процедуры применения нормального алгоритма.

1. Распознавай первое вхождение (α_1). Если вхождение было, то переходи к пункту 2, если нет — к пункту 3.

2. Применяй подстановку. Если она заключительная, то рассматривай итоговое слово, в противном случае переходи к пункту 1.

3. Распознавай следующее по порядку вхождение и, если вхождение имело место, переходи к пункту 2; если же вхождение не имело места, то вновь переходи к пункту 3. и т. д.

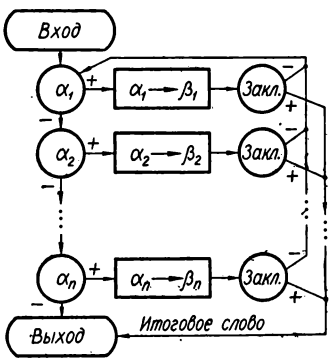

Рис. 27

Выше были рассмотрены основные понятия алгоритмической системы А. Маркова: алфавит, слово, вхождение слова, подстановка, заключительная подстановка, применение подстановки, нормальный алгоритм, применение нормального алгоритма. Наступило время показать на примерах достоинства марковских алгоритмов.

Предварительно сделаем важное замечание — в ряде случаев для построения нормального алгоритма к данному алфавиту A приходится добавлять буквы (осуществлять расширение алфавита). В таком случае говорят, что алгоритм построен над алфавитом A. Рассмотрение начнем с очень простых примеров.

З а д а ч а. В алфавите $A = \{a, b, c$ задано слово конечной длины. Требуется сконструировать нормальный алгоритм, применение которого к данному слову приведет к тому, что будет получено слово, содержащее все буквы a в его правой части.

Решением может быть такой алгоритм:

$$ab \to ba$$
$$ac \to ca.$$

Убедимся в эффективности алгоритма на примере. Пусть исходное слово такое:

cabac.

Применяем алгоритм. Пытаемся применить первую подстановку ($ab \to ba$) — она оказывается применимой. Получаем:

cbaac.

К полученному слову вновь пытаемся применить первую подстановку, но она оказывается неприменимой, и мы пробуем применить вторую из подстановок алгоритма, что нам и удается два раза. После этого работа алгоритма завершается и результатом является слово

cbcaa.

З а д а ч а. В алфавите $A = \{|, +, *\}$ задано некоторое слово конечной длины (это слово является набором только палочек). Сконструировать нормальный алгоритм, применение которого к данному слову, состоящему из четного числа палочек, приводило бы к слову $*$, а применение этого же алгоритма к слову, состоящему из нечетного числа палочек, к слову $+$.

Решением, или лучше сказать одним из решений, может быть такое:

$$||\to *$$
$$|\to +$$
$$*+\to +$$
$$**\to *$$

Для желающих попробовать свои силы в разработке алгоритмов предлагается такая задача:

В алфавите, включающем в себя цифры троичной системы счисления, дано некоторое число. Требуется построить нормальный алгоритм, применение которого к данному числу приведет к образованию нового числа, которое будет наименьшим из всех чисел, какие только можно составить из всех цифр, входящих в данное число.

Рассказ-задача 7

АЛГОРИТМ НАД АЛГОРИТМАМИ

Перейдем от рассмотрения отдельных алгоритмических систем — системы Эмиля Поста, Аллана Тьюринга и Андрея Маркова — к изучению их взаимосвязи.

Выше не раз подчеркивалось, что какая-либо задача, решенная в одной из рассматриваемых систем, всегда будет решена и в любой другой из числа рассмотренных. Это значит, что всякой программе для решения конкретной задачи на машине Поста можно поставить в соответствие хотя бы один нормальный алгоритм для решения той же задачи или построить машину Тьюринга, которая также успешно будет решать этот же класс задач. Это утверждение справедливо для любой из рассмотренных нами алгоритмических систем.

Как же найти эти новые алгоритмы?

Можно заняться решением задачи заново, не обращаясь к уже имеющемуся алгоритму в какой-либо иной алгоритмической системе. Можно попробовать поступить иначе: взять алгоритм, разработанный для решения этого класса задач в какой-либо из изученных нами алгоритмических систем, например текст программы для машины Поста, и попробовать формально, совершенно не вникая в суть задачи, так его преобразовать, чтобы получить искомое решение задачи уже в другой алгоритмической системе.

Мы избираем второй путь, будем искать способы, пользуясь которыми мы сможем формально получить алгоритм в любой из изученных нами систем без обращения к условию задачи. Исходным материалом для получения нового алгоритма будет только текст алгоритма в какой-либо из изученных систем.

Начнем с отыскания способа конструирования нормального алгоритма, исходя из текста данной программы для машины Поста.

Пусть нам дана программа, работая по которой, машина Поста (рис. 28) сотрет отдельно расположенную метку и затем присоединит ее к массиву, расположенному на некотором расстоянии справа.

1. ↕ 2 4. ← 5
2. → 3 5. ⋁ 6
3. ? 2 / 4 6. !

Прежде чем разбирать предлагаемый способ, отметим, что эта задача также была решена школьниками из

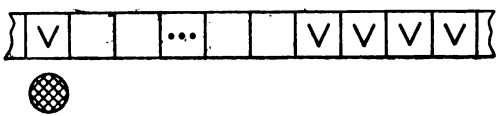

Рис. 28

Малой Академии наук «Искатель». Приводимый ниже вариант решения принадлежит девятикласснику Володе Кисляковскому.

Как и всегда, конструирование нормального алгоритма следует начинать с определения алфавита. Условимся

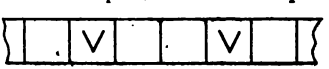

Рис. 29

о следующем: каждому состоянию информационной ленты будем ставить в соответствие слово, записанное с помощью букв: 1 и 0. Буква 1 будет заменять метку ⋁, а буква 0 — пустые секции. Например. Дано состояние ленты, изображенное на рис. 29. Ему соответствует слово 1001.

Кроме букв 1 и 0 в алфавит включим буквы: a_1, a_2,..., a_n, с помощью которых нам удастся смоделировать систему команд.

Итак, алфавит:

$$A = \{1, 0, a_1, a_2, a_3, \ldots, a_n\}.$$

Команду «стереть метку» $m. \updownarrow k$ будем моделировать подстановкой

$$\boxed{a_m 1 \to a_k 0}$$

Команду «поставить метку» $m. \vee k$ моделируем подстановкой

$$\boxed{a_m 0 \to a_k 1}$$

Команда сдвига каретки влево $m. \leftarrow k$ моделируется не одной, а двумя подстановками:

$$\boxed{\begin{array}{l} 1a_m \to a_k 1 \\ 0a_m \to a_k 0 \end{array}}$$

Аналогично, двумя же подстановками, моделируем сдвиг вправо $m. \to k$

$$\boxed{\begin{array}{l} a_m 1 \to 1a_{m} \\ a_k 0 \to 0a_k \end{array}}$$

Команда условной передачи управления $m.?\begin{matrix} k \\ n \end{matrix}$ моделируется также двумя подстановками:

$$\boxed{\begin{array}{l} a_m 0 \to a_k 0 \\ a_m 1 \to a_n 1 \end{array}}$$

И последняя команда, команда $m.!$, тоже моделируется двумя, но уже заключительными, подстановками:

$$\boxed{\begin{array}{l} a_m 0 \to .\ 0 \\ a_m 1 \to .\ 1 \end{array}}$$

Итак, что же мы смоделировали? Во-первых, — состояние ленты, во-вторых, — каждую команду в отдельности. Теперь необходимо смоделировать каретку и найти способ получения нормального алгоритма из отдельных подстановок и пар подстановок.

Посмотрите, как оригинально моделирует каретку Володя. Показываем на примере.

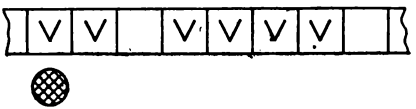

Рис. 30

Пусть дано состояние машины, изображенное на рис. 30. Этому состоянию ставим в соответствие слово

$$a_i\,1101111.$$

Еще пример. Дано состояние машины (рис. 31). Этому состоянию машины ставим в соответствие слово

$$11\,a_i\,01111.$$

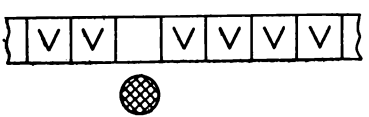

Рис. 31

Алгоритм в целом будем конструировать по следующему правилу:

1. Каждой команде машины Поста ставим в соответствие ее модель, которая представляет собой либо одну, либо две подстановки.

2. Состояние машины моделируем словом, составленным из букв 1 и 0. Букву a_i помещаем внутри слова (или в каком-либо другом месте) так, чтобы та буква, под которой расположена каретка, оказалась соседней справа от буквы a_1.

Применив эту рекомендацию для конструирования алгоритма, исходя из вышеприведенной программы, получим таблицу команд для машины с соответствующими им подстановками (табл. 3). Интересно, что столбец подстановок образовал нормальный алгоритм.

