В·Н·Касаткин

ЛОГИЧЕСКОЕ ПРОГРАММИРОВАНИЕ В ЗАНИМАТЕЛЬНЫХ ЗАДАЧАХ

Как ЭВМ
решает
вычислительные
задачи?
Как удается
вычислительной
машине
принимать логические
решения?
О том,
как это делается,
разъяснено
на примерах
в предлагаемой
читателям
книге.

Киев
·Техніка·
1980

ББК 32.973
6Ф7.3
К28

Касаткин В. Н.

К28 Логическое программирование в занимательных задачах.— К.: Техніка, 1980.— 79 с., ил.

В пер.: 1 р. 10 к. 26 000 экз.

В книге в популярной и занимательной форме рассказывается о применении ЭВМ для решения некоторых невычислительных задач. Показано, как вычислительные машины выступают соперником человека в играх, как они решают логические задачи, как с их помощью моделируется распознавание простейших образов, обсуждается использование ЭВМ в исследовании некоторого класса искусственных эволюций.

Рассчитана на широкий круг читателей.

К $\frac{30502\text{-}191}{M202(04)\text{-}80}$ 151.80. 2405000000

ББК 32.973
6Ф7.3

Рецензенты д-р экон. наук *А. В. Крушевский*,
д-р физ.-мат. наук *Н. В. Яровицкий*

Редакция литературы по энергетике,
электронике, кибернетике и связи
Зав. редакцией *З. В. Божко*

ПРЕДИСЛОВИЕ

В предлагаемой читателю книге ведется рассказ о том, как готовятся задачи к их решению на электронных вычислительных машинах (ЭВМ) с программным управлением. Читатель приглашается к рабочему столу программиста, ему рассказываются в деталях конкретные особенности подготовки задач к машинному решению. В качестве задач-примеров выбраны задачи с занимательным сюжетом, для решения которых нужна смекалка.

Выбранные задачи можно условно назвать логическими, так как для их решения требуются не столько вычисления, сколько логические рассуждения. Такие задачи встречаются очень часто. Так, играя в домино, шахматы или шашки, мы, «рассчитывая» свой ход, к арифметике не обращаемся, не вычисляем, а рассуждаем.

Для решения таких задач на ЭВМ программист должен уметь использовать все возможности машины, т. е. обращаться не только к удивительным вычислительным способностям ее, но и привлекать на помощь умение ЭВМ быстро перебирать варианты, пользоваться большими вспомогательными таблицами, хранящимися в ее памяти, переименовывать используемые величины, сортировать их и т. д. От умения использовать столь разнообразный набор машинных средств зависит стиль и красота получаемых алгоритмов и решений.

Наряду с занимательностью сюжета, в каждой задаче подчеркиваются и необычные, занимательные особенности ее решения. При анализе предлагаемых в книге задач, сначала рассматривается возможность построения алгоритма ее решения в форме, удобной для использования человеком «вручную», т. е. без применения им какой-либо вспомогательной техники. После этого ведется конструирование алгоритма, который затем ставится в основу для написания текста программы. Этот предмашинный алгоритм зачастую значительно отличается от того, каким пользуется человек.

Для читателя разобраться в решении каждой задачи — это, во-первых, понять и научиться применять алгоритм, заданный в форме «для человека», и, во-вторых, изучить схемы алгоритмов «для машины». Схемы, приведенные в книге,

изображены в форме, близкой к стандартной, незначительное отступление от стандарта сделано умышленно, чтобы облегчить работу читателям. Читатели, знакомые с алгоритмическим языком АЛМИР-65, могут познакомиться с текстами рабочих программ, которые приведены в приложении вместе с инструкциями по их использованию на ЭВМ серии МИР и СМ ЭВМ.

Для понимания существа всех рассматриваемых в книге вопросов достаточно математической подготовки в объеме средней школы, но работа с книгой требует от читателя внимательности и настойчивости, качеств, столь характерных для разработчика алгоритмов и программ. Наградой за труд будет расширение представлений о возможностях применения ЭВМ и некоторых тонкостях в работе по подготовке задач к их решению на ЭВМ.

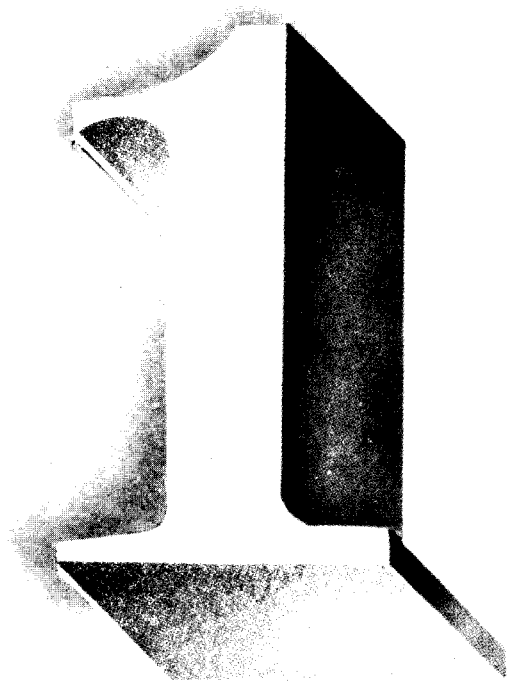
Разработка текстов программ осуществлена в лаборатории педагогической кибернетики Симферопольского госуниверситета. Автор благодарит сотрудников лаборатории Л. И. Владыкину и Т. П. Гнатенко, а также С. С. Нешева за большой труд по подготовке и отладке всех программ.

Отзывы и пожелания по книге просим направлять по адресу: 252601, Киев, 1, ГСП, Крещатик, 5, издательство «Техніка».

ЭВМ

СОПЕРНИК ЧЕЛОВЕКА В ИГРЕ

Глава



Игра
•Гонки
по вертикали•

Игра
•Тригекс•

Игра
•НИМ•
теория
и
алгоритм

Глава *1*

Машина выступает соперником человека в играх. Она «обдумывает» сложившиеся ситуации в ходе игры, «принимает» решения о том, какой очередной ход делать. Проводя поединок, машина руководствуется стратегией, основные положения которой хранит в своей памяти.

Если соперников разместить в разных комнатах и ход машины сообщать человеку с помощью записки, напечатанной, например, на машинке, то человек может до конца игры и не догадаться, что его соперником была ЭВМ, настолько действия его партнера по поединку были «разумными и логичными». ЭВМ ловко избегала ловушек и сама умела подстраивать замысловатые козни.

На чем же основана эта столь впечатляющая разумность машинного поведения? Конечно, на заранее разработанной программе ее поведения в игре конкретного класса.

Ниже, на примерах моделирования процесса трех игр ЭВМ с человеком, раскрываются некоторые особенности конструирования игровых программ.

ИГРА «ГОНКИ ПО ВЕРТИКАЛИ»

Условия игры. Полем для игры является вертикальная полоска, разделенная на N одинаковых клеточек. В игре участвуют двое, ходят по очереди. Сделать ход — это значит продвинуть шашку, стоящую к началу игры на клетке с нулевым номером, вверх на p клеток ($1 \leqslant p \leqslant M$, $M < N$). Очередной ход не должен повторять предшествующий ход соперника.

На рис. 1 показано два хода: первый — на три клетки вверх, второй — на две. Победителем считается тот из играющих, который сумеет первым достигнуть клетки с номером N или принудит соперника перешагнуть ее.

Задача состоит в том, чтобы найти секрет беспроигрышного участия в такой игре. Это значит перед началом игры уметь ответить на два вопроса:

каким по очереди, первым или вторым, вступать в игру?

Ведь это зависит от того, какие числа взяты в качестве значений N и M;

как следует рассчитывать свой ход после хода противника?

О том из участников игры, который знает ответы на оба вопроса, говорят, что он владеет беспроигрышной стратегией в игре «Гонки по вертикали».

Поиск беспроигрышной стратегии будем осуществлять необычно: на время отложим рассмотрение исходной игры и обратимся к другой, вспомогательной игре. Изучив особенности этой игры, находим секрет ее беспроигрышного проведения. Затем постараемся установить глубокую аналогию между вспомогательной игрой и игрой «Гонки по вертикали», что позволит нам сформулировать искомую беспроигрышную стратегию в рассматриваемой игре. Опишем вспомогательную игру.

Игровое поле для этой игры представляет собой решетку из M вертикальных и N горизонтальных линий.

Все линии перенумерованы так, как показано на рис. 2. Играют двое, ходят по очереди. Сделать первый ход — это значит провести из самой левой нижней точки игрового поля отрезок, например так, как показано на рис. 3 (сплошная линия). Штрихами показано, как можно сделать следующий ход. Делая ход за ходом, участники игры вычерчивают на игровом поле ломаную линию, при этом она не должна иметь вертикальных звеньев и, что очень важно, номер столбца, на котором заканчивается отрезок, вычерчиваемый при выполняемом ходе, показывает, на сколько строк вверх продвинулся соперник за этот ход.

Обратимся еще раз к рис. 3. Здесь, выполняя ход, игрок поднялся на третью строку (его отрезок завершен на третьем столбце), выполнив второй ход, его соперник поднялся еще на две строки и поэтому конец вычерченного им отрезка находится на втором столбце. Если теперь начинавший игру участник пожелает подняться на одну строку, то он должен провести свой отрезок в точку, отмеченную крестиком. Из описания игрового поля ясно, что больше, чем на M строк вверх за один ход продвигаться нельзя.

Победителем в игре считается тот из участников, который сумеет первым завершить ход на горизонтали N или заставит соперника перешагнуть ее.

Приступая к отысканию беспроигрышной стратегии, вводим понятие «особая точка» игрового поля. Особой называется точка игрового поля, для которой всякий проведенный из нее по правилам игры отрезок ведет к проигрышу за конечное число ходов. Основное свойство особых точек — никаким одним правильным ходом из одной точки нельзя попасть в другую особую точку. Особые точки, отмеченные на игровом поле, для случая $N=13$ и $M=4$ показаны на рис. 4, особые точки выделены кружками, а неособые —

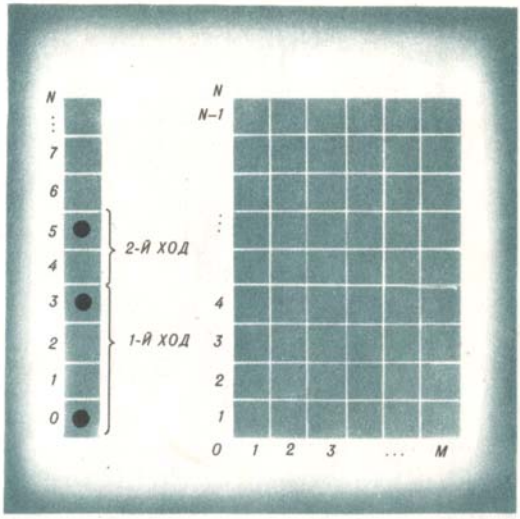


Рис. 1 Рис. 2

Рис. 3 Рис. 4

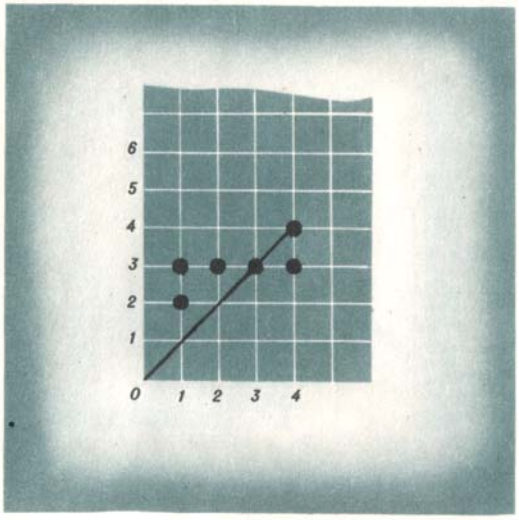


Рис. 5

крестиками. Еще раз подметим, что из рис. 4 видно, что одним ходом из любой особой точки в другую попасть нельзя, а за два хода всегда можно. Из сказанного следует стратегия игры:

заполните копию игрового поля особыми точками;

посмотрите, есть ли возможность первым ходом попасть в особую точку, и если есть, то вступайте в игру первым. Если нет, то «уступайте» право первого хода вашему сопернику.

Ясно, что, действуя таким образом, вы не дадите своему сопернику возможности попасть в особую точку и рано или поздно он проиграет.

Рассмотрим пример. Пусть $N=13$ и $M=4$, тогда в игру следует вступать первым и ходить либо до третьей, либо до четвертой вертикали. На рис. 5 показаны два возможных первых хода в игре.

Итак, беспроигрышная стратегия во вспомогательной игре найдена. Человек, решивший сыграть партию, знает, как ответить на вопрос, каким по очереди вступать в игру, и знает, как сделать свой очередной ход. Для этого, конечно, он должен уметь изготовлять себе «шпаргалку» — копию игрового поля с отмеченными особыми точками. Как это делать — будет рассказано ниже, а сейчас займемся сопоставлением только что разработанной вспомогательной игры с игрой «Гонки по вертикали». Отыщем аналогию между этими играми.

Из приведенного описания игр видно, что хотя внешне эти игры различные, ведь они проводятся на совсем непохожих игровых полях, у них много общего. Во-первых, каждой позиции, возникающей в игре «Гонки по вертикали», можно всегда поставить в соответствие одну единственную позицию во вспомогательной игре. Во-вторых, каждому ходу в одной игре соответствует единственный ход в другой и наоборот. В-третьих, конец игры обнаруживается одновременно, и если в одной из игр игрок *А* победил игрока *Б*, то и во второй игре победителем является игрок *А*. Такие игры, сходные в своей логике, называют изоморфными.

Все сказанное позволяет сделать такой вывод: вычерченное игровое поле с нанесенными на нем особыми точками можно рассматривать и как «шпаргалку» для игры «Гонки по вертикали». Для этого достаточно сказать себе вместо: «Провожу отрезок до n-й вертикали» фразу: «Продвигаю шашку на n-ю клетку вверх».

Рекомендуем проверить эту рекомендацию в практическом поединке, сыграв с кем-нибудь партию в игру «Гонки по вертикали», используя игровое поле на 13 клеток и ограничение: продвигать шашку вверх можно не более чем на четыре клетки. Иначе говоря, игра проводится при начальных условиях $N=13$ и $M=4$. В игре используйте шпаргалку, изображенную на рис. 4.

Приступаем ко второй части решения задачи, начинаем разработку алгоритма, пригодного для составления программы, работая по которой, ЭВМ сможет успешно соперничать с человеком.

Алгоритм для ЭВМ

Прежде всего заметим, что обучение машины будет вестись так, чтобы она успешно играла во вспомогательную игру. Для этого нам необходимо научить ЭВМ находить все особые точки игрового поля, приготовленного для этой игры. Игровые поля при этом могут быть самыми различными и для любого из них машина должна не только найти, но и запомнить весь рисунок поля с отмеченными особыми точками. Заглядывая в нарисованную в ее памяти шпаргалку, машина и будет вести поединок с человеком.

Начнем с того, что сообщим машине полные размеры игрового поля, т. е. укажем число строк N и число столбцов M. Игровое поле в памяти машины «изобразим» в виде массива чисел из N строк и M столбцов. Образуя массив, присвоим ему имя A и то, что он будет иметь N строк и M столбцов, обозначим так: $A[N, M]$. Читателю ясно, что все точки верхней строки есть особые, в качестве числа, обозначающего особую точку, выбираем ноль. На рис. 6, *а* показан фрагмент схемы алгоритма, который разъясняет, как все элементы верхней строки массива $A[N, M]$ получают значение «0». Это делается в цикле, параметр цикла обозначен буквой I, до тех пор, пока $I \leqslant M$, элементы $A[N, 1]$, $A[N, 2]$ и т. д. до $A[N, M]$ получают значение «0». На рис. 6, *б* показано игровое поле, сконструированное к этому моменту. Затем ЭВМ, работая в соответствии со схемой алгоритма, показанной на рис. 7, *а*, присвоит всем остальным элементам массива $A[N, M]$ значение «2». На рис. 7, *б* показано игровое поле в том виде, в каком оно будет получено: «0» — обозначены особые точки, а цифрой «2» — все точки массива, пока еще не обследованные.