Таблица 3

Пост	Марков	Пост	Марков
1. ↕ 2	$a_1 1 \to a_2 0$	4. ← 5	$0a_4 \to a_5 0$
			$1a_4 \to a_5 1$
2. → 3	$a_2 0 \to 0a_3$	5 ∨ 6	$a_5 0 \to a_6 1$
	$a_2 1 \to 1a_3$		
3. ? 2 / 4	$a_3 0 \to a_2 0$	6. !	$a_6 0 \to .\,0$
	$a_3 1 \to a_4 1$		$a_6 1 \to .\,1$

Проверим работу полученного алгоритма на примере. Пусть исходное состояние машины такое, как показано на рис. 32. Моделью будет слово

$$a_1\,10000111.$$

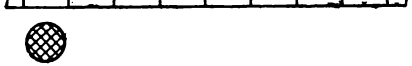

Рис. 32

К этому слову применяем первую подстановку:

$$a_1 1 \to a_2 0$$

и получаем слово

$$a_2\,00000111.$$

После применения подстановки

$$a_2 0 \to 0 a_3$$

получим слово

$$0 a_3 000000111.$$

Теперь можно применить только одну подстановку:

$$a_3 0 \to a_2 0,$$

после чего вновь придется применять

$$a_2 0 \to 0 a_3$$

и так поступать до тех пор, пока буква a_3 не окажется рядом с буквой 1 в слове

$$0000000 a_3 111.$$

Применим теперь подстановку

$$a_3 1 \to a_4 1$$

и получим:

$$0000000 a_4 111.$$

Применяем подряд несколько подстановок:

$$0 a_4 \to a_5 0,$$

$$a_5 0 \to a_6 1,$$

$$a_6 1 \to . \ 1.$$

Результатом работы алгоритма есть слово

$$0000001111.$$

Полученное слово совпадает с ожидавшимся ответом. Это успех — *мы научились формально, руководствуясь только текстом программы для машины Поста, конструировать нормальные алгоритмы.*

Возможно, что конструирование нормального алгоритма для решения этой задачи, если его осуществлять без «заглядывания» в текст программы для машины Поста, приведет к более короткому алгоритму. Однако для нас

способ привлекателен тем, что вся работа протекает формально.

Мы научились моделировать машину Поста средствами алгоритмической системы А. Маркова. Следует заметить, что задача моделирования одних машин на других является весьма важной в кибернетике. К такому моделированию прибегают, в частности, на этапе разработки нового типа вычислительных машин. В качестве примера укажем на использование моделирования функций ЭЦВМ ИЛЛИАК-IV при ее проектировании.

«Создание ИЛЛИАК-IV без использования в процессе проектирования других ЭВМ было бы невозможно. Так, на протяжении двух лет две ЭВМ средней мощности модели БЕРРОУЗ-5500 почти полностью использовались для изготовления чертежей или фотошаблонов печатных схем системы ИЛЛИАК-IV и для разработки диагностических текстовых программ для логики и аппаратурных средств системы ИЛЛИАК-IV»[1].

Конечно, наше моделирование машины Поста средствами алгоритмической системы А. Маркова очень просто в сравнении с моделированием такой супер-ЭЦВМ, какой является ИЛЛИАК-IV (ИЛЛИАК-IV — это комплекс из 65 ЭЦВМ, который обладает быстродействием в 200 млн. действий в секунду). Однако рассмотренный пример позволил нам, с одной стороны, показать взаимосвязь между алгоритмическими системами и выразить эту связь формально (по существу, мы построили алгоритм преобразования алгоритмов по форме). С другой стороны, этот пример позволил нам обратить внимание читателей на задачу моделирования одних вычислительных машин средствами других.

Подчеркнем последнюю мысль следующим примером. Представьте, что мы разработали ряд программ для ЭЦВМ

---

[1] Слотник Д. Самая быстродействующая электронно-вычислительная машина.—«Успехи физических наук», 1972, вып. 3, с. 557—575.

серии «Минск-22». Программы оказались важными для многих пользователей вычислительными машинами. Однако не у всех пользователей имеется ЭЦВМ «Минск-22», ряд из них имеет другие вычислительные машины (например М-222). Могут ли они воспользоваться нашими программами? Могут, если сумеют разработать алгоритм и программу для машины, которой они располагают. Такую программу, работая по которой их ЭЦВМ сумеет заменить текст программы на языке ЭЦВМ «Минск-22» текстом программы на своем языке. Работа их машины при решении какой-либо задачи будет протекать в два этапа: сначала текст чужой программы машина переведет на свой язык — получит рабочую программу. После этого машина будет решать задачу, руководствуясь уже понятной ей программой.

Разумеется программа-переводчик, с помощью которой чужой текст заменяется своим, должна быть такой, чтобы с ее помощью можно было бы всякую программу для другой машины заменить рабочей программой для данной ЭЦВМ. Разработка таких программ — дело сложное, требует высокой квалификации программиста. Программы очень сложны и содержат подчас десятки тысяч команд на внутреннем языке машин.

Рассмотренный пример позволил нам только очертить суть задачи.

В заключение этой главы предлагаем читателю задачу — отыскать способы разработки алгоритмов в любой из рассмотренных алгоритмических систем, опираясь на уже известный алгоритм в какой-либо из них. Один из способов — способ конструирования машины Тьюринга на основе любой программы для машины Поста — мы даем в разделе «Ответы и решения».

ЕСТЬ ЛИ ФОРМУЛА?

В качестве заключительной рассмотрим задачу о вычислении расстояния между двумя датами в днях.

Пусть в пределах одного тысячелетия указаны две сколько угодно удаленные друг от друга произвольные даты. Необходимо вычислить расстояние между ними в днях, или, иначе говоря, определить, сколько дней отделяет одну дату от другой.

Даты задаются в виде трех чисел:

День	*Месяц*	*Год*
21	апреля	1961
21	04	1961

или

12. 10. 1025,

5. 03. 1857.

Решение должно быть таким, чтобы расстояние между двумя любыми датами вычислялось единым методом. Необходимо предложить удобный алгоритм решения этой задачи.

Если бы эта задача была предложена математику, то несколько лет назад решение задачи он, вероятнее всего, стал бы отыскивать в виде некоторой расчетной формулы, попытался бы получить искомое «расстояние», как функцию от двух дат. Если R — искомое расстояние; $d_р$ — ранняя дата; $d_п$ — более поздняя дата, то искомая функция

$$R = F(d_р, d_п). \qquad (1)$$

Предлагаем читателям попробовать свои силы в подборе наиболее простой расчетной формулы типа (1). Трудности обнаруживаются быстро. Одна из трудностей состоит в том, что количество дней в месяцах неодинаковое в разные годы. Другая трудность заключается в том, что имеются высокосные и невысокосные годы.

Словом, расстояние R в днях как функцию от двух дат $d_р$ и $d_п$ записать в виде одной или даже нескольких формул совсем непросто.

Современный математик попытается привлечь для решения задачи вычислительную машину с программным управлением. Алгоритм решения задачи он будет разыскивать в форме программы для ЭЦВМ. Наш рассказ о решении задачи и будет рассказом о том, как составляется такая программа. Решать задачу мы намерены на ЭЦВМ украинского производства, на вычислительной машине «Мир-1». Нам предстоит научиться программировать для этой машины. «Как, — может удивиться читатель, — разве это возможно — научится программировать, прочтя всего несколько страниц популярной книги?» «Да, — отвечаем мы, — можно!»

Несколько замечаний о программировании, которые послужат введением в основной рассказ.

Все программы, которые мы составляем для машины Поста, функциональные схемы машин Тьюринга разрабатывались нами весьма подробно: команды писались одна за другой, при составлении программы мы учитывали все ситуации, которые могут встретиться в процессе решения задач конкретного класса. Такой способ программирования принято называть **ручным программированием.** Программирование вручную — основной способ подготовки программ и широко используется при программировании для ЭЦВМ различного класса. Знать особенности, достоинства и недостатки этого способа необходимо.

Кратко остановимся на недостатках. Одним из основных является громоздкость программ. При решении сложных задач приходится выписывать сотни и тысячи команд, что приводит к ошибкам и опискам. Если учесть, что для большинства машин команды кодируются числами восмиричной системы, то становится понятным трудность не только написания текста, но и его анализа, поиска и исправления ошибок. Программирование, даже по известному проекту алгоритма, вместе с отладкой программы могло продолжаться несколько месяцев.

Еще одним весьма существенным недостатком программирования вручную была трудность разбора программ, написанных другим лицом. Комментарии и пояснения в этом случае мало помогают, анализ текста бывает столь затруднительным, что подчас легче вновь составить программу, нежели разобраться в программе, составленной кем-то. Все это мешало обмену опытом, публикация программ была очень сложным делом.

Указанные недостатки быстро обнаружились и вскоре появились предложения по их устранению. Ниже мы рассмотрим как «программирование вручную» удается заменить **программированием автоматическим.**

Нам предстоит изучить один из алгоритмических языков и выписать алгоритм решения задачи о вычислении расстояния между датами на этом языке.

Алгоритмические языки — это специально разрабатываемые искусственные языки, предназначенные исключительно для записи алгоритмов. Текст на алгоритмическом языке является еще одной формой существования алгоритмов. Со времени появления первых алгоритмических языков (начало 50-х годов) к настоящему времени во всем мире зарегистрировано рождение и использование более 1000 различных алгоритмических языков. Каждый, кто интересуется применением вычислительных машин в настоящее время, должен иметь ясное представление о таких ведущих языках, как АЛГОЛ, ФОРТРАН, КОБОЛ, МИР.

Использование алгоритмических языков значительно изменило порядок подготовки рабочей программы. Благодаря алгоритмическим языкам труд программистов сегодня менее однообразен и обременителен и более эффективен. После появления алгоритмических языков очень сильно изменился и расширился круг людей, которые могут пользоваться вычислительными машинами. Изучив какой-либо из алгоритмических языков, к помощи вычислительной машины может обратиться биолог, химик, экономист или специалист в какой-нибудь иной отрасли

науки или производства. Причем, и это мы подчеркиваем, для подготовки задачи к решению на ЭЦВМ помощь профессионального программиста не потребуется.

Приводим схему решения задачи с применением алгоритмического языка (рис. 33).