Теперь можно приступить к отысканию остальных особых точек, которые

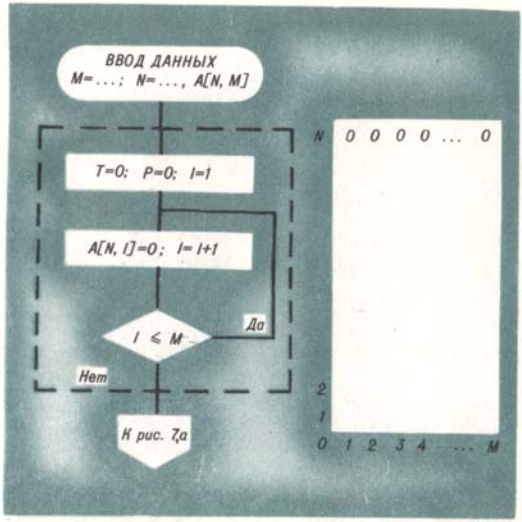
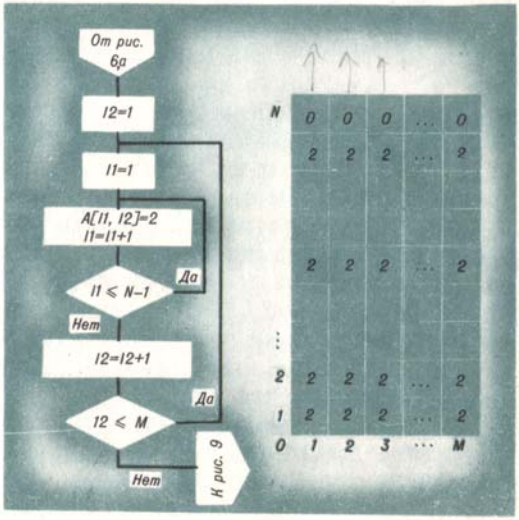


Рис. 6 а б

Рис. 7 а б



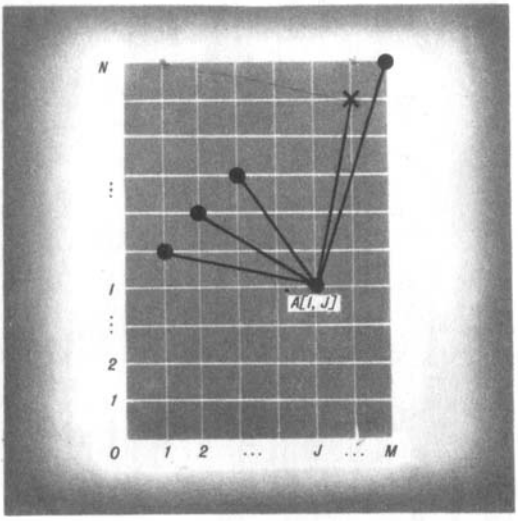


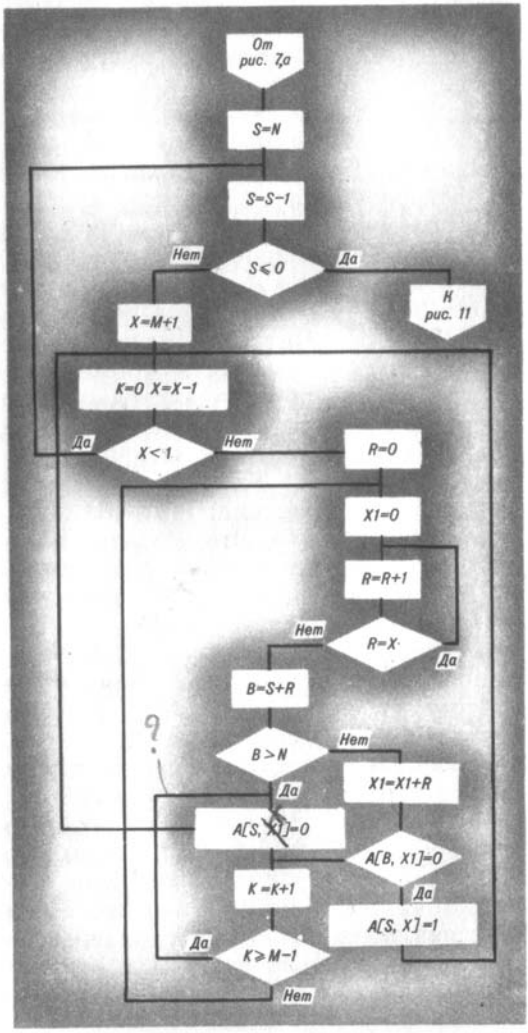
Рис. 8

могут оказаться на данном игровом поле. Делаем это так: рассматриваем все точки массива по очереди и для каждой из них, например $A[I, J]$, мысленно выполняем все возможные, разрешаемые правилами игры, ходы из этой точки. Этот процесс разъясняет рис. 8.

На рисунке кружочками обозначены точки, в которые можно попасть за один ход. Выполняя эту процедуру для каждой точки, а проверку следует начать с точки $A[N-1, M-1]$, т. е. с точки, которая на рис. 8 отмечена крестиком, нужно внимательно следить за тем, в какой — особой или неособой точке можно оказаться, совершив очередной ход. Если, делая ходы из обследуемой точки, мы хотя бы один раз окажемся в особой точке, то можно сделать вывод, что обследуемая нами точка (из нее делают воображаемые ходы) $A[I, J]$ является неособой. Необходимо помнить, что из особой точки в другую особую за один ход попасть нельзя. Если мы установили, что элемент $A[I, J]$ соответствует неособой точке, то ему присваивается значение «1». Если же ни один из правильно совершаемых ходов из $A[I, J]$ не приводит в особую точку, то считаем данную точку особой и присваиваем элементу $A[I, J]$ значение «0». В том, что это так, предлагаем убедиться самостоятельно, разметив игровое поле небольшого размера вручную; поле можно взять размером $N=7$, $M=3$.

Работу машины, которой мы поручаем отыскание особых точек, можно организовать так, как показывает схема алгоритма, изображенная на рис. 9.

Рис. 9

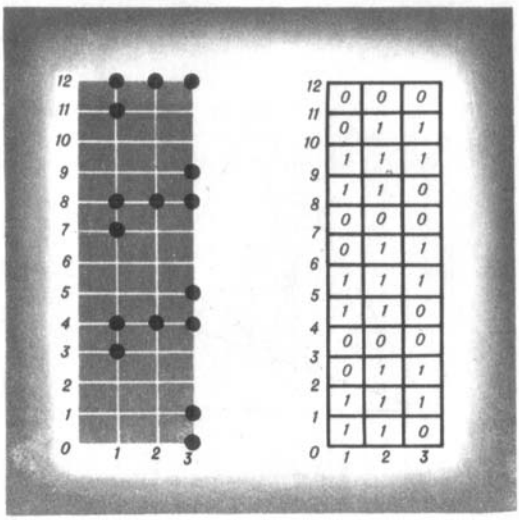


В этой схеме использованы обозначения: S и B — номера строк; X — номер столбца; R — вспомогательная переменная, смысл которой можно понять, если внимательно рассмотреть схему и рис. 8; K — счетчик, с помощью которого проверяется число сделанных ходов при исследовании одной точки; $X1$ — вспомогательная переменная. После работы ЭВМ по этой части общего алгоритма формируется игровое поле с нанесенными на нем особыми точками $A[I, J]=0$. На рис. 10, *а, б* показан пример таким образом подготовленного поля для игры при начальных условиях $N=12$ и $M=3$.

На этом подготовка игрового поля завершена. Как же будет протекать процесс игры?

Приступая к игре и имея в качестве соперника человека, машина, как и человек, должна решить, какой по очереди ей вступать в игру — первой или второй. Для этого она должна проверить, можно ли ей первым ходом попасть в особую точку типа $A[I, J]=0$. Если это возможно, то ЭВМ должна ходить первой, в противном случае — она «уступает» ход человеку.

Рис. 10 а б



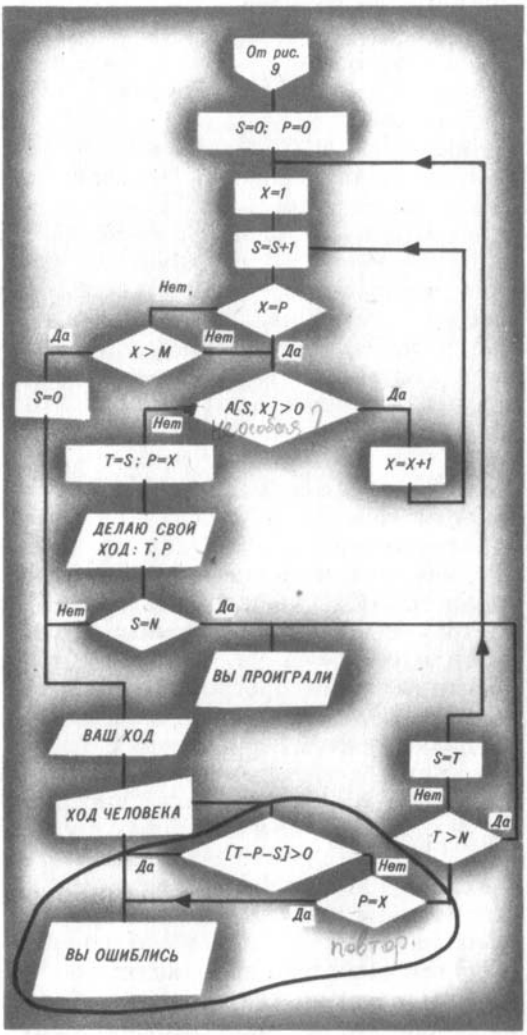


Рис. 11

После того, как решение о вступлении в игру принято, всякий следующий очередной ход будет состоять в том, что из точки (всегда неособой) машина должна «продвигаться» в особую. Последовательность машинных действий, соответствующая этой части общего алгоритма, задается схемой, изображенной на рис. 11, где S — номер горизонтали; X — номер вертикали на игровом поле.

В числе заготовленных реплик, с которыми ЭВМ обращается в ходе поединка к человеку, предусмотрены также реплики «Вы проиграли» и «Вы ошиблись».

На рис. 11 выделен блок проверки каждого хода человека. Проверяется, не сделал ли человек какую-нибудь из двух возможных ошибок: распознаватель $|T-P-S|>0$ выясняет, не сделана ли человеком ошибка, когда он завершил вычерчивание отрезка на вертикали K, в то время как его ход $R \neq K$. Распознаватель $P=X$ проверяет, не повторил ли человек предыдущий ход соперника. Буквами T и P обозначены соответственно номер строки и номер столбца, на котором следует завершать вычерчивание прямой. Если этот же ход переформулировать как ход для игры «Гонки по вертикали», то это будет означать: «Сдвинь шашку вверх на P клеток», тем самым продвинься до горизонтали T.

Полный текст программы, написанной для ЭВМ МИР, и протокол одного поединка даются в приложении.

ИГРА «ТРИГЕКС»

Условие игры. Игра проводится на специальном игровом поле, изображенном на рис. 12.

В игре участвуют двое, ходят по очереди. Один из соперников ходит белыми шашками, другой — черными. Сделать ход — это значит установить шашку своего цвета в один из кружков игрового поля. Победителем считается тот, кто сумеет первым поставить три своих шашки вдоль одной из девяти прямых.

Прежде чем приступать к обучению ЭВМ игре в «Тригекс» и создавать алгоритм беспроигрышного ведения игры, необходимо накопить игровой опыт. Сыграем несколько партий с кем-нибудь.

Быстро обнаруживается, что вступать в игру следует первым и начинать игру ходом на поле *1, 2* или *3*. Затем, как показывает опыт, в ответ на любой первый ход соперника свой, уже второй ход, следует делать так, чтобы две своих шашки стояли на одной прямой. После этого сопернику выбирать уже не приходится, он должен ставить свою шашку на ту же прямую, на которой размещены наши. Иначе, при следующем ходе мы выиграем.

Третий наш ход и следующие за ним ходы при правильной игре приводят к одной из трех ситуаций:

а) заканчивается партия (мы выиграли);

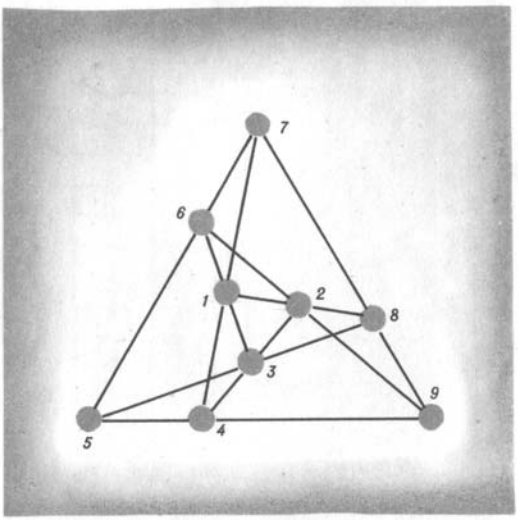
б) шашка ставится на линию, на которой уже размещены две шашки соперника; таким ходом эта опасная для нас линия закрывается;

в) шашка ставится на поле, через которое проходят две линии, уже содержащие по одной шашке нашего цвета.

Ясно, что после этого хода наш выигрыш неизбежен.

Вот к каким рекомендациям мы приходим, накопив опыт. Этих рекоменда-

Рис. 12



ций достаточно, чтобы человек, знакомый с ними, мог успешно играть в «Тригекс». Рекомендациями в такой форме, в какой они были приведены выше, для составления текста программы пользоваться затруднительно, так как мы должны принимать решения, пользуясь не только вычисленными, но и логическими рассуждениями. Например, прежде чем принять решение о том, как ходить, мы должны не вычислять, а как бы заглядывать вперед, продумывать ту ситуацию, которая может возникнуть после хода, и своим ходом делать ее удобной для нас.

Отсюда следует, что ЭВМ привычные ей вычисления будет заменять какими-то другими действиями. В этом случае мы имеем дело с решением задачи больше логического характера.

Машинная программа создается на основе следующего проекта алгоритма:

1. ЭВМ всегда вступает в игру первой и при этом ходит своей шашкой на одно, любое из полей: *1, 2* или *3*.

2. После ответного хода соперника второй ход ЭВМ должна делать, руководствуясь табл. 1, которая хранится в памяти ЭВМ и в которую она будет заглядывать как в шпаргалку. Например, если первый ход ЭВМ — «1» (она установила шашку в кружок, отмеченный цифрой 1), а человек ответил ходом «3», то в соответствии с таблицей машина должна ответить ходом «2». Если бы после первого хода машины «1» человек ответил ходом «7», то второй ход машины должен быть «6».

Таблица 1

Ход ЭВМ	Ход человека								
	1	2	3	4	5	6	7	8	9
1		3	2	3	2	2	6	3	3
2	3		1	1	1	1	3	3	4
3	2	1		1	6	2	2	2	1

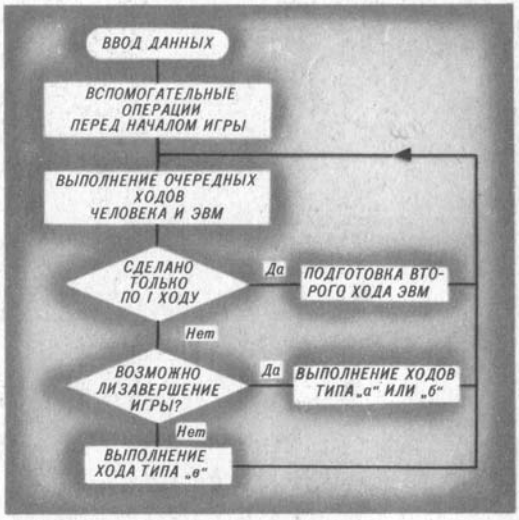


Рис. 13

Еще раз подчеркнем, что таблица используется ЭВМ только для принятия решения только о втором ходе.

3. Свой третий ход и все последующие ЭВМ должна делать, исходя из соображений:

закончить игру (привести ее к ситуации «а»), если этого сделать нельзя, то привести игру к ситуации «б» (закрыть опасную линию, образованную соперником), если и это невозможно, или нет необходимости, то привести игру к ситуации «в».

На рис. 13—16 показаны схемы алгоритмов, обеспечивающие реализацию всего процесса игры.

Прежде всего следует рассмотреть блок ВВОД ДАННЫХ (рис. 14, *а*), где $M=...$ поле, на которое ставит свою шашку ЭВМ.

Рассмотрим блок ВСПОМОГАТЕЛЬНЫЕ ОПЕРАЦИИ ПЕРЕД НАЧАЛОМ ИГРЫ. С помощью цикла с параметром I всем кружкам игрового поля присваивается значение $A[I]=0$ (рис. 14, *б*).

Следующим блоком является блок ВЫПОЛНЕНИЕ ОЧЕРЕДНЫХ ХОДОВ ЧЕЛОВЕКА И ЭВМ (рис. 14, *в*).

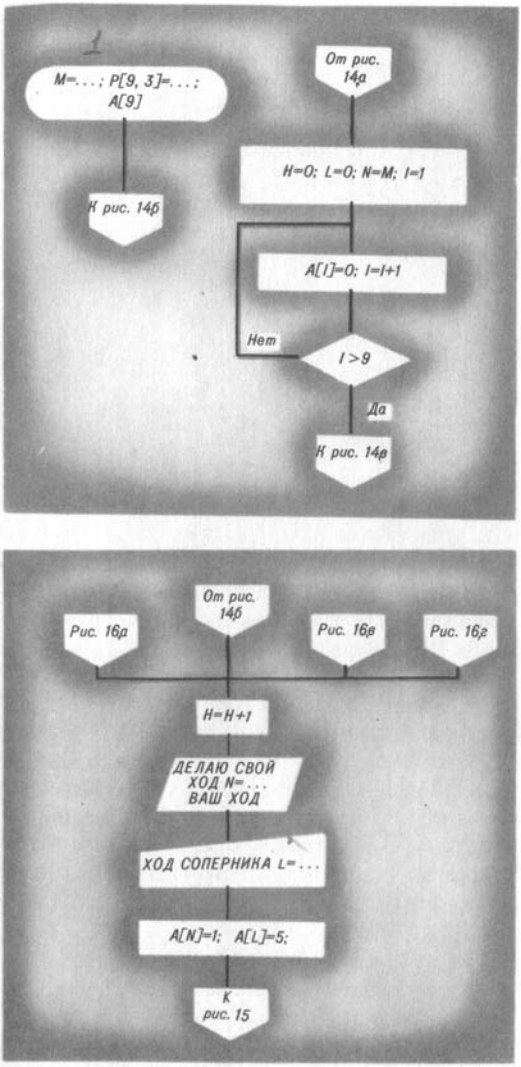


Рис. 14 а б в

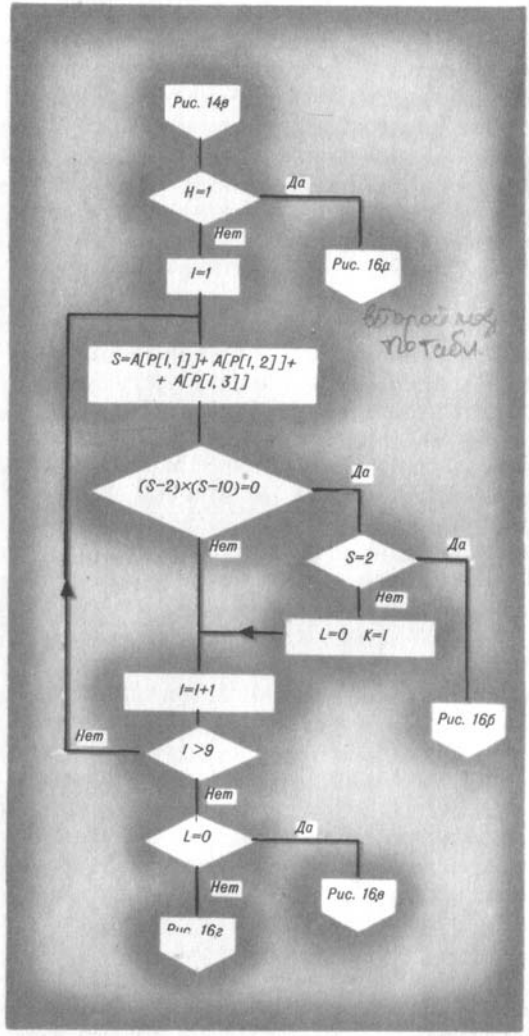
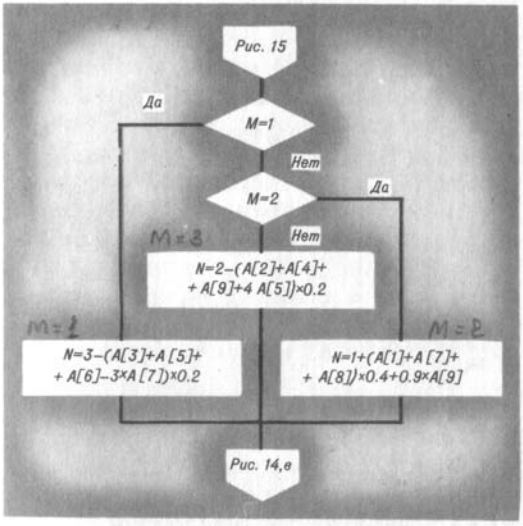
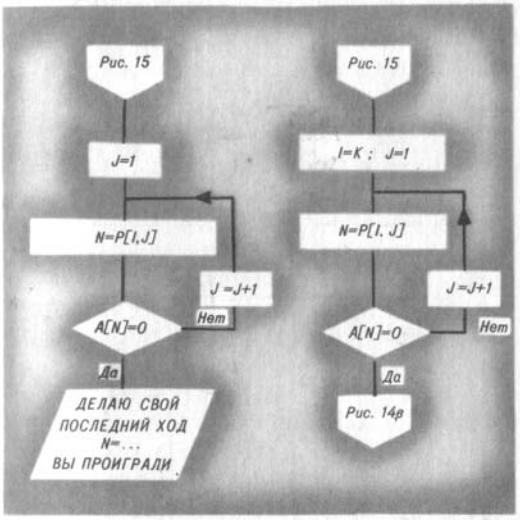


Рис. 15

В этом блоке тем кружкам, на которые поставил свою шашку человек, присваивается значение $A[L]=5$, а тем кружкам игрового поля, на которые «поставила» свою шашку ЭВМ,— значение $A[N]=1$.

На рис. 15 показана схема алгоритма, обеспечивающая выбор необходимого для ЭВМ типа хода. С помощью распознавателя $H=1$ ЭВМ выясняет, следует ли ей делать второй ход по порядку (второй ход выполняется в соответствии с табл. 1). Если машина должна делать не второй ход, то она выясняет, есть ли среди прямых такая, на которой расположено две шашки одного цвета, а третьи кружки пустые. Это делается с помощью цикла, параметр которого обозначен буквой I. Если в ходе проверки ЭВМ обнаружи-





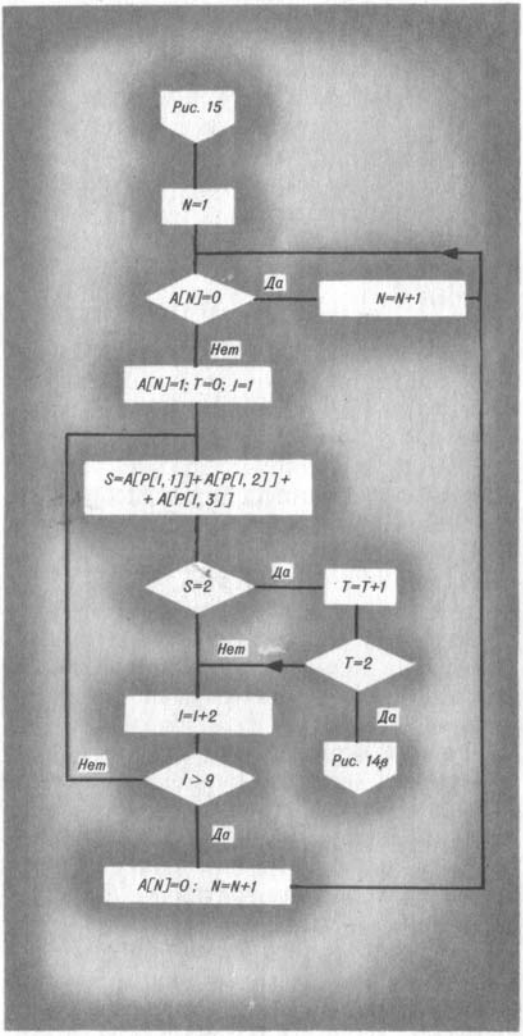


Рис. 16 а б в г

вает такую прямую, то она делает ход типа «а», приводя игру к ситуации «а», и выигрывает поединок или делает ход типа «б», тем самым не давая выиграть человеку. Если в ходе проверки прямая с двумя шашками одного цвета и пустым третьим кружком отсутствует, то ЭВМ делает ход типа «в».

На рис. 16, *а* изображена схема алгоритма, соответствующая блоку ПОДГОТОВКИ ВТОРОГО ХОДА ЭВМ, т. е. хода, соответствующего табл. 1. Заметим, что если $M=1$, то первый ход ЭВМ сделала, поставив шашку на кружок 1, если $M=2$, — на кружок 2 и, если $M=3$,— на кружок 3. Буквой N обозначен номер кружка, на который ходит машина, а номер вычисляется с помощью формул, приведенных в схеме.

На рис. 16, *б* показана схема алгоритма для выполнения ходов типа «а».

Выполнение хода типа «б» задается схемой, изображенной на рис. 16, *в*. На рис. 16, *г* показана схема, в соответствии с которой ЭВМ выполняет ход типа «в». В приложении дается текст программы, протокол одного поединка ЭВМ с человеком и инструкция по применению программы.

ИГРА «НИМ» — ТЕОРИЯ И АЛГОРИТМ

О происхождении этой широкораспространенной и занимательной игры известно немного. По свидетельству М. Гарднера теорию игры «НИМ» впервые разработал профессор Гарвардского университета (США) Чарльз Л. Бутон. Он же в 1901 году придумал название игре — «НИМ».

Для нас игра интересна тем, что беспроигрышная стратегия строится в основном на невычислительном алгоритме. Излагаемые ниже идеи никак не связаны с теорией Ч. Л. Бутона.

Условия игры. Участвуют в игре два соперника, ходят по очереди. В начале игры имеется K групп предметов (камешков, фишек или спичек). До первого хода первая группа содержит m_1 предмет, вторая m_2, третья m_3, k-я группа m_k предметов.

Во время своего хода каждый из играющих может взять себе из любой одной, и только одной группы, любое конечное число предметов, хоть и все имеющиеся к этому ходу предметы в группе. Победителем считается тот, кто, сделав очередной ход, возьмет себе все оставшиеся предметы.

Игра «НИМ» относится к играм антагонистическим, т. е. к играм, в которых не бывает ничьих, один из соперников всегда выходит победителем. Приступая к этой игре, нужно уметь ответить на два вопроса:

каким по очереди, первым или вторым, вступать в игру?

как рассчитывать свой очередной ход в зависимости от хода соперника?

Не зная ответов на эти вопросы, выиграть можно только случайно. Мы же должны построить теорию беспроигрышной игры, на основе которой разработать такой алгоритм, используя который, можно научить ЭВМ успешно соперничать в поединках с человеком.

Построение теории беспроигрышной игры начнем с введения необходимых определений и обозначений. Прежде всего, каждое из чисел $m_1, m_2, m_3, \ldots, m_k$ запишем в двоичной системе счисления и полученные двоичные числа поместим во вспомогательную таблицу, в которой будет столько строк, сколько групп предметов имеется перед началом игры. Иначе говоря, число строк равно K. Число столбцов во вспомогательной таблице зависит от числа разрядов в наибольшем из двоичных чисел:

$$(m_1)_2,\ (m_2)_2,\ (m_3)_2,\ \ldots,\ (m_k)_2.$$

Приведем пример, пусть $K=5$; $m_1=11$, $m_2=23$, $m_3=9$, $m_4=39$, $m_5=30$, тогда таблица будет иметь вид:

	2^5	2^4	2^3	2^2	2^1	2^0	
m_1	0	0	1	0	1	1	11
m_2	0	1	0	1	1	1	23
m_3	0	0	1	0	0	1	9
m_4	1	0	0	1	1	1	30
m_5	0	1	1	1	1	0	39

Различные начальные условия игры приводят к различным вспомогательным таблицам. Среди всевозможных таблиц выделим два типа: особые и неособые.

Особой назовем таблицу, если в каждом ее столбце число единиц четно. Если в столбце одни нули, то будем считать, что в этом столбце число единиц четно.

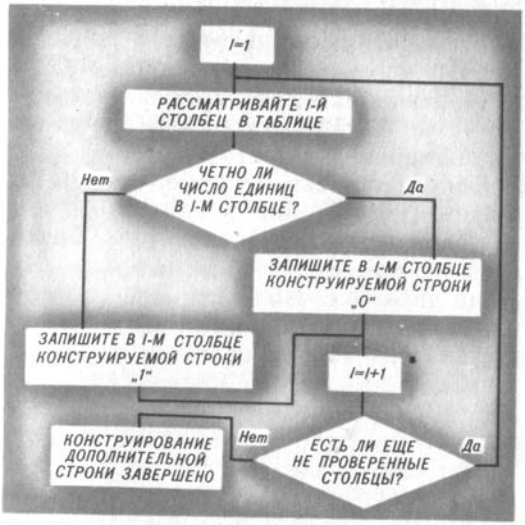


Рис. 17

Неособой назовем таблицу, если хотя бы в одном ее столбце число единиц нечетно.

Рассмотренная таблица является неособой, так как в ее первом, третьем и четвертом столбцах число единиц нечетно. К любой неособой таблице можно приписать одну единственную строку, образуя из данной новую — особую таблицу. Для доказательства этого утверждения указываем алгоритм конструирования такой единственной дополнительной строки. Алгоритм задаем в виде схемы (рис. 17). Ясно, что две различные строки, пользуясь таким алгоритмом, построить нельзя. Полученная таким образом строка называется особой относительно рассматриваемой таблицы и аналогично другим является записью двоичного числа.

Рассмотрим важное для дальнейшего утверждение. Пусть дана таблица в K строк, тогда среди них найдется одна такая, в которой записанное число будет больше числа, записанного в строке, особой относительно таблицы, составленной из $(K-1)$ оставшихся строк.

* Оператор $I=I+1$ означает, что после его выполнения значение переменной I будет увеличено на 1.

Разъясним это на примере. Дана таблица в четыре строки ($K=4$):

1	0	1	1	0	$m_1 = 22$
0	1	0	1	1	$m_2 = 11$
1	0	1	1	1	$m_3 = 23$
1	0	0	1	1	$m_4 = 19$

Эта таблица неособая. Составим строку, особую относительно таблицы, полученную из данной путем выделения первой строки

1 0 1 1 0	$m_1 = 22$–выделенная нами строка исходной таблицы
0 1 0 1 1 1 0 1 1 1 1 0 0 1 1	таблица, к которой будет разыскиваться особая строка
0 1 1 1 1	$m_{oc} = 15$– строка, особая относительно таблицы из трех данных строк

В данном случае оказалось, что первая строка исходной таблицы содержит число $m_1 = 22$, большее, чем число ($m_{oc} = 15$), содержащееся в строке, особой относительно таблицы из трех других строк. Однако то, что именно первая строка оказалась требуемой, это всего лишь случай, ведь выше только утверждалось, что такая строка в таблице найдется, но способ ее отыскания указан не был.

Если бы мы в исходной таблице из четырех строк выделили бы не первую, а вторую строку, то оказалось бы, что $m_2 < m_{oc}$. Действительно, пусть мы выделили вторую строку

0 1 0 1 1	$m_2 = 11$

Из оставшихся трех строк составили таблицу и сконструировали строку, особую относительно этой таблицы

1 0 1 1 0 1 0 1 1 1 1 0 0 1 1	таблица, составленная из трех оставшихся строк
1 0 0 1 0	$m_{oc} = 18$ строка, особая относительно составленной таблицы

Число, соответствующее выделенной нами второй строке $m_2 = 11$, оказалось не больше, а меньше числа $m_{oc} = 18$, записанного в особой строке.