После того, как пользователь получил в свое распоряжение алгоритм (в любой удобной для него форме), он са-

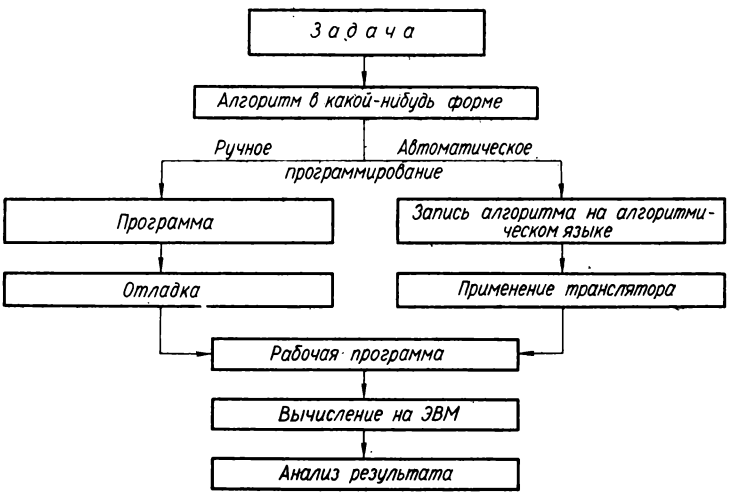

Рис. 33

мостоятельно переписывает его на подходящем алгоритмическом языке. Затем текст алгоритма «набивается» на перфоленту, перфокарты или печатается с помощью пишущей машинки так, что оказывается введенным в память ЭЦВМ. Введенный в ее память текст машина рассматривает как некую данную ей для обработки информацию. Эту информацию — текст алгоритма — машина должна

преобразовать так, чтобы в результате получить алгоритм, но уже в форме рабочей программы. **Рабочая программа** — это программа, записанная на **внутреннем языке** данной машины, т. е. на языке кодов ее команд. (Читатель, конечно, узнал знакомую задачу — задачу формального перехода от одной формы существования алгоритма к другой). Алгоритм такого формального преобразования алгоритма в этом случае является текстом программы для машины и, конечно, используя эту программу, машина сможет всякий алгоритм, написанный, например, на ФОРТРАНЕ, перевести на внутренний язык. Такие программы, с помощью которых алгоритм, данный на алгоритмическом языке, преобразуется в алгоритм в форме рабочей программы, называются **трансляторами**. На одной и той же машине можно использовать несколько трансляторов. Полученная после трансляции текста рабочая программа хранится в оперативной памяти ЦВМ и теперь уже может быть применена для решения задачи, поставленной пользователем.

Подчеркнем, что трансляторы — это сложные и искусно составленные программы. Пишутся эти программы вручную. Тексты трансляторов чаще всего набиваются на перфолентах и перфокартах и хранятся как оригиналы. Для работы используются их копии, записанные на магнитных лентах и магнитных барабанах. В последнее время стали появляться **аппаратурные трансляторы** в виде специальных узлов ЦВМ, которые могут решать единственную задачу — задачу трансляции. Такой аппаратурный транслятор имеется на ЭЦВМ серии «Мир», он осуществляет трансляцию текста алгоритма с языка МИР на внутренний язык машины.

Изучение алгоритмического языка ЭЦВМ серии «Мир» мы осуществим по краткому и неформальному его описанию.

АЛГОРИТМИЧЕСКИЙ ЯЗЫК МИР

(краткое и неформальное описание)

Прежде чем написать на алгоритмическом языке МИР текст программы, необходимо ознакомиться с основными понятиями языка. К таким понятиям относятся:

алфавит языка,

основные операторы и описания,

правила синтаксиса.

Ниже даны основные символы алгоритмического языка МИР:

1. Буквы — все латинские буквы и те из русских, написание которых не совпадает с латинскими
2. Цифры — все арабские цифры десятичной системы счисления и никакие другие
3. Знаки отношений — $>$, $<$, $\geqslant$, $\leqslant$, $=$
4. Знаки разделений — [,], (,), ,, ;, $_{10}$, .
5. Знаки арифметических действий — $+$, $\times$, $\uparrow$, $-$, /
6. Указатели встроенных функций — $\sqrt{}$, E, F, Σ, Π, SIN, COS, CTG, TG, LN, LG, ABS, EXP, S
7. Служебные слова — «ГДЕ», «ВЫЧИСЛИТЬ», «ЗАМЕНИТЬ», «РАЗРЯДНОСТЬ», «ВЫВОД», «ДЛЯ», «ШАГ», «ЕСЛИ», «ТО», «ИНАЧЕ», «НА», «СТРОКА», «СТОП», «ПРОБЕЛ», «ИДТИ», «КОНЕЦ», «ВЫПОЛНИТЬ», «ПОЛОЖИТЬ», «ДО», «ВМЕСТО», «МАССИВ», «ТАБЛИЦА», «ЗАГОЛОВОК», «СТЕРЕТЬ», «ЗАПИСАТЬ»

Как видим, среди символов имеются все десятичные цифры, все буквы русского и латинского алфавита, знаки арифметических действий, знаки разделения (скобки круглые, квадратные, точка, запятая, точка с запятой), значок$_{10}$, указатели функций: $\sqrt{}$, F (дробная часть от значения аргумента), E (целая часть от значения аргумента), Σ (знак суммы), Π (знак произведения) и $\int$ (знак интеграла).

В языке МИР разрешается при написании программ использовать несколько **служебных слов**, которые, хотя и составлены из нескольких символов, машиной рассматриваются как единый символ. Как единый символ машина воспринимает и названия функций: SIN, COS, TG, CTG, LN, LG, ABS (так обозначается функция вычисления абсолютной величины какого-либо числа) и др.

В тексте программы могут использоваться постоянные и переменные величины. Переменные во время работы машины могут менять свое значение столько раз, сколько этого требует процесс вычисления. Значением переменных величин могут быть **числа целые** и **действительные.** В роли постоянных величин могут также быть только целые и действительные числа. С комплексными числами машина «Мир» непосредственно действий вести не может. Целые числа можно задавать машине, не ограничивая себя количеством цифр. Ведя вычисления, машина может обрабатывать очень большие (по количеству цифр) числа. Действительные числа вводятся в машину в одной из форм:

1. В виде смешанного числа или в виде правильной десятичной дроби:

0.5 .5

Так изображается десятичная дробь 0,5. Вместо запятой для отделения целой части от дробной используется точка. Ноль целых можно не писать.

12.36 — изображение смешанного десятичного числа 12,36.

2. В виде нормализованного числа десятичной системы счисления: $0.856_{10} + 2$ — изображение числа 85,6 (0,856 — мантисса, +2 — порядок),

0.7123_{10} — 1 — изображение числа 0,071231 (0,71231— мантисса, — 1 — порядок).

3. В полулогарифмической форме:

$2.34_{10} + 1$ — изображение числа 23,4,

$15.065_{10} + 3$ — изображение числа 15065,

$2.05_{10} - 2$ — изображение числа 0,0205.

Переменные снабжены именами — индентификаторами. Индентификатор составляется из букв и цифр и при этом обязательно начинается с буквы. Один идентификатор от другого машина отличает по первым четырем составляющим его символам. Примеры индентификаторов:

A, X, ALFA, B1, C35, PI, S.

Служебные слова и названия функций в языке МИР не могут выступать в качестве индентификаторов. В ходе вычислений каждая переменная может быть использована только после того, как ей будет присвоено конкретное числовое значение. В языке МИР используются не только простые переменные, но и переменные с индексами. Переменные с индексами обозначаются индентификаторами специального вида.

Рассмотрим предварительно переменные с одним индексом. Пусть нам дана некоторая числовая последовательность, имеющая k членов:

$$a_1,\ a_2,\ a_3,\ \ldots,\ a_n,\ a_{n+1},\ \ldots,\ a_k.$$

Каждый член последовательности имеет свой конкретный номер (индекс). Обозначение члена последовательности на языке МИР осуществляется так:

A [1] — 1-й член последовательности,

A [15] — 15-й член последовательности,

A [N] — член последовательности с номером N,

A [K] — член последовательности с номером K.

Числовая последовательность может называться **одномерным массивом**, имеющим свое имя (индентификатор). Например:

TEMP [1] — первый член массива TEMP,

TEMP [I] — i-член массива TEMP.

Кроме одномерных массивов, в языке МИР данные и получаемые результаты могут образовывать **двумерные массивы**, которые в математике часто именуют **матрицами**. Все элементы двумерного массива принято изображать на

бумаге (и в памяти машины) в виде таблицы, состоящей из нескольких строк и столбцов.

Пусть дан массив:

2	—3	—5	1
0	4	2,5	—6
5	2	3	4

Присвоим этому массиву имя MAS1. Чтобы сообщить машине полное содержание массива, следует напечатать:

MAS1 [3, 4] = 2, —3, —5, —1, 0, 4, 2,5, —6 5, 2, 3, 4.

В квадратных скобках сначала указывается число строк, затем число столбцов. Если мы хотим обратиться к конкретному члену массива, то нужно указать имя массива и в квадратных скобках номер строки и через запятую — номер столбца. Например запись: MAS1 [2,2] + MAS1 [3,4] означает, что речь идет о сумме двух элементов одного и того же массива MAS1. Первое слагаемое есть 2-й элемент 2-й строки, а второе слагаемое есть 4-й элемент 3-й строки. Если же мы напишем MAS1 [I, K], то эта запись означает, что перед нами обозначение (имя) переменной, являющейся элементом двумерного массива MAS1. К моменту действия с этой переменной значения индексов I и K должны быть вычислены.

В алгоритмическом языке МИР допускается использование только одномерных и двумерных массивов.

ОСНОВНЫЕ ОПЕРАТОРЫ ЯЗЫКА МИР

Важнейшей частью всякой программы является набор операторов, т. е. указаний о действиях над данными величинами. Ниже описываются основные операторы языка и указываются правила их написания.

Арифметический оператор или оператор присваивания

Указание выполнить арифметический оператор задается следующим типом выражений:

$$\boxed{A = X},$$

где A — имя переменной (индентификатор), а X — конкретное число, или арифметическое выражение. Выполняя оператор, машина присвоит переменной A значение X. Например, оператор $B12 = 13_{10} + 2$ означает, что после его выполнения значение переменной B12 будет равно 1300.

Выполнение оператора

$$A[1,5] = .2_{10} - 3$$

приведет к тому, что элемент двумерного массива (пятый элемент в первой строке) будет равен 0,0002.

Обратим внимание на то, что всякий текст, предлагаемый машине «Мир», может быть введен на пишущей машинке (о других способах ввода текста речь пойдет ниже). Печатание на машинке требует отказа от привычного способа записи показателя степени.

Привычная нам запись

a^3 заменяется на A ↑ 3

или

$\sqrt[4]{b^3}$ заменяется на B ↑ (3/4).

Указание о делении делается с помощью знака / (косая черта). Например, запись на языке МИР

A ↑ 3/(B ↑ 2 — c)

означает

$$\frac{a^3}{b^2 - c}.$$

Используя знаки действий, скобки, имена переменных, функции и постоянные, в языке МИР записывают арифметические выражения.

В правой части оператора присваивания может стоять **арифметическое выражение**. Смысл термина арифметическое выражение несколько отличается от того, что под этим понимают в школьной математике. Поясним на примерах.

Пусть дан арифметический оператор

$$A[26] = ((X \uparrow 3 - 5) \uparrow 2)/(X + 2.1). \qquad (1)$$

Выполнить его — значит, зная значение X (а оно к моменту выполнения оператора должно быть машине известно), вычислить значение арифметического выражения, записанного справа, и присвоить это значение переменной A[26].

В обычных обозначениях запись правой части такова:

$$\frac{(x^3 - 5)^2}{x + 2{,}1}.$$

Не все равенство (1) называется арифметическим выражением, а лишь правая его часть. Дело в том, что оператор присваивания — это указание к действию, указание «вычислить правую часть и присвоить полученное значение переменной величине, имя которой записано в левой части оператора». Рассмотрим пример:

$$A26 = A26 + 1. \qquad (2)$$

Выполняя этот оператор, машина к известному ей значению переменной A26 прибавит единицу и в дальнейшем будет считать, что A26 имеет значение на единицу большее, нежели перед выполнением оператора (2).

Часто используются и такие записи оператора:

$$A = A + A \text{ или } B12 = (B12 + 1) \uparrow 3 \text{ и др.}$$

Оператор присваивания может в составе арифметического выражения содержать функции из перечня, приве-

денного выше. Например:

2. B32 = TG(X ↑ 2 — 3) + 17.4

11. A[1,5] = 264.3/(X — 3.54)

M. У[5] = (2 × X(1/3)) — ABS(X + TG(X/2))

Перед каждым из написанных выше операторов помещены **метки** (они заканчиваются точкой). Метки служат для того, чтобы можно было давать указание машине о выполнении того или иного оператора, метки как бы именуют оператор. Ниже мы покажем пример того, как с помощью метки программист дает указание машине переходить с одной части программы к другой.

Условный оператор

Условный оператор предназначен для того, чтобы поручать машине проверку некоторых условий и, в зависимости от выполнения этих условий, менять ход своей работы.

Указание о выполнении условного оператора машине дается в виде текста:

"ЕСЛИ" А "ТО" (P_1) "ИНАЧЕ" (P_2)

или

"ЕСЛИ" А "ТО" (P)

Буквой А обозначено условие, которое машина должна проверять. Примеры условий:

$$X \leqslant 5.12$$

$$A12 > B\ 2 + 5$$

$$B3 \geqslant 16.5 - B1$$

$$C = 0$$

Буквами P_1 и P_2 обозначены операторы. Оператор P_1 выполняется машиной в качестве следующего, если

проверяемое условие А выполнено; оператор P_2 выполняется в качестве следующего, если проверяемое условие не выполнено.

Пример.

„ЕСЛИ“ X > 20.2 “ТО“ (У = 2 × X ↑ 3) “ИНАЧЕ“ (У = 0)

Получив такой оператор, машина должна распознать, является ли значение переменной X к данному моменту работы машины больше, чем 20. 2. Если это так, то машина присвоит переменной У значение $y = 2x^3$, если же X не больше, чем 20.2, то переменной У будет присвоено значение нуль.

Второй вариант условного оператора дает указание машине выполнить оператор Р при выполнении условия А и переходить к следующему в тексте программы оператору, если условие А не выполнено.

В качестве операторов Р могут выступать не только операторы присваивания, как это имело место в приведенных примерах, а и другие. В частности, это может быть и условный оператор.

Оператор безусловной передачи управления, или оператор перехода

Этот оператор позволяет программисту давать указание машине о том, какой оператор ей следует выполнять в качестве следующего, меняя, тем самым, текущий порядок в работе машины.

Имеется две формы записи такого оператора:

“ИДТИ“ “НА“ М	или	“НА“ М

Буквой М обозначена метка того оператора, который будет выполняться в качестве следующего. При этом оператор с меткой М может быть расположен в любом месте программы.

Оператор цикла

Оператор цикла служит для того, чтобы поручать машине выполнение некоторых операторов, входящих в «тело цикла» несколько раз.

Указание об этом делается так:

"ДЛЯ" X = A "ШАГ" K "ДО" M "ВЫПОЛНИТЬ" P

Буквой X обозначен параметр цикла, A — начальное его значение, K — шаг цикла или приращение параметра, M — конечное значение параметра.

Буквой P обозначен оператор или группа операторов, образующих так называемое «тело цикла». Этими операторами могут быть любые из применяемых в языке.

Пример. Дан оператор, работая по которому машина найдет сумму 100 первых натуральных чисел.

5. СУМ = 0; "ДЛЯ" X = 1 "ШАГ" 1 "ДО" 100
"ВЫПОЛНИТЬ" СУМ = СУМ + X

Действительно, сначала при X = 1 машине придется к СУМ = 0 прибавить X = 1 и после этого считать СУМ = = 1. Затем, выполняя оператор, машина увеличит X на единицу (на величину шага) и X станет равным двум. Полученное значение X = 2 машина прибавит к СУМ = 1 и получит СУМ = 3. Так будет продолжаться до тех пор, пока X не станет равным 100. При X = 100 в последний раз будет выполнен оператор, образующий в данном случае «тело цикла», а именно: СУМ = СУМ + X.

Еще один пример.

6. Ф = 1; "ДЛЯ" У = 1 "ШАГ" 1 "ДО" 100
"ВЫПОЛНИТЬ" Ф = Ф × У

Разберите пример самостоятельно и убедитесь, что в результате переменная Ф станет Ф!, а в данном случае Ф = 100!

В заключение приведем сведения об операторах, с помощью которых машине дается указание о печатании промежуточных и окончательных результатов вычислений, а также данных, хранящихся в ее памяти.

Операторы вывода

В языке ЭЦВМ «Мир» используется несколько операторов вывода, применяя которые программист может приказать машине отпечатать:

а) имя переменной и вычисленное или хранящееся в памяти машины ее числовое значение. Форма оператора:

"ВЫВОД" A_1, A_2, A_3, ..., A_n

б) только значения переменных без указания их имени. Форма оператора:

"ВЫВОД" "ЗНАЧЕНИЙ" A_1, A_2, ..., A_n

в) имя массива и полное его содержание строка за строкой. Форма оператора:

"ВЫВОД" "МАССИВА" М

г) таблицу значений одной или нескольких переменных, состоящую из нескольких строк и столбцов. Форма оператора:

"ВЫВОД" "ТАБЛИЦЫ" К, A_1 A_2, ..., A_n

Буквой К обозначен номер таблицы. Буквой А с индексами обозначены имена переменных. Все значения одной переменной в таблице располагаются в один столбец.

Мы рассмотрели только основные операторы языка ЭЦВМ «Мир». Располагая только ими, программист может составлять законченные программы для решения большого круга задач.

Подобно тому, как во всех рассмотренных выше алгоритмических системах мы специально подчеркивали, что алгоритм конструируется из его элементов (команд, подстановок) по специальным правилам, написание программы, располагая операторами, следует вести по строго очерченным законам.

Прежде всего заметим, что структура всякой программы, записанная на языке ЭЦВМ «Мир», всегда имеет одну из двух стандартных форм:

"РАЗРЯДНОСТЬ" К. P_1; P_2; P_3; ...; P_n "ГДЕ" Q_1; Q_2; Q_3; ...; Q_n "КОНЕЦ"

или

"РАЗРЯДНОСТЬ" К. P_1; P_2; P_3; ...; P_n "КОНЕЦ"

Буквой К обозначена величина разрядности, которая является показателем точности вычислений. Указывая разрядность, равную, например, 10, программист требует тем самым от машины производить все промежуточные и окончательное вычисления таким образом, чтобы во всяком результате имелось 10 цифр. Если вычисления производятся исключительно во множестве целых чисел, то разрядность указывается равной 1. P_i — операторы; Q_i — описатели. Роль описателя состоит в том, чтобы давать информацию машине о переменных, использованных в операторах.

Ниже приводится несколько примеров программ для решения несложных задач.

З а д а ч а 1. Вычислить и отпечатать под именем СУМ сумму 100 первых нечетных натуральных чисел.

Программа (вариант 1):

```
"РАЗРЯДНОСТЬ" 1.
    СУМ = 0; "ДЛЯ" X = 2 "ШАГ" 1 "ДО" 201
    "ВЫПОЛНИТЬ"
    СУМ=СУМ+X; «ВЫВОД» СУМ «КОНЕЦ»
```

Программа (вариант 2):

```
"РАЗРЯДНОСТЬ" 1.
    "ВЫВОД" СУМ "ГДЕ" СУМ = ∑(X = 1, 100, 2 ×
    × X — 1)
                                    "КОНЕЦ"
```

В данном случае была использована одна из специальных функций, приведенных выше. Результат работы машины в обоих случаях одинаков. На печать будет выдано число СУМ = 10100.

З а д а ч а 2. Необходимо составить таблицу значений двух функций $У = X^2$ и $У = 3X — 4$ для 5 первых натуральных значений аргумента. Таблицу требуется отпечатать.