Еще раз подчеркнем: нам известно, что из K строк исходной таблицы какая-то одна содержит число, большее, чем число, содержащееся в строке, особой относительно таблицы, составленной из других строк данной таблицы. Никаких указаний о том, как отыскивать такую строку, не рассматривалось. Задачу отыскания строки, удовлетворяющей требованию, будем решать, исходя из следующих соображений:

выбор подходящей строки — дело несложное, но обременительное. Осуществить выбор всегда возможно, перебрав $n \leqslant K-1$ вариантов. Можно действовать так: выделить первую строку и составить строку, особую для таблицы, составленной из остальных $K-1$ строк. Сравнить числа m_1 и m_{oc}, и если $m_1 \geqslant m_{oc}$, то задача решена. В противном случае такие же действия следует совершить, выделив вторую строку, затем третью и т. д.

Опираясь на приведенные выше соображения, сформулируем и докажем основную в нашей теории теорему.

Теорема. Если таблица, составленная по начальным условиям, особая, то начинающий игру проигрывает.

Доказательство (используется метод математической индукции) *.

Условимся обозначать сумму чисел всех строк особой таблицы буквой S_i. Ясно, что все значения S_i, соответствующие всевозможным особым таблицам, можно расположить в возрастающем порядке так, что между любыми двумя нельзя расположить еще хотя бы одну особую таблицу:

$$S_1 < S_2 < S_3 < \ldots < S_i < S_{i+1} < \ldots$$

Если отбросить, как тривиальный и особый, случай, когда $m_1 = 1$ и $m_2 = 1$, то наименьшее $S_2 = 4$, что соответствует особой таблице, содержащей в двух своих столбцах числа $m_1 = m_2 = 2$. Такая таблица соответствует ситуации: имеется две группы, в каждой из них по два предмета. Легко убедиться в том, что начинающий проигрывает. Таким образом, мы выяснили, что при $i = 1$ и $i = 2$ заключение теоремы верно.

Далее проводим индукцию по «i». Предположим, что начинающий игру в «особой позиции» проигрывает для всех тех особых таблиц, в которых

$$m_1 + m_2 + m_3 + \ldots + m_k \leqslant S_i.$$

Рассмотрим одну из особых таблиц, в которых

$$m_1 + m_2 + m_3 + \ldots + m_k = S_{i+1},$$

и покажем, что в таких условиях начинающий также проигрывает. Пусть начинающий делает ход, уменьшая одно из чисел исходной таблицы (помним, что она является особой), например число m_2, и в таблицу вместо этого числа в ту же строку вписывает число m_2'. В возникшей уже неособой таблице отыскиваем строку r, в которой число m_r больше числа m_{oc}, находящегося

* Для читателей, не знакомых с методом математической индкуции, доказательство можно опустить.

в строке, особой относительно таблицы, составленной из $(K-1)$ строк, т. е. $m_r > m_{oc}$.

Если теперь наш соперник пойдет так, что в группе с номером r останется m_{oc} предметов, то новая таблица окажется особой.

В этой таблице будет две новых строки: одна из них будет содержать число m_2', а другая — число m_{oc}. Сумма всех чисел строки будет

$$m_1+m_2'+\ldots+m_e+\ldots+$$
$$+m_{oc}+\ldots+m_k<S_i.$$

Отсюда следует, что теорема верна и для $i+1$, следовательно, теорема доказана.

Доказанная теорема дает основание для формулирования алгоритма беспроигрышной стратегии в игре «НИМ» при любых начальных условиях.

Алгоритм для игры «НИМ»

Алгоритм задается в виде схемы, показанной на рис. 18.

Приведем пример использования алгоритма для успешной игры при конкретных начальных условиях. Пусть имеется четыре группы предметов, т. е. $K=4$. В группах содержатся следующие количества предметов: $m_1=22$, $m_2=19$, $m_3=23$ и $m_4=11$.

Прежде чем принять решение, как ходить, решаем вопрос — каким по очереди ходить? Действуем, заглядывая в схему алгоритма. Составляем вспомогательную таблицу

1	0	1	1	0	$m_1 = 22$
1	0	0	1	1	$m_2 = 19$
1	0	1	1	1	$m_3 = 23$
0	1	0	1	1	$m_4 = 11$

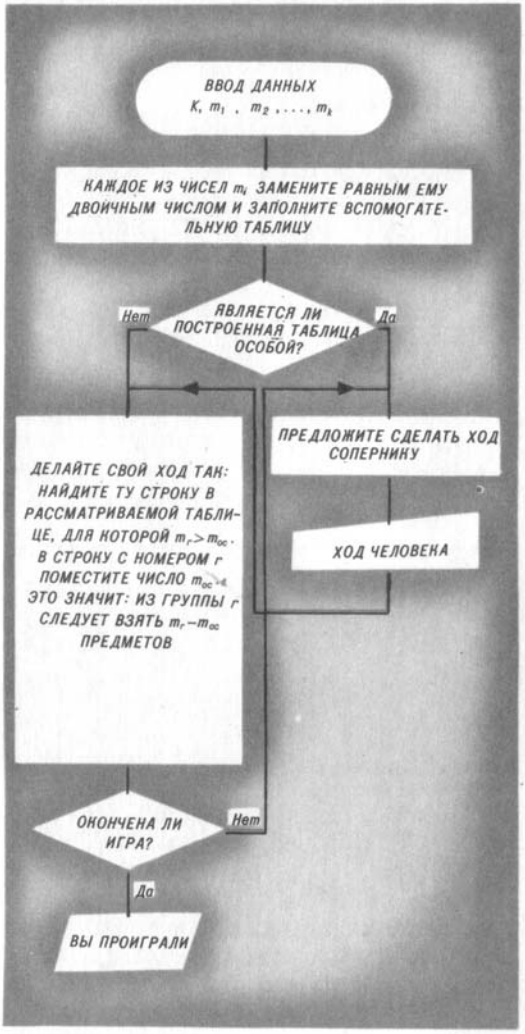


Рис. 18

Выясняем, что таблица является неособой, и поэтому первый ход берем на себя. Нам предстоит теперь, действуя в соответствии с алгоритмом, найти такую строку с номером r, чтобы $m_r > m_{oc}$. Мы это уже делали выше и знаем, что $r=1$ — это первая строка и $m_r=22$, а $m_{oc}=15$. В соответствии с алгоритмом, в первую строку необходимо поместить 15 предметов. Для этого уберем из первой группы семь

лишних предметов. После такого хода таблица будет иметь вид

0	1	1	1	1	$m_1 = 15$
1	0	0	1	1	$m_2 = 19$
1	0	1	1	1	$m_3 = 23$
0	1	0	1	1	$m_4 = 11$

Эта таблица уже особая и ходить должен соперник. Пусть соперник из третьей группы берет 21 предмет, оставляя в ней всего два предмета. После этого таблица будет иметь вид

0	1	1	1	1	$m_1 = 15$
1	0	0	1	1	$m_2 = 19$
0	0	0	1	0	$m_3 = 2$
0	1	0	1	1	$m_4 = 11$

К этой таблице мы должны вновь применить процедуру отыскания нужной нам строки. Такой строкой оказалась вторая, содержащая число $m_2 = 19$, а строка особая в этом случае содержит число $m_{oc} = 6$. Следовательно, наш второй ход: берем из второй группы $m_2 - m_{oc} = 19 - 6 = 13$ предметов. Таблица после нашего хода будет иметь вид

0	1	1	1	1	$m_1 = 15$
0	0	1	1	0	$m_2 = 6$
0	0	0	1	0	$m_3 = 2$
0	1	0	1	1	$m_4 = 11$

Таблица особая, ход за соперником. Пусть он берет 11 предметов из первой группы. Мы вновь составляем таблицу и опять отыскиваем решающую строку

0	0	1	0	0	$m_1 = 4$
0	0	1	1	0	$m_2 = 6$
0	0	0	1	0	$m_3 = 2$
0	1	0	1	1	$m_4 = 11$

Решающей строкой в этот раз оказалась четвертая. Нетрудно рассчитать (и даже увидеть), что нам следует взять все 11 предметов. После нашего хода таблица будет иметь вид

0	0	1	0	0	$m_1 = 4$
0	0	1	1	0	$m_2 = 6$
0	0	0	1	0	$m_3 = 2$
0	0	0	0	0	$m_4 = 0$

Опять ход за соперником. Пусть он берет из второй группы пять предметов. Вычерчиваем новую таблицу

0	0	1	0	0	$m_1 = 4$
0	0	0	0	1	$m_2 = 1$
0	0	0	1	0	$m_3 = 2$
0	0	0	0	0	$m_4 = 0$

Эта таблица неособая и мы должны сделать ход так, чтобы таблица стала особой. Мы уже научились ориентироваться в таблицах и поэтому «в уме»

догадываемся, что взять нужно один предмет из первой кучки. Таблица после нашего хода будет иметь вид

0	0	0	1	1	$m_1 = 3$
0	0	0	0	1	$m_2 = 1$
0	0	0	1	0	$m_3 = 2$
0	0	0	0	0	$m_4 = 0$

В игре сложилась ситуация: имеется три группы, содержащие соответственно $m_1=3$, $m_2=1$ и $m_3=2$ предметов. Если соперник возьмет себе три предмета из первой группы, мы возьмем один предмет из третьей, после чего наша победа несомненна.

Мы научились пользоваться алгоритмом и убедились на примере в том, что он приводит к успеху. Однако мы убедились и в том, что играть быстро нам не удастся — слишком трудно отыскивать свой очередной ход. Алгоритм неудобен для использования человеком в практической игре, так как сложно разобраться в таблицах за то время, которое дается на обдумывание хода. Однако алгоритм вполне удобен для вычислительной машины, ведь ей не грозит «просрочка времени», она успеет обдумать свой ход. Из рассказанного ясно, что алгоритм базируется не на какой-то одной вычислительной формуле, а включает много вспомогательных процедур, в том числе и несколько несложных вычислений.

ЭВМ вооружается алгоритмом

Перейдем к более детальному конструированию алгоритма с тем, чтобы его можно было передать для применения ЭВМ. Детализацию будем осуществлять шаг за шагом, создавая небольшие блоки будущего алгоритма.

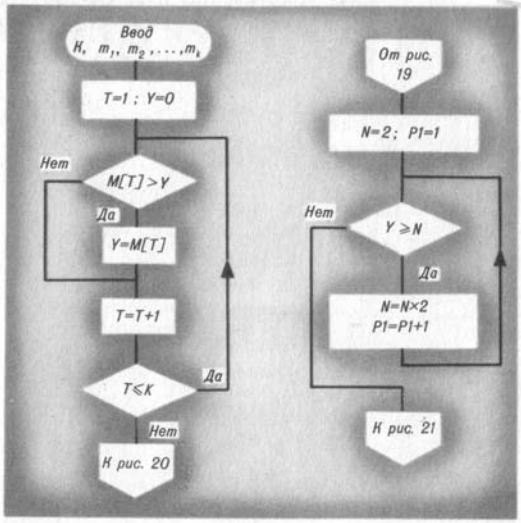


Рис. 19 *Рис. 20*

Первым блоком будет оператор ввода данных (рис. 19), в который вводится число строк K и массив $M[T]$ чисел m_i.

Отыскиваем наибольшее число Y в массиве $M[T]$, состоящем из чисел $M[1]=m_1$, $M[2]=m_2,\ldots,M[K]=m$. Это необходимо для того, чтобы узнать число столбцов в первой из вспомогательных таблиц. После того, как найден наибольший элемент Y, находим число столбцов в первой из вспомогательных таблиц, последовательно сравнивая число Y со степенями числа 2 до тех пор, пока не будет выполнено условие $2^{n-1}<Y\leqslant 2^n$. Найденное значение n присвоим переменной $P1$. Схема полученного блока показана на рис. 20. Выполнив работу, соответствующую этому блоку, машина будет знать значение $P1$ — число столбцов в первой вспомогательной из таблиц.

После этого можно приступить к замене каждого из десятичных чисел m_i равными им двоичными числами и к образованию вспомогательной таблицы. В схеме буквой T обозначен номер строки, а буквой P — номер столбца вспомогательной таблицы, буквами A

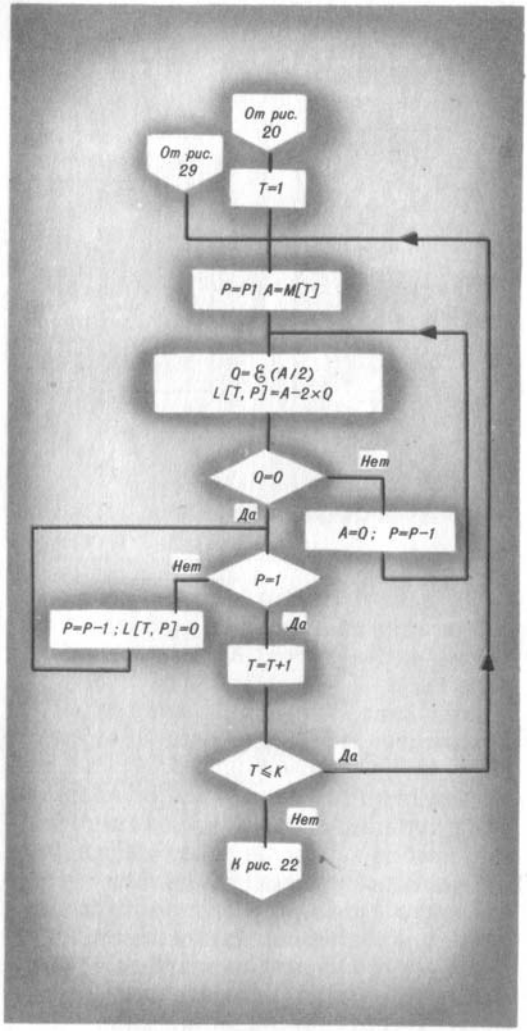


Рис. 21

и Q — рабочие переменные и символом L массив, которому будет соответствовать таблица двоичных чисел из K строк и $P1$ столбцов. Схема алгоритма, обеспечивающая создание вспомогательной таблицы двоичных чисел, показана на рис. 21. Таблица при этом заполняется строка за строкой сверху вниз, а в строке — справа налево.

Таблица составлена, можно приступать к выяснению, является ли она особой. Делается так, как показано на схеме рис. 22. В схеме использован распознаватель $\{R/2\}=0$. Выполнение равенства означает, что дробная часть числа $R/2$ равна нулю, т. е. число R — четное. Выяснив все действия, узнаем, является ли таблица особой $(V=0)$ или неособой $(V=1)$. Если таблица

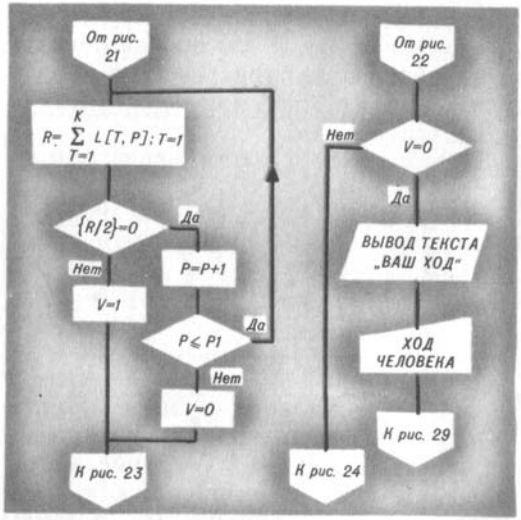
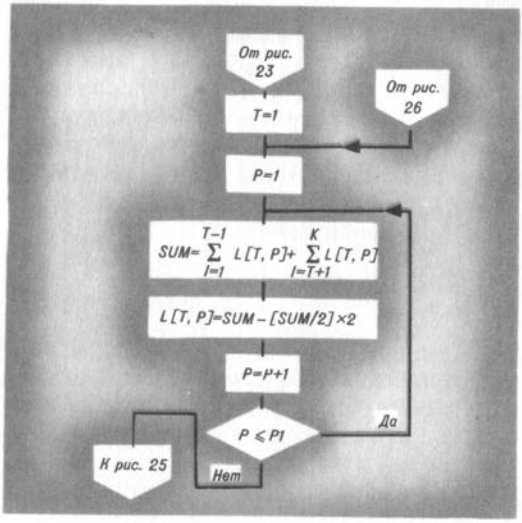


Рис. 22

Рис. 23

Рис. 24



особая, то первый ход, в соответствии с теорией, мы должны предложить сопернику. Следующий блок, приведенный на рис. 23, показывает, как это оформляется.