Программа:

```
"РАЗРЯДНОСТЬ" 1.
    "ДЛЯ" X = 1 "ШАГ" 1 "ДО" 5 "ВЫПОЛНИТЬ"
    "ВЫВОД" "ТАБЛИЦЫ" 1, X, У1, У2
    "ГДЕ" У1 = X ↑ 2; У2 = 3 × X — 4 "КОНЕЦ"
```

В результате машина напечатает следующую таблицу:

X	Y1	Y2
1	1	—1
2	4	2
3	9	5
4	16	8
5	25	11

З а д а ч а 3. В массиве из 10 различных чисел необходимо найти самое большое и отпечатать его. Массив имеет имя М.

Программа:

Предложенная задача носит логический характер. Идея решения может быть такой. Обозначим разыскиваемое самое большое число в массиве через МАХ. Предположим, что таким числом является первый элемент массива М, а именно M_1. Переменной МАХ присвоим значение M_1. Это можно сделать оператором присваивания (МАХ = М [1]). После чего будем сравнивать все остальные числа один за другим с величиной переменной МАХ. Если какой-либо элемент массива М окажется больше чем МАХ, то значение МАХ изменим, присвоив ему значение того элемента, который оказался больше. Так будем продолжать до тех пор, пока не будут проверены все элементы массива М. После этого значение МАХ будет выдано на печать.

```
"РАЗРЯДНОСТЬ" 1.
  I = 1; MAX = M[1]; 5. I = I + 1; "ЕСЛИ" M[I] >
  > MAX "ТО"
  (MAX = M[I]; "НА" 5) "ИНАЧЕ" ("НА" 5);
  "ВЫВОД" MAX
  "ГДЕ" M[10] = ...                    "КОНЕЦ"
```

Для решения конкретной задачи при вводе программы в машину необходимо отпечатать все числа, образующие массив М.

Теперь вернемся к задаче о вычислении расстояний между двумя произвольными датами в днях. Оценим наши возможности. Мы знаем, как составляется программа из таких операторов, как оператор присваивания, оператор цикла, условный оператор, оператор перехода и вывода. Опираясь только на такие сведения, попробуем составить текст программы для ЭЦВМ «Мир».

Предлагается следующая идея решения задачи (рис 34). Обозначения на рисунке такие:

Г1 — начало года первого события (например, 1961)

М1 — начало месяца, в котором произошло первое событие (5)

Д1 — день события (17)

Г2, М2 и Д2 — эти же величины соответственно для второго события (1975, 7, 13).

Вычисления будем производить в несколько этапов. Сначала найдем расстояние Р1 — число дней, которые

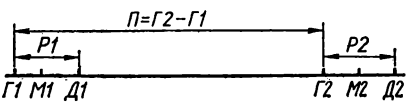

Рис. 34

отделяют первое событие от начала его года. Затем вычислим аналогично Р2 — число дней, отделяющих второе событие от начала его года. После этого определим, сколько дней содержит разность Г2—Г1. И лишь после этого можно будет вычислить искомое расстояние.

Каждый из этих этапов состоит из нескольких частей.

Прежде чем писать текст программы, напомним читателю, что среди операций, которые может выполнять машина, есть операция «Взятие целой части от числа» — обозначение Е (Х). Если Х = 4,3, то Е (4,3)= 4.

Предварительно занесем в память машины массив чисел М, каждый элемент массива М есть число дней в соответствующем месяце. Затем нам придется выяснить, является ли Г1 годом высокосным, чтобы знать, сколько дней в феврале этого года. Это сделать можно так. Г1 разделить на 4, взять от частного целую часть и ее умножить на 4. Если произведение окажется равным числу Г1, то год Г1 — высокосный. После этого Р1 найти уже нетрудно. Следует найти сумму дней во всех месяцах, предшествующих М1, и к результату прибавить Д1.

Аналогично находится и Р2.

Выяснение числа дней, которые отделяют начало Г1 от начала Г2, проводим в два этапа. Сначала мы выясним, сколько високосных лет находится в промежутке Г2—Г1. Для этого достаточно разделить Г2—Г1 на 4. После этого разность Г2 — Г1 в днях (П) выразится так:

$$П = (Г2 - Г1)\ 365 + В,$$

где В — число високосных лет в отрезке Г2 — Г1.

Наконец, находим искомое расстояние:

$$Р = П + Р2 - Р1 \text{ (в днях)}.$$

Текст программы приводим с краткими пояснениями:

"РАЗРЯДНОСТЬ" 10.

М[1] = М[3] = М[5] = М[7] = М[8] = М[10] =М[12] = 31;

М[4] = М[6] = М[9] = М[11] = 30;

"ЕСЛИ" Е(Г1/4) × 4 = Г1 "ТО" (М[2] = 29)

"ИНАЧЕ" (М[2] = 28;

(мы указали, сколько дней содержится в феврале Г1);

С1 = 0; "ДЛЯ" Х = 1 "ШАГ" 1 "ДО" М1 — 1

"ВЫПОЛНИТЬ" С1 = С1 + М[Х];

(вычислено количество дней во всех месяцах, предшествовавших М1);

$$Р1 = С1 + Д1;$$

(найдено общее количество дней, прошедших от начала Г1);

"ЕСЛИ" Е(Г 2/4) × 4 = Г2 "ТО" (М[2] = 29)

"ИНАЧЕ" (М[2] = 28);

(выяснено, сколько дней в феврале года Г2);

С2 = 0; "ДЛЯ" Х = 1 "ШАГ" 1 "ДО" М2 — 1

"ВЫПОЛНИТЬ" С2 = С2 + М[Х];

Р2 = С2 + Д2;

В = Е((Г2 — Г1)/4); П = (Г2 — Г1) × 365 + В;

(найдено число дней между началом Г1 и началом Г2);

Р = П + Р2 + Р1; "ВЫВОД" Р

"ГДЕ" Г1 = ...; М1 = ...; Д1 = ...;

Г2 = ...; М2 = ...; Д2 = ... "КОНЕЦ".

В тексте программы значения Г1, Г2, М1, М2, Д1, Д2 могут быть введены произвольными и поэтому предлагаемая программа решает задачу не для конкретного случая, а в самом общем виде. Текст программы на алгоритмическом языке ЭЦВМ «Мир» предстает как еще одна форма задания алгоритма.

Предлагаем читателю выступить «в роли» машины, которая умеет выполнять предписания, приведенные в тексте программы, и найти расстояние между датами:

Г1 = 1961	Г2 = 1973
М1 = 4	М2 = 9
Д1 = 12	Д2 = 27

Вычисленное расстояние покажет, сколько дней отделяет первый космический полет Ю. Гагарина от полета корабля «Союз-12».

ПРИЛОЖЕНИЕ

От абстрактной модели к действующей машине

Задача, которую мы намерены обсуждать и решать в этом разделе, есть задача инженерная. Доступно ли ее решение учащимся? Что даст рассказ о решении этой задачи читателю, интересующемуся математикой?

Ответ на первый из этих вопросов дала практика. Школьники из Малой Академии наук «Искатель» не только хорошо разбираются в том,

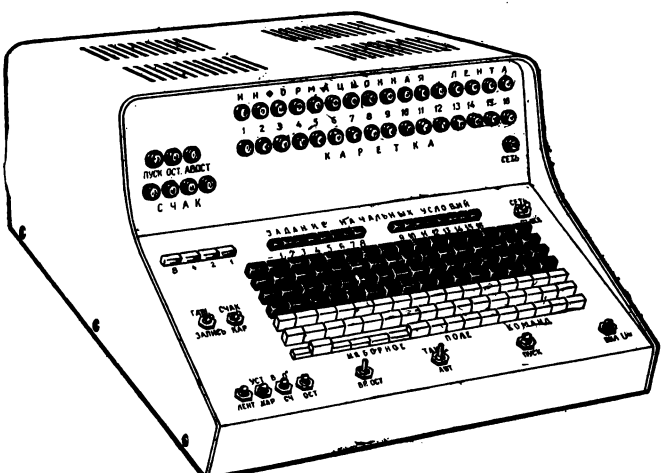

Рис. 35

как устроена действующая машина, как она работает но и сами нашли несколько усовершенствований для первого ее образца (рис. 35).

Отвечая на второй вопрос, следует сказать, что изучение конструкции машины позволит рассмотреть главные вопросы работы ЦВМ с программным управлением, понять в чем суть автоматической ее работы, разобраться в том, как текст программы, написанный программистом на бумаге, превращается в **рабочую программу**, которой руководствуется машина.

Познакомимся с тем, что принято называть **внутренним, или машинным, языком**, выясним, что такое **память ЦВМ**, ее назначение и использование.

Если учесть, что большинство идей о конструкции ЦВМ в свое время исходило от математиков, то небольшая экскурсия в инженерно-математическую область будет полезной.

Начнем с выяснения некоторых ограничений, которые нам придется ввести, задавая требования к действующей машине Поста.

Прежде всего следует ввести ограничение по «длине» информационной ленты. Ясно, что в действующей машине она не может быть бесконеч-

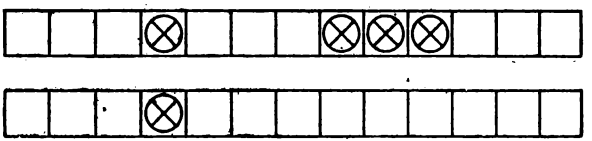

Рис. 36

ной. Практика показала, что для учебных и демонстрационных целей достаточно иметь ленту из 16 секций (ячеек). В каждую секцию машина может помещать метку или стирать ее. В роли **метки** действующей машины мы будем использовать горящую электрическую лампочку, расположенную внутри секции информационной ленты. Если лампочка зажжена, то секция, в которой она находится, считается отмеченной, в противном случае секция считается неотмеченной.

В роли **каретки** в действующей машине также будет использоваться горящая лампочка. Под каждой ячейкой информационной ленты мы располагаем ячейку с лампочкой. Если какая-либо из этих (ниже расположенных) лампочек горит, то это значит, что каретка расположена под данной ячейкой информационной ленты.

На рис. 36 показано, что каретка расположена под отмеченной, четвертой слева, ячейкой информационной ленты.

Еще раз подчеркнем, что в нижнем ряду всегда горит одна единственная лампочка, указывающая расположение каретки. Зажечь и потушить каждую лампочку можно, подавая сигнал 1 в сигнальную ячейку. Каждая ячейка информационной ленты и каждая лампочка, выступающая в роли каретки, снабжена **сигнальной ячейкой.** На рис. 37 показано, как лампочка соединена с сигнальной ячейкой (СЯ).

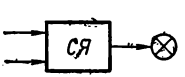

Рис. 37

Подача кратковременного сигнала 1 на верхний вход сигнальной ячейки вызывает загорание лампочки, которая будет гореть до тех пор, пока на нижний вход сигнальной ячейки не будет подан такой же кратковременный сигнал 1. В роли сигнала 1 будет выступать *подача напряжения в 5 вольт.*

Условимся об **алфавите машинного языка** разрабатываемой машины Алфавит будет включать всего две буквы: букву 1 и букву 0. Все необходимые **машинные слова** будем составлять только из этих букв.

Для чего же нам потребуются эти машинные слова, как эти слова будут использоваться машиной? Представьте себя в роли исполнителя программы для машины Поста. Что Вам будет необходимо прежде всего? Ясно, что прежде всего следует «видеть» или «держать в уме» текст программы, по которой придется работать. Действующая машина также должна руководствоваться программой. Текст программы в этом случае мы должны ввести в память машины, каждая команда должна быть записана на машинном языке. Номер команды мы будем выражать числом двоичной системы счисления, а для конкретных указаний введем такие машинные слова (табл. 4).

Таблица 4

Обычное обозначение	Машинное слово
→	011
←	100
∨	001
↕	010
!	101
? <	110

Приводим несколько команд и их запись на машинном языке:

1.←5	0001	100	0101
	1	→	5
7.∨13	0111	001	1101
	7	∨	13

Каждая команда, записанная на машинном языке, помещается в пассивное запоминающее устройство (ПЗУ). В машине, которую мы разрабатываем, ПЗУ будет состоять из 16 **ячеек памяти**. В каждую ячейку ПЗУ можно поместить только одну команду. Это дает еще одно, преду-

сматриваемое нами, ограничение — на нашей машине можно будет решать задачи программы, решения которых содержат не более 16 команд.

Еще одно замечание по форме записи команды на машинном языке. Мы не будем запоминать номер исполняемой команды, а будем помещать в ПЗУ только две части команды: слово, обозначающее суть операции (код операции) и адрес (номер той команды, которая будет исполняться вслед за данной). Для записи такой команды достаточно 7 букв: три буквы — для кода операции (КОП) и четыре — для адреса следующей команды (КА). Мы помним, что адрес не может быть больше, чем 16_{10} или 10000_2

Детально конструкцию ПЗУ мы опишем ниже, а сейчас рассмотрим так называемые **регистры** — регистр кода адреса и регистр кода операции.

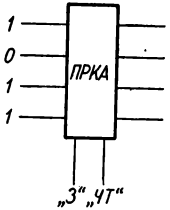

Рис. 38

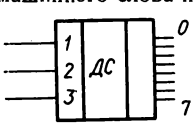

Рис. 39

Регистр — это устройство, предназначенное для запоминания подведенного к нему машинного слова и для демонстрации этого слова. В случае необходимости запоминание, демонстрация и стирание машинного слова производится «по разрешению».

Схематически регистр можно изобразить так, как показано на рис. 38. Здесь изображен регистр кода адреса. На выходы регистра извне подается машинное слово 1011, буквой З обозначен вход для сигнала, разрешающего «запись в регистр». Буквами ЧТ обозначен вход для сигнала, разрешающего «чтение» машинного слова, находящегося в регистре.

Подав кратковременный сигнал 1 (сигнал, разрешающий запись) на вход З, мы введем слово 1011 в регистр, где оно будет храниться до тех пор, пока мы его сами не сотрем. Подавая сигнал 1 на вход ЧТ, мы можем передавать слово далее в другие узлы машины.

В разрабатываемой нами машине два регистра: приемный регистр кода адреса (ПРКА) и приемный регистр кода операции (ПРКОП).

Остановимся еще на устройстве дешифратора, который входит в число узлов машины. Дешифратор условно можно изобразить так, как показано на рис. 39. На входы дешифратора подаются двоичные трехбуквенные слова. Каждому слову дешифратор ставит в соответствие один из 8 сигналов на его выходах. Например, если на вход подается слово 110, то сигнал 1 появится только на выходе, обозначенном цифрой 6. Конечно, дешифраторы могут иметь не только три, но и другое число входов. Принцип же их работы один: двоичному слову на входе каждый раз ставится в соответствие один и тот же сигнал на выходе. В нашей машине будет использован трехвходовый дешифратор для дешифрации кода операции и четырехвходовый для дешифрации кода адреса.

Важным узлом создаваемой машины является **счетчик**. Схематически изобразить счетчик можно так, как показано на рис. 40. Счетчик имеет четыре группы входов. Входы параллельного занесения служат для того, чтобы поместить в счетчик любое четырехразрядное двоичное число Подача сигнала 1 на вход, обозначенный нами как «Вх + 1», вызывает увеличение любого двоичного числа, находящегося в счетчике, на единицу. Подача такого же сигнала 1 на вход «Вх — 1» вызывает уменьшение хранящегося в счетчике двоичного числа на единицу. Наконец подача стандартного сигнала 1 на вход «Уст. «0» приведет к стиранию числа в счетчике, счетчик после этого будет хранить и демонстрировать на своих выходах число 0000_2.

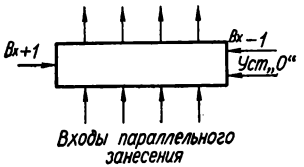

Рис. 40

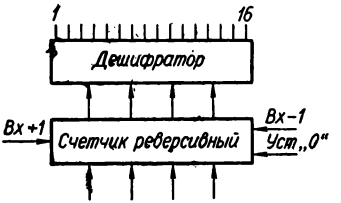

Рис. 41

Таким образом, устроенный счетчик принято называть **реверсивным двоичным счетчиком.** Если выходы реверсивного двоичного счетчика (а этих выходов четыре) присоединить к входам четырехвходового дешифратора, то мы получим так называемый одноразрядный **реверсивный сдвиговый регистр.** Схематическое его изображение дано на рис. 41.

Перечислим еще раз узлы, из которых будет собираться действующая машина.

1. Информационная лента на 16 секций.
2. Каретка.
3. Приемный регистр кода адреса.
4. Приемный регистр кода операции.
5. Реверсивный счетчик.
6. Реверсивный сдвиговый регистр.
7. Дешифратор кода операции.
8. Пассивное запоминающее устройство.

О других узлах бедет сказано ниже.

Дальнейший рассказ будем вести, используя блок-схему действующей машины. Описание будет вестись языком, который принят в инженерных документах. Начинаем с рассмотрения структуры машины.

На рис. 42 дана структурная схема действующей машины Поста. Машина имеет в своем составе пассивное запоминающее устройство (1) для хранения программ, выходные шины которого связаны с регистром (2) для приема и хранения кода операций и с регистром (3) для приема

и хранения кода адреса. Выходы регистра (2) связаны с дешифратором кода операций (4) Выходы дешифратора (4) связаны с реверсивным сдвиговым регистром (5), который осуществляет управление движением каретки. Два других выхода дешифратора (4) связаны с информационной лентой (7), по этим выходам приходят сигналы на зажигание и тушение лампочки в секциях информационной ленты. Один из выходов дешифратора служит для передачи команды «Стоп» в схему управления

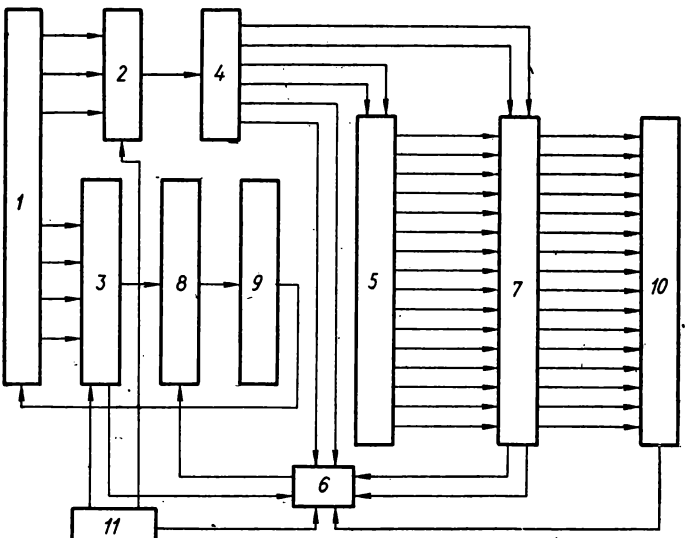

Рис. 42

(6), и последний выход дешифратора (4) служит для передачи команды «?» в устройство управления (6).

Выходы приемного регистра кода адреса (3) связаны со схемой управления (6) для выяснения, имеется ли в регистре (3) какой-либо номер, отличный от 0000. Выходы регистра (3) связаны также со счетчиком (8), который служит для формирования адреса команды, которую машине придется выполнять в следующем цикле. Выходы счетчика (8) являются входами дешифратора (9), который в свою очередь связан с пассивным запоминающим устройством (1).

Выходы реверсивного сдвигового регистра (5) связаны и с информационной лентой (7), которая является в свою очередь 16-разрядным регист-

ром, на выходах которого стоят сигнальные ячейки с лампочками. Каждая секция информационной ленты (7) имеет выход, подаваемый в устройство контроля (10). Устройство контроля служит для того, чтобы схема управления (6) могла в нужный момент узнать, отмечена та или иная секция или нет.

Машина содержит устройство управления (11), в котором вырабатываются сигналы рабочего цикла. Устройство управления связано с регистрами (2) и (3), а также со схемой управления (6).