Если $V=1$, то первый ход должны делать мы. Перед ходом в данной таблице необходимо найти такую строку с номером T, для которой выполняется условие $m_T > Л$, где $Л$ — число, содержащееся в строке, особой относительной таблицы, составленной из данной путем исключения из нее строки с номером T. Делается это так: сначала выделяем первую строку и формируем строку, особую относительно таблицы, составленной из оставшихся строк данной. Это можно выполнить, пользуясь схемой алгоритма, показанной на рис. 24. На рис. 25 показано, как двоичное число в особой строке заменяется равным ему десятичным числом $Л$.

Теперь следует выяснить, выполняется ли условие $Л < M[T]$. Если это условие выполняется, то первая строка используется для того, чтобы выполнить ход, в противном случае T увеличиваем на единицу и процесс повторяем для строки «$T+1$» (рис. 26).

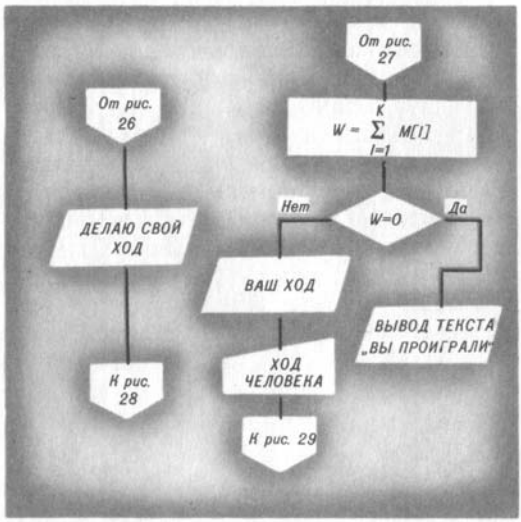


Рис. 27 Рис. 28

ЭВМ должна сделать свой ход. Для его выполнения все уже подготовлено. Ход состоит в том, что в группе с номером T мы оставляем $Л$ предметов (рис. 27). После этого необходимо проверить, не являлся ли этот ход последним, не достигнута ли уже победа? Проверкой является суммирование всех предметов во всех группах (рис. 28).

Рис. 25 Рис. 26

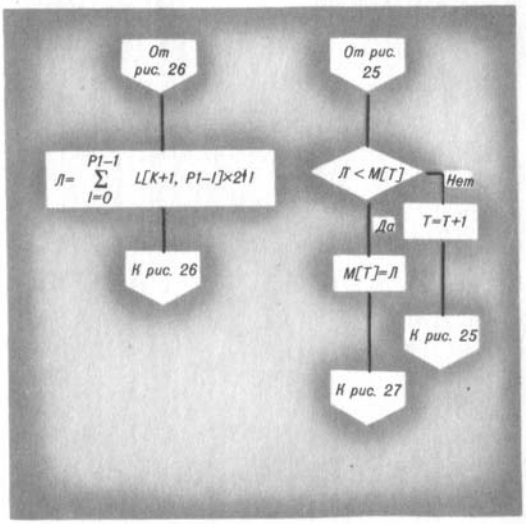
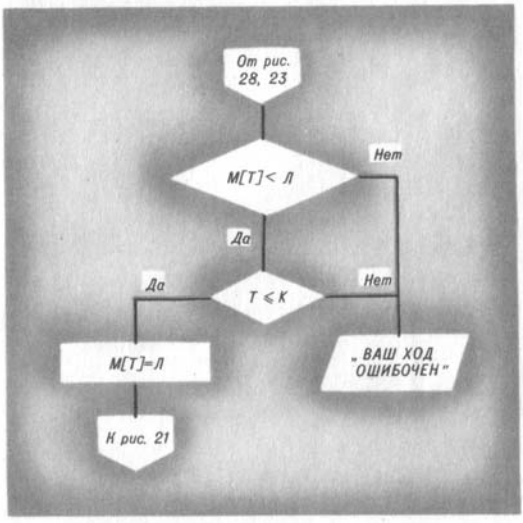


Рис. 29



После хода человеком ЭВМ должна проверить, не сделал ли человек ошибки, выполняя свой очередной ход (рис. 29). Ошибкой может быть неверное указание номера группы K или числа оставляемых в ней предметов. Указание, которое машина напечатает на бумаге,— есть команда для человека сделать свой ход заново и уже без ошибки. Если ход человека сделан без ошибки — работа машины повторяется вновь в соответствии со схемой, показанной на рис. 19. Может показаться, что схема очень громоздка, но следует помнить о том, что пользуется ею ЭВМ, а машине не страшны эти многочисленные мелкие расчеты. Естественно, что при написании текста программы для конкретной ЭВМ удается добиться нескольких упрощений, и в целом программа для игры в «НИМ» получается несложной.

В приложении дается текст программы, протокол одного поединка ЭВМ с человеком и инструкция по применению программы.

ЭВМ

ЗАНИМАЕТСЯ ЛОГИКОЙ

глава

Алгебра
высказываний-
краткая
справка

Чему
равна
истинность
высказывания ?

По
единому
алгоритму

Глава 2

Среди задач, с которыми часто приходится встречаться людям, немало таких, которые принято называть логическими. Кто не знает шуточной задачи о перевозке волка, козы и капусты с одного берега реки на другой, задачи, при решении которой властвует не арифметика, а умение рассуждать. К помощи логики мы прибегаем, составляя расписания, распутывая противоречивые показания и во многих других случаях. В логических задачах исходными данными являются не только числа, а и неожиданные, подчас весьма запутанные суждения. Эти суждения и связи между ними бывают иногда столь противоречивы, что такие твердые орешки не под силу «раскусить» и вдумчивому математику.

Нельзя ли и в этом случае на помощь привлечь арифметические возможности ЭВМ? Оказывается, можно. Ниже рассказывается о том, как, располагая только средствами арифметики, можно обеспечить решение логических задач. Моделирование логических умений, если при этом термин «логическое умение» следует понимать в широком смысле,— важная задача программирования.

АЛГЕБРА ВЫСКАЗЫВАНИЙ — КРАТКАЯ СПРАВКА

Задать конкретную алгебру для ее изучения и использования это значит:

указать множество элементов, над которыми будут выполняться действия (операции);

определить каждую операцию, т. е. дать четкие правила действий ввести обозначения операций;

изучить свойства определенных операций;

привести примеры использования рассматриваемой алгебры для решения прикладных задач.

Учитывая сказанное, в краткую справку об алгебре высказываний введены сведения:

определение «высказывания» и примеры конкретных высказываний;

определение и обозначение операций:

логическое сложение (дизъюнкция),

логическое умножение (конъюнкция),

логическое следование (импликация),

эквивалентность,

логическое отрицание;

сводка основных свойств всех введенных операций над высказываниями;

определение тождественных высказываний (тавтологий) и способ отыскания их.

Итак, объектами алгебры высказываний являются отдельно рассматриваемые повествовательные предложения, относительно каждого из которых имеет смысл говорить, истинно оно или ложно. Такие предложения называются простыми высказываниями. Например:

1. «Вильнюс — столица Литвы».
2. «$13>27$».
3. «Число $2^{11213}-1$ является простым».

Простые высказывания 1 и 3 — истинные, высказывание 2 — ложно.

Приводим примеры предложений, не являющихся высказываниями:

1. «Посмотрите в окно».
2. «Который час?».
3. «$2x+7>12$».

Еще раз подчеркнем, что отличительным признаком любого высказывания является его свойство быть истинным или ложным, а этим свойством три вышеприведенных предложения не обладают.

Условимся простые высказывания обозначать большими буквами и, если высказывание истинно, будем писать $A=1$, а если ложно $A=0$.

В алгебре высказываний над простыми высказываниями определены следующие операции.

Логическое умножение. Соединение двух простых высказываний A и B в одно составное с помощью союза И называется логическим умножением или конъюнкцией, а результат операции, который также является высказыванием,— логическим произведением.

Указание о логическом перемножении простых высказываний A и B обозначается так: $A \cdot B$, иногда AB, или: $A \wedge B$ (конъюнкция высказываний).

Например, пусть даны простые высказывания:

A — «Вильнюс является столицей Литвы»,

B — «В Вильнюсе проживает 1 миллион человек».

Тогда логическим произведением или конъюнкцией этих высказываний будут составное высказывание: «Вильнюс является столицей Литвы и в Вильнюсе проживает 1 миллион человек».

Логическое сложение. Перед тем, как привести определение этой операции, дадим некоторые разъяснения. Союз ИЛИ в обиходе мы применяем в двух значениях: исключающем и неисключающем. Разъясним это примерами.

1. Рассмотрим повествовательное предложение: «Володя вчера в шесть часов вечера читал книгу или ехал в автобусе на стадион». Союз ИЛИ использован в этом предложении в неисключающем смысле — Володя мог читать и одновременно ехать в автобусе. Одно не исключает другого.

2. Рассмотрим еще одно повествовательное предложение: «Володя вчера наблюдал за ходом матча с западной или с восточной трибуны». Здесь союз ИЛИ имеет исключающий характер, две описываемые ситуации исключают друг друга: нельзя наблюдать один и тот же матч одновременно с двух противоположных трибун.

В рассматриваемой алгебре высказываний союз ИЛИ будет употребляться только в неисключающем смысле.

Соединение двух простых высказываний A и B в одно с помощью союза ИЛИ, употребляемого в неисключающем смысле, называется логическим сложением или дизъюнкцией, а полученное составное высказывание — логической суммой.

Указание о необходимости выполнить логическое сложение высказываний A и B записывается так: $A+B$ или $A \vee B$ (дизъюнкция высказываний A и B).

Например, пусть исходными являются простые высказывания: A — «Шесть — число кратное трем», B — «$19>37$».

Тогда логической суммой этих двух высказываний является составное высказывание «Шесть — число кратное трем или $19>37$».

Логическое следование (импликация). Соединение двух высказываний в одно с использованием оборота речи «Если..., то...» называется операцией логического следования или импликацией. Указание выполнить операцию импликации над высказываниями A и B записывается так: $A \rightarrow B$ (читается: «A имплицирует B» или «B следует из A»).

Например, даны высказывания: A — «Трижды по восемь равно 24», B — «Кит — морское животное».

Составное высказывание, полученное после выполнения операции импли-

кации, будет таким: «Если трижды по восемь равно 24, то кит — морское животное».

Эквивалентность. Соединение двух простых высказываний A и B в одно с использованием оборота речи, или как принято говорить, связки «...тогда и только тогда, когда...» называется операцией эквивалентности. Высказывания, над которыми проводится операция эквивалентности, помещаются вместо многоточий. Указание совершить операцию эквивалентности над высказываниями A и B записывается так: $A \longleftrightarrow B$, иногда $A \sim B$ читается: «А эквивалентно В»).

Например, даны простые высказывания: A — «Земля вращается вокруг Солнца по эллиптической орбите», B — «Число 35 кратно 19».

Составное высказывание, полученное после выполнения операции эквивалентности, будет таким:

«Земля вращается вокруг Солнца по эллиптической орбите тогда и только тогда, когда число 35 кратно 19».

В приведенных выше примерах грамматически формально связаны иногда по смыслу совершенно не подходящие друг к другу высказывания, такое соединение высказываний не противоречит определению рассмотренных операций.

Все эти операции были двухместными, т. е. выполнялись над двумя высказываниями. В алгебре высказываний определена и широко используется одна одноместная операция.

Логическое отрицание. Присоединение частицы НЕ к сказуемому данного простого высказывания A называется операцией логического отрицания. Указание выполнить логическое отрицание над высказыванием A записывается так: $\bar{A}$ (читается «A с чертой»). Иногда вместо приведенного определения используют другое, ему эквивалентное: присоединение слов «неверно, что..» ко всему данному высказыванию A называется операцией логического отрицания. В результате выполнения операции логического отрицания получается новое высказывание.

Например, пусть дано высказывание: A — «Число 5 является делителем числа 30».

Тогда отрицанием его, образованным по первому определению, будет высказывание:

$\bar{A}$ — «Число 5 не является делителем числа 30», а если использовать второе определение операции отрицания, то получим

A — «Неверно, что число 5 является делителем числа 30».

Следовательно, истинность составных высказываний, образованных в результате выполнения логических операций над простыми высказываниями, зависит только от истинности исходных высказываний. В табл. 2 приведены все значения сложных высказываний при всех комбинациях значений используемых высказываний в каждой из логических операций. Знак «1» обозначает слово «истинно», а «0» — «ложно».

Таблица 2

A	1	1	0	0
B	1	0	1	0
A + B	1	1	1	0
AB	1	0	0	0
$A \rightarrow B$	1	0	1	1
$A \longleftrightarrow B$	1	0	0	1
$\bar{A}$	0	0	1	1

При образовании сложных высказываний из простых можно использовать не одну, а несколько логических операций, например,

$\bar{A}+B$; $\quad A \longleftrightarrow \bar{B}$;

$\bar{A}\bar{B}+C$; $\quad \bar{A} \rightarrow B$.

Составные высказывания можно обозначать одной буквой, например,

$$X=A+B;\ У\equiv\overline{A\to B};\ K\equiv C\overline{Д\to B}.$$

В дальнейшем над высказываниями X, $У$ и K в свою очередь можно выполнять логические операции, считая их простыми высказываниями.

Введем понятие таблицы истинности высказывания, включающей все значения, которые может принять сложное высказывание. Рассмотрим пример составления таблицы истинности сложного высказывания $\overline{A\bar{B}+\bar{A}}$ (табл. 3).

Таблица 3

A	B	$\bar{A}$	$\bar{B}$	$A\bar{B}$	$A\bar{B}+\bar{A}$	$\overline{A\bar{B}+\bar{A}}$
1	1	0	0	0	0	1
1	0	0	1	1	1	0
0	1	1	0	0	1	0
0	0	1	1	0	1	0

При рассмотрении таблиц истинности может оказаться, что для двух составных высказываний эти таблицы совпадают. Иначе говоря, эти высказывания принимают одинаковые значения при одинаковых значениях входящих в них переменных (простых высказываний). Высказывания, таблицы истинности которых совпадают, называют эквивалентными. В табл. 4 приведены три пары таких высказываний. Эквивалентные высказывания можно соединять знаком «$\equiv$». При необходимости в формулах вместо одного высказывания можно вписывать ему эквивалентное.

Таблица 4

A	B	A+B	B+A	AB	BA	$A\leftrightarrow B$	$B\leftrightarrow A$
1	1	1	1	1	1	1	1
1	0	1	1	0	0	0	0
0	1	1	1	0	0	0	0
0	0	0	0	0	0	1	1
		$A+B\equiv B+A$		$AB\equiv BA$		$A\leftrightarrow B\equiv B\leftrightarrow A$	

Используя табл. 4, можно записать:

$$A+B\equiv B+A;$$

$$AB\equiv BA \text{ и } A\leftrightarrow B\equiv B\leftrightarrow A.$$

Среди высказываний особое значение имеют так называемые тождественно истинные высказывания, т. е. такие, значение истинности которых всегда равно 1 и не зависит от того, какие значения истинности принимают высказывания, входящие в них, например,

$$A+\bar{A};\qquad AB\leftrightarrow BA;$$

$$A\to A;\qquad (A+B)\leftrightarrow(B+A).$$

Тождественно истинные высказывания называют часто тавтологиями. В форме тавтологий выражены основные свойства операций алгебры высказываний.