После ознакомления с тем, из каких узлов состоит машина, рассмотрим ее **рабочий цикл**. Сначала опишем работу машины на словах, а затем изобразим рабочий цикл в виде граф-схемы.

Работа на машине начинается с того, что с пульта управления в пассивное запоминающее устройство вводится программа решения задачи. На информационную ленту заносится исходная информация (дело сводится к тому, что в необходимых секциях информационной ленты вручную зажигаются лампочки). Каретка помещается в нужное место (зажигается лампочка под выбранной секцией информационной ленты). В счетчик с пульта заносится номер команды, которую машина будет выполнять первой. На этом подготовка машины к работе завершается. Вся остальная работа машины после ее пуска будет протекать без нашего вмешательства и закончится либо по команде «Стоп», предусмотренной в программе, либо по команде «Авост» — аварийная остановка. Аварийная остановка происходит по ошибке программиста: поручил машине поместить уже отмеченную секцию или стереть метку, а в обозреваемой секции метки нет. Машина может работать и без остановки, что может быть специально запрограммировано или вызвано ошибкой программиста.

На рис. 35 изображен пульт и демонстрационное поле машины. Слева показаны четыре лампочки. Они подключены к выходам счетчика и по ним можно следить за тем, какая по номеру команда выполняется.

Наступил момент осуществить «Пуск» — нажимаем кнопку «Пуск», и машина автоматически начинает выполнять один свой рабочий цикл за другим. Каждый рабочий цикл состоит из 7 микротактов. За один рабочий цикл машина успевает извлечь из ПЗУ команду, которую ей предстоит выполнить, выполняет ее и подготавливает к выполнению следующую.

Даем описание одного рабочего цикла действующей машины Поста.

После пуска машины из пассивного запоминающего устройства (1) (рис. 42) извлекается команда, которая поступает на регистры (2) и (3) (мы помним что из семи букв одной команды три используются для обозначения кода операции, а четыре — для кода адреса). Регистр (2) служит для приема кода операции — на одном из шести выходов дешифратора (4) появляется сигнал о том, какая операция была записана в ячейке ПЗУ.

Еще раз обратимся к изображению дешифратора (4) (рис. 43).

Над каждым выходом указаны команды, выходы пронумерованы сверху вниз.

Если сигнал появляется на первом выходе дешифратора (4) (рис. 43), то машина зажжет лампочку в той секции информационной ленты, под которой в данный момент горит лампочка каретки. Если сигнал появляется на втором выходе дешифратора (4), то лампочка в обозреваемой секции будет потушена. При этом, если в обозреваемой секции информационной ленты (7) до появления сигнала из дешифратора (4) была метка, то после появления сигнала из дешифратора (4) (с первого его выхода), устройство которого (10) выдает в схему управления (6) сигнал «Авост», будет сигнал остановки по ошибке в программе. Действительно, машина не может зажечь уже горящую лампочку и поэтому останавливается. То же самое произойдет, если обозреваемая секция пуста, лампочка, помещенная в ней, не горит и проходит сигнал со второго выхода дешифратора (4). При появлении сигнала I на третьем выходе дешифратора будет занесена единица на вход «Вх—1» реверсивного счетчика и каретка сдвинется вправо. Аналогично, если сигнал появится на четвертом выходе дешифратора (4). Если сигнал появится на пятом выходе дешифратора (4), то произойдет остановка машины, предусмотренная программистом.

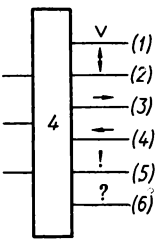

Рис. 43

Вернемся к началу. Рассмотрим работу машины в случае если выполняемая команда — одна из первых четырех (→, ←, ∨, ↕). После выполнения любой из этих команд работа машины протекает одинаковым образом: схемой управления (6) осуществляется контроль состояния приемного регистра кода адреса (3). Если в регистре (3) хранится код 0000, то в счетчик (8) из схемы управления (6) поступает сигнал на вход «Вх+1» и адрес, хранящийся в счетчике (8), увеличивается на единицу. Если же в регистре (3) хранится любое другое число, то счетчик (8) устанавливается в нуль и в него переписывается содержимое регистра (3), после чего цикл работы может быть повторен.

Отдельно рассмотрим случай, когда сигнал появляется на шестом выходе дешифратора (4) — дешифратор сообщает, что в качестве выполняемой в этом цикле команды выступает команда передачи управления (УП):

Работа протекает так. Если обозреваемая секция информационной ленты (7) отмечена, то по сигналу об этом из устройства контроля (10) схема управления (6) устанавливает счетчик (8) в нуль, и в него перепи-

сывается содержимое регистра (3). Если же обозреваемая секция информационной ленты (7) не отмечена, то в счетчик (8) из схемы управления (6) поступает сигнал «+1» и цикл работы устройства может быть повторен.

Последний абзац требует небольшого разъяснения. Команда передачи управления имеет не одну, а две отсылки. Если секция пуста, то в

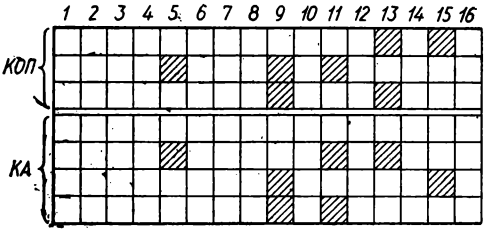

Рис. 44

качестве следующей нужно выполнять одну команду, а если отмечена — другую.

Мы уже говорили, что в ячейку ПЗУ можно поместить код операции и адрес той команды, которая будет выполняться в качестве следующей. На рис. 44 показаны две команды, хранящиеся в 5-й и 9-й ячейках ПЗУ. В пятой ячейке хранится команда ↕ (стереть метку), а в девятой ячейке — команда → (сдвиг вправо).

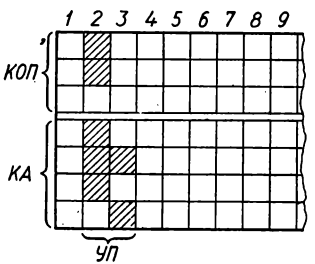

Рис. 45

Если клетка заштрихована, то это значит, что в ней записана единица. Прочтите, какие команды записаны в ячейках 11, 13 и 15.

Ясно, что команду передачи управления («?») в одну ячейку не поместить. Мы поступим так: под эту команду будем отводить две ячейки и располагать команду так, как показано на рис. 45. На рис. 45 показано, как в ПЗУ заносится команда 2.?$\begin{matrix}5\\14\end{matrix}$

Ясно теперь, что если секция отмечена, то в соответствии с описанием цикла в качестве следующего в приемный регистр следует заносить код адреса. Для этого нужно предварительно очистить счетчик (8). Если обозреваемая секция не отмечена, то в соответствии с описанием цикла, нужно в счетчик (8) заносить адрес 5, который хранится не во 2-й, а в 3-й ячейке. Вот почему в счетчике (8) прибавляется +1, и лишь после этого

из ячейки 3 будет переписан хранящийся в ней код адреса и цикл будет повторен.

Все эти описания можно представить в виде граф-схемы (рис. 46). Как и в других граф-схемах, в прямоугольниках выписаны операторы,

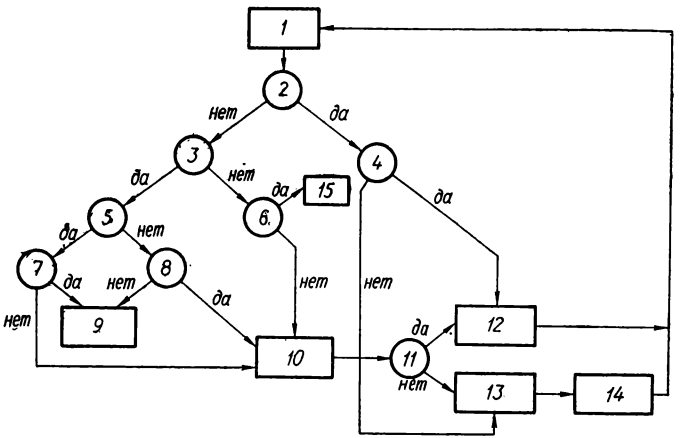

Рис. 46

а в кружочках — распознаватели. Даем описание операторов и распознавателей, используемых в граф-схеме:

1 — прием команды из пассивного запоминающего устройства в регистры (2) и (3);

(2) — выяснение, является ли извлеченная из ПЗУ команда командой передачи управления («?»);

(3) — выяснение, являеся ли команда командой «пометить секцию» (∨) или «стереть метку» (↕);

(4), (7), (8) — выяснение, является ли обозреваемая секция отмеченной;

(5) — выяснение, является ли команда командой «пометить секцию»(∨);

(6) — выяснение, является ли команда командой «Стоп»;

9 — аварийная остановка машины;

10 — выполнение команды;

11 — выяснение, является ли код, хранящийся в регистре адреса (3), числом 0000;

12 — прибавление единицы к числу, находящемуся в счетчике (8);

13 — установка счетчика (8) в нуль;

14 — запись в счетчик (8) содержимого регистра адреса (3);

15 — остановка машины по команде.

ОТВЕТЫ И РЕШЕНИЯ

Рассказ-задача 1

КАК РАСКРЫВАЮТ ТАЙНЫ ЧЕРНЫХ ЯЩИКОВ

1. Чтобы обеспечить непрерывное звучание Пения, необходимо
а) применить в состоянии a_1 воздействие b_4 (в течение минуты играть на Органе при зажженном Ладане). Это приведет к переходу в состояние a_3 (звучит только Пение);
б) затем применить воздействие b_1 (прекратить игру на Органе и сжигание Ладана). Это обеспечит устойчивость состояния a_3.

2. Чтобы обеспечить непрерывное звучание Смеха, необходимо:
а) в состоянии a_1 применить воздействие b_1 (в течение минуты не играть на Органе и не жечь Ладан). Это приведет к переходу в состояние a_2);
б) затем применить одно из двух воздействий: b_1 или b_3 (любое на выбор).