Наряду с тождественно истинными рассматриваются и тождественно ложные высказывания, т. е. такие, значение истинности которых всегда равно нулю, например,

$$A\bar{A} \text{ или } AB(A\to B).$$

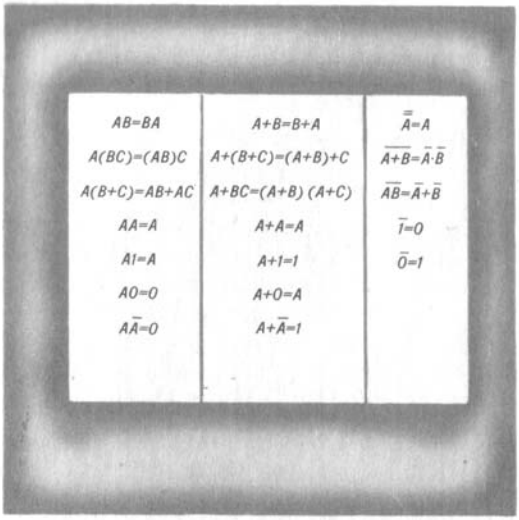
Заметим, что если в формуле сложного высказывания, как часть ее, встречается тождественно ложное или тождественно истинное высказывание, то вместо него можно записать соответственно 0 или 1. Поскольку ниже наибольшее внимание будет уделяться операциям логического сложения, умножения и отрицания, то свойства этих операций выделены на рис. 30.

Отметим также два важных свойства операций импликации и эквивалентности:

$$A\to B\equiv\bar{A}+B;$$

$$A\leftrightarrow B\equiv A\bar{B}+\bar{A}B.$$

В том, что последние два высказывания верны, легко убедиться, если составить таблицы истинности для каждого из них. Эти два свойства позволяют формулы, в которых встречаются знаки импликации и эквивалентности, преоб-



$AB=BA$	$A+B=B+A$	$\bar{\bar{A}}=A$
$A(BC)=(AB)C$	$A+(B+C)=(A+B)+C$	$\overline{A+B}=\bar{A}\cdot\bar{B}$
$A(B+C)=AB+AC$	$A+BC=(A+B)(A+C)$	$\overline{AB}=\bar{A}+\bar{B}$
$AA=A$	$A+A=A$	$\bar{1}=0$
$A1=A$	$A+1=1$	$\bar{0}=1$
$A0=0$	$A+0=A$	
$A\bar{A}=0$	$A+\bar{A}=1$	

Рис. 30

разовывать в тождественные им формулы, но такие, в которых будут использоваться только операции логических умножения, сложения и отрицания.

ЧЕМУ РАВНА ИСТИННОСТЬ ВЫСКАЗЫВАНИЯ?

Заметим, что если сложное высказывание состоит из небольшого числа простых, то таблицу истинности для него можно заполнить без особого труда, вручную. Однако с ростом числа различных букв (а это значит и увеличения числа простых высказываний), входящих в формулу сложного высказывания, объем работы резко возрастает. Действительно, если букв две (действия проводятся над двумя высказываниями), то таблица истинности будет содержать четыре строки, если букв три, то в таблице будет уже восемь строк, т. е. с прибавлением каждой новой буквы число строк будет удваиваться. Вообще, если число простых высказываний, входящих в составное, равно m, то число строк в таблице истинности $n=2^m$. При $m=12$ $n=2^{12}=4096$ и в дальнейшем растет очень быстро. Ясно, что вручную заполнять столь громоздкие таблицы истинности невозможно.

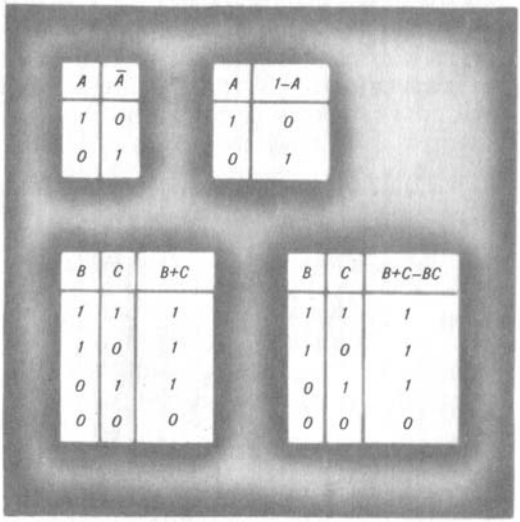
Однако задача вычисления всех значений истинности сложного высказывания, пусть даже заданного громоздкой формулой, очень важна, в частности, важна для конструкторов, создающих логические устройства-автоматы для преобразования информации.

Нельзя ли поручить заполнение таблицы истинности вычислительной машине?

Оказывается, можно, и если машина умеет выполнять логические операции, то задача становится тривиальной. Можно, однако, научить машину заполнять таблицу истинности и в том случае, если она не умеет выполнять логические операции.

Разработка программы для вычислительной машины, работая по которой она сумеет правильно заполнить таблицу истинности сложного высказывания и при этом будет применять только дей-

Рис. 31 а б *Рис. 32 а б*



A	$\bar{A}$
1	0
0	1

A	$1-A$
1	0
0	1

B	C	$B+C$
1	1	1
1	0	1
0	1	1
0	0	0

B	C	$B+C-BC$
1	1	1
1	0	1
0	1	1
0	0	0

ствия арифметические,— есть еще одна занимательная задача на программирование.

Прежде чем поручать машине заполнение таблицы истинности какого-либо сложного высказывания в целом, научим ее вычислять значения отдельно рассматриваемых логических произведения, суммы и отрицания. Приведем таблицу, пользуясь которой находят значения логического отрицания (рис. 31, *а*). Рядом дана таблица (рис. 31, *б*), в которой формула «1—А» есть уже формула арифметическая.

Пользуясь такой арифметической формулой, можно получить такие же значения для $\overline{A}$, как и при выполнении над A операции логического отрицания. Нам удалось построить арифметическую модель логической операции отрицания.

Логическое и арифметическое умножение дают одни и те же результаты, поэтому особую модель логического умножения не строим.

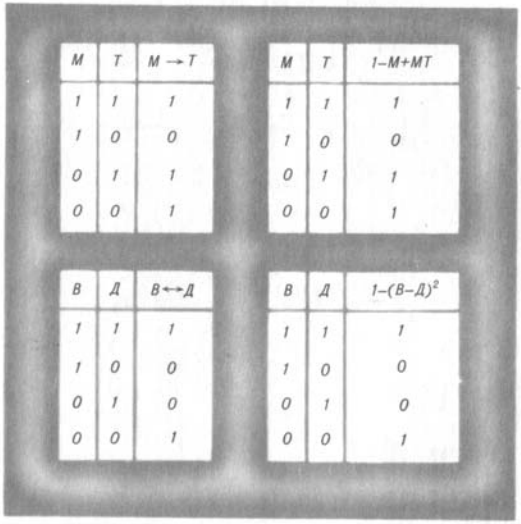
Значение логической суммы находим, пользуясь таблицей, показанной на рис. 32, *а*. Арифметической моделью будет таблица, изображенная на рис. 32, *б*.

Более сложно моделируется операция импликации. На рис. 33, *а* и *б* показаны формулы и таблицы, из которых ясно, как это делается.

На рис. 34, *а* и *б* показано как моделируется операция эквивалентности.

Еще раз подчеркнем, что мы научились «моделировать» основные логические операции, иначе говоря, научились вычислять значения логической суммы, произведения, импликации, эквивалентности и отрицания, используя только арифметические действия.

Продолжим решение поставленной задачи. Если формула высказывания, все значения истинности которого необходимо найти, сложная, то предварительно необходимо построить арифметическую модель, соответствующую формуле сложного высказывания.



M	T	M → T
1	1	1
1	0	0
0	1	1
0	0	1

M	T	1–M+MT
1	1	1
1	0	0
0	1	1
0	0	1

В	Д	В↔Д
1	1	1
1	0	0
0	1	0
0	0	1

В	Д	1–(В–Д)²
1	1	1
1	0	0
0	1	0
0	0	1

Рис. 33 а б *Рис. 34 а б*

Рассмотрим пример. Пусть необходимо составить таблицу истинности для высказывания, заданного формулой

$$X \equiv (A \rightarrow B)(\overline{A}+\overline{B}).$$

Арифметическую модель получаем так:

$A \rightarrow B$ заменяем на $1-A+AB$;

$\overline{A}+\overline{B}$ заменяем сначала на $\overline{AB}$, а затем на $1-AB$ и наконец всю формулу заменяем на $У=(1-A+AB)(1-AB)$.

Полученная формула есть уже формула арифметическая, каждая буква в ней принимает значения не «истина» и «ложь», а единица и нуль. Вычисленные по этой формуле значения могут быть либо только единицей, либо нулем, но истолковывать их по окончании вычислений мы должны соответственно как «истина» и «ложь».

После того, как арифметическая модель построена, задача сводится к тому, чтобы найти все значения полученного арифметического выражения. Делать это следует последовательно, перебирая всевозможные наборы значений букв, входящих в формулу.

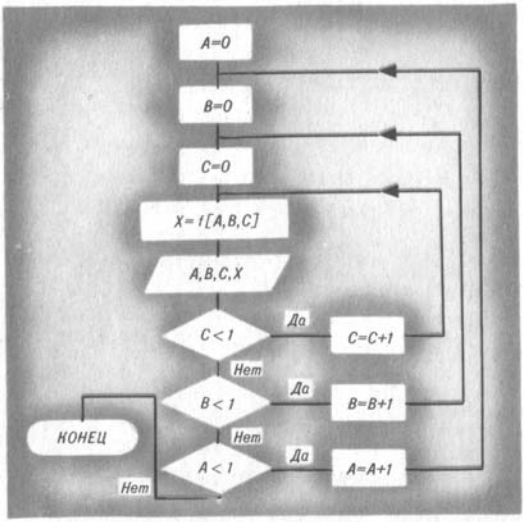


Рис. 35

Алгоритм такой работы для арифметического выражения, содержащего три переменных, например, *A, B* и *C*, может быть таким, как заданный схемой, изображенной на рис. 35.

Приведенная схема алгоритма включает в себя три цикла, вложенные один в другой. Внутренний цикл имеет параметр *C*, средний — параметр *B* и внешний — параметр *A*. Каждый из этих параметров принимает всего два числовых значения: 1 и 0.

Заполнение таблицы начинается с вычисления значения арифметического выражения, содержащего переменные *A, B* и *C* при $A=B=C=0$. Затем параметр *C* принимает значение $C=1$ и вычисляется второе значение арифметического выражения, которое мы обозначили как $F[A, B, C]$.

Таблица 5

A	B	C	X = F (A, B, C)
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Дальнейшие вычисления организуются так, что таблица заполняется сверху вниз (табл. 5).

Разработка алгоритма, работая по которому ЭВМ сумеет заполнить таблицу истинности для сложного высказывания, содержащего три простых, оказалась несложной.

Нетрудно сообразить, как следует дополнить схему алгоритма, если число простых высказываний в сложном будет увеличено.

Умение решать эту несложную, но занимательную для программиста задачу дает основание взяться за решение еще одной занимательной задачи — научить ЭВМ решать логические задачи.

ПО ЕДИНОМУ АЛГОРИТМУ

Пусть нам дана следующая логическая задача:

«Алеша, Боря и Гриша нашли в земле сосуд. Рассматривая удивительную находку, каждый высказал по два предположения:

Алеша: «Это сосуд греческий и изготовлен в V веке».

Борис: «Это сосуд финикийский и изготовлен в III веке».

Гриша: «Это сосуд не греческий и изготовлен в IV веке».

Учитель истории сказал ребятам, что каждый из них прав только в одном из двух предположений.

«Где и в каком веке изготовлен сосуд?»

Несколько замечаний о решении таких задач. В условии нам даны высказывания школьников и учителя, некоторые из них противоречат друг другу.

Решить задачу — это значит найти истинное высказывание, отвечающее на поставленный в задаче вопрос.

Еще раз подчеркнем, что в качестве данных и в качестве разыскиваемой величины также выступает высказывание. При решении алгебраических задач в школе буквами обозначали неизвестные количества (число дней, скорость или что-то другое), а сейчас буквами обозначаем высказывания.

В данной задаче примем следующие обозначения:

«Это сосуд греческий» — $Г$.

«Это сосуд финикийский» — $Ф$.

«Сосуд изготовлен в V веке» — $П$.

«Сосуд изготовлен в III веке» — $Т$.

«Сосуд изготовлен в IV веке» — $Ч$.

После того, как введены обозначения простых высказываний, записываем сложные высказывания — предположения школьников.

Алеша сказал: «Это сосуд греческий и изготовлен в V веке» — такое сложное высказывание можно записать так: «$Г \cdot П$».

Со слов учителя следует, что это высказывание ложно, ведь Алеша прав только в чем-то одном: или $Г=1$, или $П=1$. Истинным будет высказывание

$$Г\overline{П}+\overline{Г}П=1.$$

Действительно, либо первое слагаемое $Г\overline{П}$ истинно (Алеша верно угадал, что сосуд греческий, но ошибся во времени его изготовления), либо истинно второе слагаемое $\overline{Г}П$ (Алеша не угадал место изготовления, но угадал время изготовления).

Рассуждая аналогично, получим еще два сложных истинных высказывания

$$Ф\overline{Т}+\overline{Ф}Т=1;\ \overline{Г}\overline{Ч}+Г\overline{Ч}=1.$$

Каждое из этих высказываний будем рассматривать как логическое уравнение. При составлении этих трех логических уравнений использованы высказывания ребят и замечание учителя. Но этого еще недостаточно, следует учесть, что ложными будут и высказывания

$$Ф\overline{Г}=0;\ П\overline{Т}=0;\ П\overline{Ч}=0 \text{ и } Т\overline{Ч}=0$$

или, что то же самое:

$$\overline{Ф}+\overline{Г}=1;\ \overline{П}+\overline{Т}=1;$$

$$\overline{П}+\overline{Ч}=1 \text{ и } \overline{Т}+\overline{Ч}=1.$$

Содержание каждого ясно — речь идет о том, что сосуд изготовлен только в одной из стран и в одном из веков. У нас имеется теперь семь уравнений с пятью высказываниями. Образуем из них систему

$$\begin{cases} \overline{Г}П+Г\overline{П}=1 \\ \overline{Ф}\overline{Т}+Ф\overline{Т}=1 \\ \overline{Г}\overline{Ч}+Г\overline{Ч}=1 \\ \overline{Ф}+\overline{Г}=1 \\ \overline{П}+\overline{Т}=1 \\ \overline{П}+\overline{Ч}=1 \\ \overline{Т}+\overline{Ч}=1 \end{cases}$$

Если все эти истинные высказывания логически перемножить, то получим сложное высказывание, в котором сведено все, что говорится о сосуде:

$$X=(Г\overline{П}+\overline{Г}П)(Ф\overline{Т}+\overline{Ф}Т)(\overline{Г}\overline{Ч}+$$
$$+Г\overline{Ч})(\overline{Ф}+\overline{Т})(\overline{П}+\overline{Т})(\overline{П}+\overline{Ч})(\overline{Т}+\overline{Ч}).$$

Обозначим эту формулу так:

$$X\equiv F(Г,Ф,П,Ч,Т).$$

Теперь ясно, что решить задачу — это значит указать, при каких значениях $Г$, $Ф$, $П$, $Ч$ и $Т$ это сложное высказывание истинно. Иначе говоря, нам нужно заполнить таблицу 6 и найти ту единственную строку, в которой $X=1$.

Таблица 6

Г	Ф	П	Ч	Т	X=f (Г,Ф,П,Ч,Т)
0	0	0	0	0	
0	0	0	0	1	
0	0	0	1	0	
0	0	0	1	1	

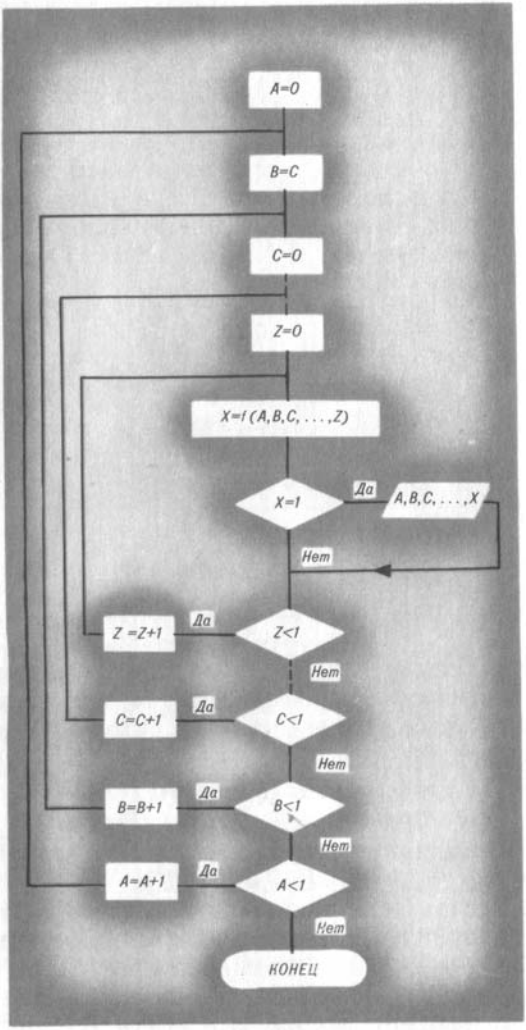


Рис. 36

По существу задача свелась к предыдущей, к задаче заполнения таблиц истинности, что мы уже научились делать.

Рассмотрим схему алгоритма (рис. 36), которая только чуть-чуть отличается от схемы алгоритма, изображенной на рис. 35. В нее введен один распознаватель $X=1$ и один оператор вывода значений $A, B,..., Z, X$. Это сделано для того, чтобы можно было обнаружить все наборы значений переменных, при которых $X=1$, и отпечатать их в качестве ответа к задаче.

Перед созданием арифметической модели высказывания $X=F(Г,Ф,П,Т,Ч)$ его можно упростить, используя свойства основных операций, например таких, как $A\bar{A}=0$ или $A+\bar{A}=1$. Но можно этого и не делать. После сказанного ясно, каким можно представить себе алгоритм, пригодный для решения на ЭВМ любой логической задачи.

В заключение отметим, что правильный ответ на вопрос, поставленный в задаче, дает высказывание «Сосуд изготовлен в Финикии в V веке».

Из сказанного ясно, что основная трудность в решении логических задач связана с получением отдельных логических уравнений, отражающих всю связь между используемыми в условии задачи высказываниями. Условие задачи бывает иногда запутанным и без применения ЭВМ разобраться в нем очень сложно. В качестве примера даем текст задачи, рассматриваемой И. М. Ягломом в статье «Алгебры Буля».

«Рассмотрим упрощенный учебный план, где неделя включает всего три учебных дня — понедельник, среду и пятницу, причем каждый день содержит не более трех пар учебных часов. В течение недели учащиеся должны иметь три пары учебных часов по математике, две — по физике и по одной — по химии, истории и физкультуре. При этом:

1) математик настаивает, чтобы его часы никогда не были последними и по крайней мере два раза — первыми;

2) физик желает, чтобы его часы также не были последними; по крайней мере один раз он хочет иметь первую пару часов; в среду он должен быть свободен первые два часа, а в пятницу, напротив того, может работать лишь первые два часа;

3) историк может преподавать лишь в понедельник в течение первых четы-

рех часов или в среду в течение третьего и четвертого часов; кроме того, он не желает, чтобы его занятия непосредственно предшествовали физкультуре;

4) химик настаивает, чтобы его занятия происходили не в пятницу и не в те дни, когда учащиеся занимаются физикой;

5) занятия по физкультуре проводятся на стадионе, и поэтому естественно требовать, чтобы они были последними в свой день; кроме того физкультурник в пятницу занят на другой работе;

6) естественно требовать, чтобы в течение каждого учебного дня у учащихся было не больше двух часов занятий по одному и тому же предмету;

7) свободные от занятий два часа в рамках учебной недели из $3\times3=9$ пар часов (из числа которых заняты лишь $3+2+1+1+1=8$ пар часов) должны приходиться на последнюю пару часов в пятницу или первую пару часов в понедельник.

Как можно составить расписание с соблюдением всех поставленных условий?

Условия задачи можно выразить следующей системой логических уравнений:

$$F_1 \equiv M_3M_6M_9(M_1M_4+M_1M_7+M_4M_7)=1;$$
$$F_2 \equiv \Phi_3\Phi_6\Phi_9(\Phi_1+\Phi_4+\Phi_7)\Phi_4\Phi_8\Phi_9=1;$$
$$F_3 \equiv (И_1+И_2+И_5)(И_1C_2+И_2C_3+И_4C_5+И_5C_6+И_7C_8+И_8C_9)=1;$$
$$F_4 \equiv X_7X_8X_9(\Phi_1X_2+\Phi_1X_3+\Phi_2X_1+\Phi_2X_3+\ldots+\Phi_9X_7+\Phi_9X_8)=1;$$
$$F_5 \equiv (C_3+C_6+C_9+C_2O_3+C_5O_6+C_8O_9)C_7C_8C_9=1;$$
$$F_6 \equiv (M_1M_2+M_1M_3+M_2M_3+M_4M_5+\ldots+M_8M_9)(\Phi_1\Phi_2+\Phi_1\Phi_3+\ldots+\Phi_8\Phi_9)=1;$$
$$F_7 \equiv O_1O_9=1.$$

Эту систему можно заменить одним сложным логическим уравнением

$$F=F_1F_2F_3F_4F_5F_6F_7=1.$$

Результатом решения задачи будет два варианта расписания.

1. Понедельник: математика, история, химия; среда: математика, физика, физкультура; пятница: физика, математика, свободные часы.

2. Понедельник: математика, физика, физкультура; среда: математика, история, химия; пятница: физика, математика, свободные часы.

Задача логическая — ключ арифметический

Среди задач, которые часто приходится рассматривать программистам, занимающимся решением экономических вопросов, есть две, издавна считающиеся «твердыми программистскими орешками».

В первой задаче речь идет о вычислении «расстояния» в днях между двумя произвольными и сколь угодно удаленными друг от друга датами. В задаче исходными данными являются две даты:

указывается год, месяц и день каждой из них и требуется узнать, сколько дней отделяет одну дату от другой.

Во второй задаче требуется по известной дате, которая может быть днем прошлого, даже очень отдаленного от нас, или днем будущего узнать, какой день недели приходится на эту дату. Например, пусть необходимо узнать, в какой день недели стартовал в космос Юрий Гагарин — в понедельник, вторник или какой-либо иной, если известна дата старта его — 12 апреля 1961 года.

Обе эти задачи трудно отнести к типичным арифметическим задачам, это скорее логические задачи. В рассуждениях при решении этих задач следует учитывать, что число дней в феврале зависит от того, является ли данный год високосным или нет. Сильно затрудняет решение второй задачи то обстоятельство, что число дней в различных месяцах неодинаково.

Если эти задачи предлагается решить на ЭВМ, то перед программистом встает вопрос, как средствами арифметики и только арифметики решить эти задачи. Программистами предложено немало остроумных программ, работая по которым вычислительная машина успешно решает обе задачи. Остановимся на решении, которое является оригинальным и соответствует принятому в книге стилю — оно является занимательным и поучительным.

Речь идет о решении, оформленном в виде двух расчетных формул: одна используется при вычислении «расстояния» в днях между датами, другая — для «вычисления» дней недели.

Рассмотрим первую из расчетных формул, для чего введем обозначения используемых переменных:
S — расстояние между датами в днях, равное разности $I_2 - I_1$, где I_1 и I_2 — индексы более ранней и более поздней дат; I — индекс даты, который является функцией от трех параметров, задающих каждую дату: день $(Д)$, номер месяца $(М)$ и год $(Г)$, $I=F(Д,М,Г)$. Например, индекс даты 12 апреля 1961 года — есть значение функции $I=F(12,4,1961)$.

В рассматриваемом решении индекс даты для дней января и февраля вычисляется по одной формуле, а для дат, дни которых приходятся на март—декабрь — по другой. Проводим обе расчетные формулы: для января-февраля

$$I=365\times Г+Д+31\times(М-1)+\varepsilon(0{,}25\times\varepsilon(Г-1))-\varepsilon(0{,}75\times\varepsilon(((Г-1)\times 0{,}01)+1)),$$

для марта—декабря

$$I=365\times Г+Д+31\times(М-1)-\varepsilon(0{,}4М+2{,}3)+\varepsilon(0{,}25Г)-\varepsilon(0{,}75\times(\varepsilon(0{,}01\times Г)+1)).$$

Буквой ε в каждой из формул обозначена функция «Целая часть от действительного значения аргумента». Значение этой функции равно наибольшему целому числу, не превышающему значение аргумента. Функция обозначается как $\varepsilon(X)$ и $[X]$ и часто называется «Антье от X» (от французского Entier, что значит «целый»). Примеры вычисленных значений функции: $\varepsilon(7)=7$; $\varepsilon(1/5)=0$ и $\varepsilon(-4{,}2)=-5$.

Решение поставленной задачи на ЭВМ после открытия приведенных выше формул сводится к умению вычислять значение функции $\varepsilon(X)$. Моделирование вычисления значений функции Антье от X на ЭВМ — это еще одна занимательная задача на программирование, которая вошла в перечень обязательных для многих типов ЭВМ. Наряду с функцией Антье от X программисты широко пользуются аналогичной функцией «Дробная часть от X», которая обозначается $\{X\}$. С применением этой функции читатель познакомился, когда рассматривал алгоритм беспроигрышной игры в «НИМ».

Рекомендуем читателю самостоятельно построить алгоритмы, в которых разрешается использовать все арифметические операции и операцию сравнения чисел так, чтобы, пользуясь ими, можно было вычислять значения функций $[X]$ и $\{X\}$.

Решение второй задачи основано на использовании значения «Индекса даты». Если вычисляемую величину «день недели» обозначить через $ДН$, то расчетная формула будет иметь вид

$$ДН=1+\varepsilon(1-1/7)\times 7+6.$$

После вычислений переменная $ДН$ может оказаться равной только одному из чисел $\{0,1,2,...,6\}$. Если $ДН=0$, то это означает, что день недели, соответствующий рассматриваемой дате, есть воскресенье, если $ДН=1$, то этот день понедельник и так далее до $ДН=6$, что означает: вычисленное значение дня недели есть суббота.

В приложении дается текст программы, работая по которой ЭВМ вычисляет расстояние между двумя датами и определяет день недели, приходящийся на каждую из них.

ЭВМ

МОДЕЛИРУЕТ ДОГАДКУ

глава

Прелюдия к задаче

Алгоритм узнавания

Глава 3

Читатель знает, как трудно иногда найти удачную идею при решении задачи, как нелегко подчас сообразить, какой ход в сложившейся позиции будет наилучшим.

Догадка ведь не спешит и озарение по заказу не приходит. Умение догадываться — удивительное из умений,— столь важное и желанное, но так таинственно. Механизмы угадывания, тайны смекалки и сообразительности — все это пока за семью печатями.

Как проникнуть в эти тайны?

О том, как ЭВМ моделирует некоторые детали процесса узнавания, рассказывается ниже. Обучая ЭВМ умению распознавать простейшие образы, человек приоткрывает пути постижения тайн восприятия мира, механизмы отражения действительности в своем мозгу.

Это еще один пример решения на ЭВМ невычислительной задачи.

ЭВМ моделирует догадку

ПРЕЛЮДИЯ К ЗАДАЧЕ

В задачах так называемого психологического практикума, задачах-тестах ставится проблема отыскать закономерность, которой руководствуются составители, отбирая числа, слова или картинки в некоторые группы.

На рис. 37, *а* показана таблица, в клетки которой вписано восемь чисел. Составитель таблицы отбирал и располагал числа, руководствуясь каким-то, только ему известным, принципом. Необходимо догадаться, каким принципом он руководствовался, и суметь подобрать число, которое можно было бы «законно», в соответствии с задуманным составителем принципом, поместить в клетку таблицы, отмеченную знаком вопроса.

Попробуйте догадаться и Вы. Проверьте себя, рассказав о том, каким принципом Вы руководствовались при поиске числа, своим товарищам. Удалось ли Вам убедить их в своей правоте сразу?

На рис. 37, *б* показана аналогичная таблица, но в ней размещены уже не числа, а слова. Задача ставится прежняя: догадайтесь до той закономерности, которой руководствовались составители таблицы при отборе и размещении слов в клетках таблицы, придумайте требуемое слово и запишите его в пустую клетку.

Еще одна похожая задача, на наш взгляд более простая. Дана таблица, в клетках которой размещены не числа и не слова, а простые незамысловатые картинки (рис. 38). Облегчение работы при решении этой задачи связано с тем, что картинку, помещаемую в пустую клетку таблицы, придумывать не нужно, ее следует только выбрать из предлагаемых шести, столь же простых перенумерованных картинок.

Вы, конечно, попробуете решить и эту задачу и в случае удачи сумеете

объяснить, почему Ваш выбор пал на ту или иную картинку. Согласитесь ли Вы с тем, что разыскиваемой будет картинка с номером 6?

Читатель наверное согласится с тем, что эти задачи действительно требуют сообразительности и умения анализировать. Проблема, которая возникает при решении такого рода задач, состоит в том, чтобы сформулировать условие, руководствуясь которым мы объединяем в одну «нашу» группу несколько чисел, слов или картинок. Затем, убедившись в том, что признаки деления чисел или картинок на «наши» и «не наши» сформулированы четко, можем производить отбор, выделять «наши» и «не наши» картинки, слова или числа из предъявляемых новых, ранее не показанных нам картинок, слов или чисел.

Рассматривая числа или картинки, предъявляемые в задаче, мы как бы обучались узнавать «свои» картинки среди многих других. Это обучение у одних людей протекает легче, другим труднее понять принцип, по которому следует «распознавать» свои образы среди других.

Задача распознавания — классификации предъявляемых объектов — исключительно важная в деятельности людей. Для нас важно быстро узнавать нужные фигуры, например, трапеции или ромбы, важно и хорошо распознавать животных, механизмы, узнавать слова другого языка и т. д.

Все эти задачи мы чаще всего учимся решать, рассматривая образцы трапеций, ромбов, животных и т. п. Метод показа — один из важных методов, используемых при обучении распознаванию образов. Во время рассмотрения образцов мы создаем в нашем сознании модель-образ фигуры, который, постепенно уточняясь, служит основой для узнавания фигуры, машины, животного или какого-либо другого объекта.

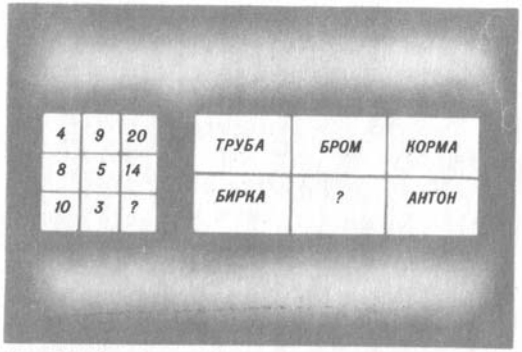
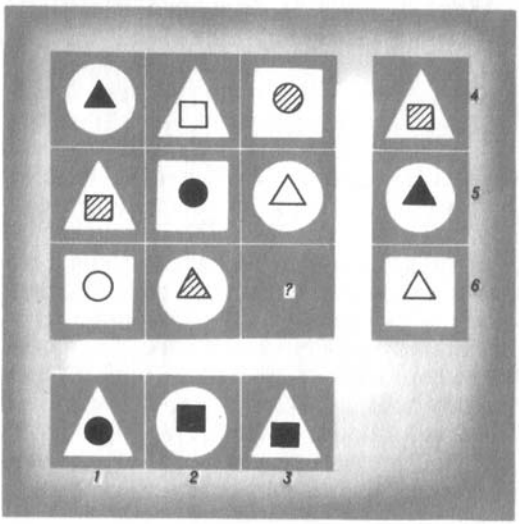


Рис. 37 а б

Рис. 38



Следовательно, если бы можно было научить ЭВМ искусству «узнавания», то круг решаемых машинами задач значительно расширился.

Ниже рассматривается один из несложных алгоритмов распознавания и программа, работая по которой ЭВМ может узнавать среди предъявляемых ей карточек «наши» и «не наши». Для обучения машины искусству распознавания карточек ей предварительно показывают карточки. Рассмотрев несколько «наших» карточек, ЭВМ «догадывается» о секрете, по которому они отбираются, и создает в своем сознании обобщенную модель «нашей» карточки, а затем, руководствуясь этой моделью, уверенно узнает «наши» карточки среди «чужих».