Рассказ-задача 2

АВТОМАТ СЛАВЫ СТРЕЛЬЦОВА

Алгоритм беспроигрышной игры, приводимый Б. А. Кордемским в книге «Математическая смекалка» (Москва, 1958, с. 525).

На столе 25 предметов (спичек). Ход человека— Вашего противника. Далее текст из книги:

«Если у противника **четное** число спичек, то надо оставить ему такое количество спичек, которое на 1 больше кратного шести (19, 13 или 7); если у противника **нечетное** число спичек, то надо оставить ему такое количество спичек, которое на 1 меньше кратного шести (23, 17, 11, 5), а если это окажется невозможным, то оставить ему количество спичек, кратное шести (24, 18, 12, 6)».

Сравните этот текст алгоритма со схемой, приведенной в задаче, и Вы обнаружите, насколько та форма представления алгоритма проще для использования и более обозрима.

АВТОМАТ НАВОДИТ ПОРЯДОК

Разыскиваемая граф-схема имеет вид, изображенный на рис. 47.

Опишем работу схемы на примере. Пусть справа к автомату приближаются детали A и B, расположенные в таком порядке:

$$AABBBA.$$

Как будет действовать автомат?

Так как исходное состояние автомата a_0, то обнаружив, что первой к нему подошла деталь A, автомат ее пропустит дальше, а сам перейдет в состояние a_1 (смотри граф-схему). После смены состояния (a_0 на a_1) автомат вновь обнаруживает, что к нему подошла деталь A. B соответствии с алгоритмом, он ее сбрасывает и остается в том же состоянии a_1.

Следующей к нему подходит деталь B, автомат ее пропускает дальше, а сам переходит из состояния a_1 в состояние a_2.

Находясь уже в состоянии a_2, он обнаруживает две подряд подошедших детали B и обе сбрасывает. Когда же перед ним окажется деталь A, он ее пропустит, а сам перейдет в состояние a_0 и работа может быть продолжена.

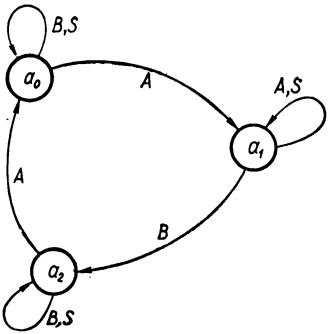

Рис. 47

Рассказ-задача 4

ЗАДАЧА ПРОФЕССОРА В. УСПЕНСКОГО

Идея решения задачи состоит в том, чтобы организовать раскачивающийся маятник. Маятник организуется с помощью двух меток, которые поочередно сдвигаются шаг за шагом — одна вправо, другая влево. Рано или поздно одна из них «наткнется» на массив, после чего к найденному массиву присоединяется данная метка, а второе «плечо» маятника (метка) стирается.

Рекомендуем проверить работу программы на простом примере.

Текст программы:

1. ?2 3
2. ← 4
3. → 4
4. ?5 3
5. ∨ 6
6. ← 7
7. ?8 15
8. → 9
9. ?10 8
10. ∨ 11
11. → 12
12. ?13 19
13. ← 14
14. ?5 13
15. → 16
16. → 17
17. ?23 18
18. ↕ 16
19. ← 20
20. ← 21
21. ?23 22
22. ↕ 20
23. !

Завершим описание решения следующими словами В. Успенского: «Автору не удалось построить более короткую программу, которая служила бы решением той же задачи. В то же время автор не знает, как доказать, что найденная программа самая короткая из возможных. Может быть кому-нибудь из читателей удастся сделать то или другое» [1].

Приведенная задача предлагалась для решения участникам областной олимпиады по кибернетике в Крыму и была решена большинством участников.

Рассказ-задача 5

ЗАДАЧА О МАШИНЕ ТЬЮРИНГА

Изобразим исходные данные (рис. 48). Слово на ленте взято только для примера.

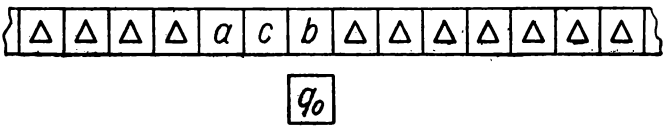

Рис. 48

Функциональная схема имеет вид (табл. 5).

Таблица 5

	q_0	q_1	q_2	q_3	q_4	q_5	q_6
a	$Лq_1$		$Пq_5$	$Пq_4$	$СЛq_7$	$ВЛq_7$	
b	$Лq_2$	$Пq_4$		$Пq_6$	$СЛq_7$		$аЛq_7$
c	$Лq_3$	$Пq_5$	$Пq_6$			$ВЛq_7$	$аЛq_7$
Δ							

[1] Успенский В. А Как работает машина Поста,— «Математика в школе», 1967, № 1—4.

Продолжение таблицы 5

	q_7	q_8	q_9	q_{10}	q_{11}	q_{12}	q_{13}	q_{14}
a	*Л*	Πq_9		$Лq_{13}$	$Лq_{12}$	*aЛ*!	*b*!	
b	*Л*	Πq_{10}	$Лq_{12}$	$Лq_{14}$	$Лq_{14}$	*aЛ*!		*c*!
c	*Л*	Πq_{11}	$Лq_{13}$				*b*!	*c*!
Δ	Πq_8							

Данное решение принадлежит учащемуся 6-го классса Саше Мебель (г. Симферополь, ср. шк. № 40).

Рассказ-задача 6

ЗАДАЧА О НАХОЖДЕНИИ НАИМЕНЬШЕГО ЧИСЛА

Необходимо построить нормальный алгоритм. Идею решения предварительно обсудим, рассматривая пример.

Пусть дано троичное число

$$12012_3.$$

Как следует поступать, чтобы образовать из цифр, входящих в число, наименьшее?

Ясно, что цифра старшего разряда должна быть наименьшей из имеющихся (однако этой цифрой не может быть 0). Ясно, что наименьшее число должно начинаться цифрой 1. Нуль, если он есть в данном случае, следует поместить сразу же за первой цифрой. Все остальные цифры должны располагаться в порядке убывания.

Нормальный алгоритм, построенный на такой идее, может быть таким:

$$10 \to 01$$
$$20 \to 02$$
$$21 \to 12$$
$$01 \to .\ 10$$
$$02 \to 20$$

Рассказ-задача 7

ЗАДАЧА О ФОРМАЛЬНОЙ ЗАМЕНЕ ТЕКСТА ПРОГРАММЫ ДЛЯ МАШИНЫ ПОСТА ФУНКЦИОНАЛЬНОЙ СХЕМОЙ МАШИНЫ ТЬЮРИНГА

Располагая текстом программы, нужно вычертить бланк функциональной схемы. Строк в этой схеме две (это ясно, ибо алфавит машины Поста содержит только две буквы), число столбцов должно быть равно числу команд в тексте программы. Каждой команде машин Поста мы ставим в соответствие одну или две команды Тьюринга. Детали будут понятны из примера.

Дана программа для машины Поста

1. ↕ 2
2. → 3
3. ? 2 / 4
4. ← 5
5. V 6
6. !

Работая по этой программе, машина Поста присоединит метку к массиву, расположенному вправо от нее на конечном расстоянии (рис. 49).

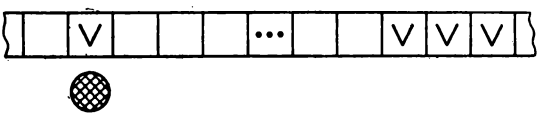

Рис. 49

Исходное состояние машины Тьюринга, соответствующее исходному состоянию машины Поста, показано на рис. 50.

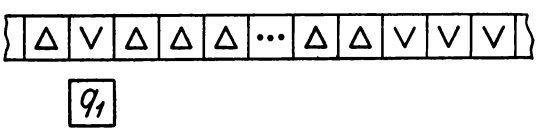

Рис. 50

Бланк для записи функциональной схемы имеет вид (табл. 6).

Заполнение начинаем с состояния q_1. Окончательный вид схемы приведен в табл. 7.

Читатель, наверное, обратил внимание на то, что каждая команда для машины Тьюринга в нашей схеме содержит не три, а только два указания, из которых одно обязательно есть указание о смене состояния.

Таблица 6

	q_1	q_2	q_3	q_4	q_5	
$\vee$						
Δ						

Таблица 7

	q_1	q_2	q_3	q_4	q_5	q_6
$\vee$	Δq_2	Πq_3	q_4	$Лq_5$		1
$\wedge$		Πq_3	q_2	$Лq_5$	$\vee q_6$	1

Возможно, что есть какой-нибудь иной формальный способ «перевода» с языка Поста на язык Тьюринга. Если попытки читателей в поисках такого алгоритма увенчаются успехом, то мы с интересом познакомимся с таким алгоритмом.

ЛИТЕРАТУРА

Глушков В. М. Введение в кибернетику. К., Изд-во АН УССР, 1964.

Ершов А. П. и др. Алгоритмические языки и программирование.— В кн.: История отечественной математики. К., «Наукова думка», 1970.

Ефремов Г. О. Алгоритмы. М., «Знание», 1964.

Касаткин В. Н., Верлань А. Ф. Секреты кибернетики. К., «Радянська школа», 1971.

Колмогоров А. Н. Алгоритм. БСЭ, т. 2. Изд. 2. М., с. 65.

Марков А. А. Теория алгоритмов.— Труды Математического института им. Стеклова. АН СССР, 1954, т. 42.

Росс Эшби У. Введение в кибернетику. М., Изд-во иностр. лит., 1959.

Трахтенброт Б. А. Алгоритмы и машинное решение задач. М., Физматгиз, 1973.

Успенский В. А. Как работает машина Поста. Цикл статей.— «Математика в школе», 1967, № 1—4.

ОГЛАВЛЕНИЕ