АЛГОРИТМ УЗНАВАНИЯ

Пусть имеется набор карточек. Каждая карточка состоит из 9 клеток. Каждая из клеток может содержать кружок или же быть пустой. На рис. 39 изображены 12 карточек — часть всего набора. Всего в наборе $2^9=512$ карточек.

Предлагается, задумав какой-то «принцип отбора», отложить в сторону несколько карточек, которые Вы отнесли к «нашим». Например, можно считать карточку «нашей», если в левой верхней и правой нижней ее клетках содержатся кружки. Разметка других клеток Вас при этом не интересует. При таком принципе отбора среди карточек, изображенных на рис. 39, нет ни одной «нашей». Принцип выделения «наших» карточек может быть и более сложным.

Этап обучения распознаванию «наших» карточек начнем с того, что в обучающую «выборку», т. е. в комплект показываемых машине карточек, включим все «наши» карточки. Пусть это будут карточки *2, 9, 10* и *12*. После этого работу будем проводить так:

предъявим ЭВМ первую из «наших» карточек. Машина должна занести в свою память полное описание этой карточки. Под полным описанием конкретной карточки будем понимать массив X, состоящий из девяти чисел, каждое из которых либо 1, либо 0. На рис. 40 изображена одна из карточек *(2)*, рядом показано, как следует оформлять массив, описывающий эту карточку. Если в клетке есть кружок, то соответствующий элемент массива X_i равен 1, в противном случае — 0.

Рассмотренную карточку будем считать эталоном «наших» карточек, напомним, что это карточка *2* из данного набора. Затем все карточки из обучающей выборки (напомним, что в этой выборке только «наши» карточки) одну за другой предъявляем ЭВМ, которая сопоставляет их с эталонной карточкой. Цель сопоставления — выделить общее, присущее только «нашим» карточкам.

Схема алгоритма сопоставления двух карточек показана на рис. 41. Сопоставление карточек — это по существу сопоставление двух числовых массивов X и X' описанного вида; X_i — клетка эталонной карточки, X_i' — соответствующая клетка сопоставленной карточки.

Работу алгоритма проследим на примере сопоставления карточки *2* с карточкой *9*.

Итак, в памяти ЭВМ уже содержится полное описание эталонной карточки *2*. Вводим в память машины описание сопоставляемой карточки *9*. Далее ЭВМ сравнивает числа X_1 и X_1' и, так как «содержание» соответствующих клеток одинаково, то в качестве следующих сравниваются вторые клетки, содержание которых также одинаково. Далее сравниваются третьи клетки, т. е. сравниваются числа X_3 и X_3'. Обнаруживается, что $X_3=0$, а $X_3'=1$, а это значит, что содержимое этих клеток несущественно: хотя этими клетка-

ми сопоставляемые карточки и отличаются друг от друга, это не мешает им быть одновременно «нашими». Делаем вывод о том, что в дальнейших сопоставлениях карточек клетку X_3 можно из рассмотрения исключить.

Переходим к сопоставлению четвертых клеток и, обнаружив, что их содержимое одинаково: $X_4=X_4'=0$, переходим к сравнению пятых клеток карточек *2* и *9*. Содержание их различно: $X_5=1$, а $X_5'=0$, поэтому и клетка X_5 исключается из рассмотрения. Затем убеждаемся в том, что содержимое шестых и седьмых клеток одинаково, и переходим к сравнению восьмых клеток. Здесь $X_8=0$, а $X_8'=1$, следовательно, в дальнейшем клетку X_8 исключаем при сравнении новых карточек. Последними сравниваются девятые клетки — они одинаковые в каждой из сопоставляемых карточек. Сравнение эталонной карточки *2* с карточкой *9* завершено.

Для дальнейшего уточнения образа «нашей» карточки переходим к сопоставлению эталонной карточки *2* с карточкой *10,* используя тот же алгоритм (карточка также «наша»). В процессе сравнения выясняется, что из рассмотрения можно исключить клетки с номером *7* и *9* — их содержимое для «наших» карточек несущественно.

Предъявляем последнюю карточку *12* из заготовленной нами обучающей выборки. В результате сопоставления этой карточки с эталонной карточкой *2* удается исключить из рассмотрения клетку *4*.

Сопоставление позволило отобрать общее, что было свойственно только тем карточкам, которые входили в обучающую выборку. Это общее заключается в том, что у «наших» карточек $X_1=X_2=0$ и при этом $X_6=1$. Итог нашей работы можно задать в виде формулы: $H=\overline{X}_1\overline{X}_2X_6$, что означает: карточку следует считать «нашей», если $X_1=X_2=0$, а $X_6=1$.

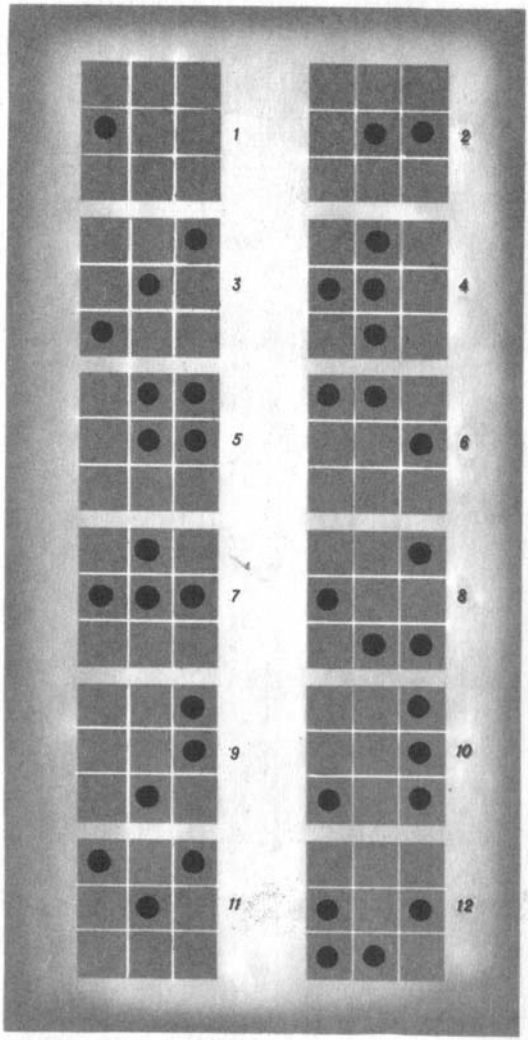


Рис. 39

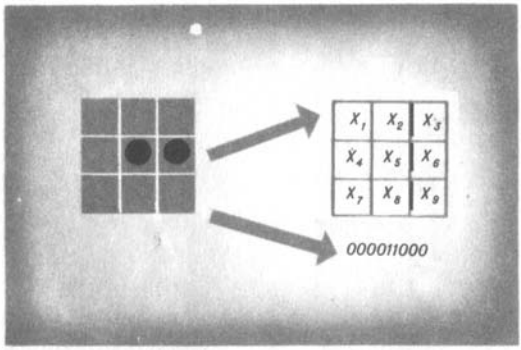
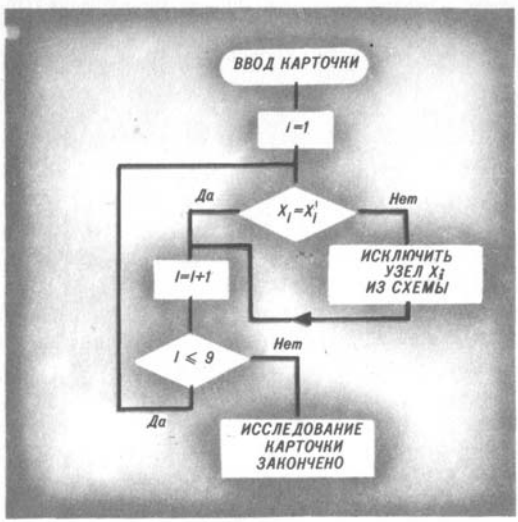


Рис. 40

Рис. 41



Формула $H=\overline{X}_1\overline{X}_2X_6$ задает необходимые условия принадлежности любой карточки к множеству «наших». Эти условия мы отыскали в процессе обучения, в процессе сопоставления карточек, входивших в обучающую выборку. Однако нужно быть осторожным, так как в процессе обучения в числе признаков, по которым отбирались «наши» карточки, могут оказаться лишние. Например, если отличительный признак для «наших» карточек задать формулой $H=X_8X_5$, а в процессе обучения показать только карточки с номерами *3, 5* и *11,* то выяснится, что к «нашим» можно отнести и карточки, описание которых задается формулой $H=X_3X_4X_5X_8X_9$. Это случилось потому, что в обучающем массиве не встретились карточки с различным содержанием клеток X_4, X_8 и X_9.

Ясно, что организованное таким образом обучение не позволит машине действовать при распознавании карточек абсолютно точно. Может оказаться так, что недостаточно обученная машина, т. е. машина, которой показали мало «наших» карточек, будет действовать против нас; будет отбрасывать некоторые карточки из тех, которые мы сами отнесли бы к «нашим». Однако и это очень важно, так как ЭВМ, обученная описанным выше методом, никогда «чужую» карточку не примет за «нашу».

Итак, мы научили ЭВМ «догадываться» до секрета, которым руководствуется человек, выделяя «свои» карточки среди множества «чужих». Для машины это значит, что она научилась выделять общее в двоичных словах, описывающих конкретные карточки. Ведь машина самих-то карточек и не видит, в ее памяти все время хранится и, по мере необходимости, преобразуется образ карточки в виде девятибуквенного (в рассмотренном примере) слова.

У некоторых читателей может сложиться впечатление, что предлагаемый

метод обучения пригоден только для простых случаев, аналогичных рассмотренному. В действительности карточки в нашем примере содержали всего девять клеток и клетки «раскрашены» были всего в два цвета — это ограничивало фантазию и возможности при конструировании требований к «нашим» карточкам. Все это так, но не следует делать вывода об ограниченности такого метода обучения в принципе.

Представьте себе, что мы имеем дело не с двухцветными, а с трехцветными или многоцветными карточками. Раскраску каждой из них можно закодировать, обозначить одним единственным двоичным словом. А это значит, что обучение отысканию секрета в раскраске многоцветных карточек сводится к обучению отыскания секрета в отборе «наших» слов в массиве двоичных слов, т. е. задача становится известной, решение ее нам доступно.

В заключение отметим, что ЭВМ, которую мы обучаем на основе рассмотренного алгоритма, может приобретать опыт, изменять условие, по которому она «узнает» «наши» карточки среди многих других. Для этого достаточно найти еще несколько «наших» карточек и провести еще один дополнительный урок. После каждого такого урока ЭВМ будет все более уверенно отличать «свои» карточки от «чужих».

Поведение машины напоминает в описанном процессе поведение человека, который, обучаясь на примерах, шаг за шагом приобретает опыт. Конечно, это сходство поверхностное, много нюансов того, как человек учится догадываться, в таком машинном процессе отразить не удается. Описанный алгоритм — это всего лишь модель обучения догадке и модель, при этом, весьма грубая.

Однако, как всякая модель важного процесса, она может найти применение и может послужить основой для более

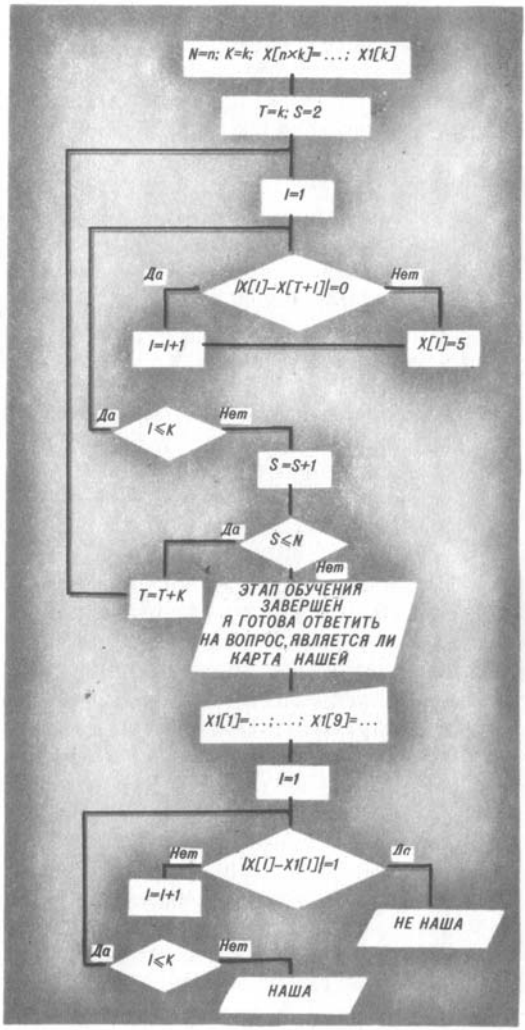


Рис. 42

тонких рассуждений и для построения более детальных моделей.

Второй частью обсуждаемой задачи является разработка алгоритма применения машиной уже найденного ею в процессе обучения признака: «карточка наша». На рис. 42 изображена схема алгоритма, который берется в основу написания программы для ЭВМ. Работая по этой программе, машина сначала обучается, а затем и распознает предъявляемые ей двухцветные карточки.

Обсудим схему алгоритма подробно. Исходными данными для применения алгоритма (они записаны в блоке ввода данных) являются: N — количество карт в обучающей выборке, K — количество клеток в каждой двухцветной карте, $X[N \times K]$ — массив, элементами которого являются коды каждой карты обучающей выборки, перечисленные подряд (сначала K цифр кода, описывающих одну карту, затем K цифр описания другой карты и так далее до кода n-й карты). В блоке ввода данных описывается и рабочий массив $X1[K]$, который используется для ввода в машину каждой распознаваемой карты. Буквами T и S обозначены рабочие переменные: T присваивается сначала значение K, а затем каждое новое ее значение на K возрастает. С помощью этой переменной удается организовать просмотр кодов карт из обучающей выборки; S — номер карты, сопоставляемой с эталонной при обучении. В качестве эталонной всегда берется карта с номером *1*.

Работа в процессе обучения начинается с того, что код эталонной карты, который включает в себя K цифр, сравнивается с кодом карты *2*, т. е. соответственно сравниваются числа X_1, X_2,...,X_k с числами X_{k+1} , X_{k+2} ,...,X_{2k}. Те элементы сравниваемых массивов, которые оказываются несущественными в эталонной карте, заменяются числом 5. После сопоставления первой карты со второй начинается сопоставление первой и третьей карт, т. е. соответственно сравниваются числа X_1, X_2,..., X_k и X_{2k+1} , X_{2k+2} ,...., X_{3k} . Этот процесс продолжается до тех пор, пока не будут сопоставлены все n карт, входящих в обучающую выборку. После этого ЭВМ выдает на печать текст «Этап обучения завершен. Вводите исследуемую карту» и ждет предъявления ей массива $X1[K]$, который и описывает показываемую для исследования ЭВМ карту. С этого момента начинается этап распознавания.

Основным на этом этапе является сравнение кода эталонной карты и кода карты, введенного в виде массива $X1[K]$, для чего в приводимой схеме алгоритма используется распознаватель $|X[I]-X1[I]|=1$, с помощью которого удается увидеть клетки сравниваемых карт. В одной из них записана 1, а в другой — 0. Это и есть главный признак того, что сравниваемая карта «не наша».

В приложении приведен текст программы для ЭВМ МИР, инструкция для применения программы и пять примеров работы машины по программе.

ЭВОЛЮЦИЯ НА ЭКРАНЕ

ЭВМ

Глава

Эволюция по заказу

Алгоритм моделирования эволюции на экране

Глава 4

С появлением ЭВМ возникла возможность моделирования новых явлений: работы энергетических систем, деятельности цеха, завода и даже отрасли народного хозяйства. С помощью ЭВМ математическое моделирование стало мощным методом исследований, на моделях проверяются варианты работы будущих гигантских плотин, космических кораблей, а также оцениваются различные подходы в организации дорогостоящих научных экспериментов и многое другое.

Математики в содружестве с биологами предпринимают первые попытки моделировать процессы жизни: явления в развитии живой клетки, поведение отдельных организмов или их коллективов.

Ниже рассказывается об удивительной искусственной эволюции, организатором и руководителем которой выступает ЭВМ.

ЭВОЛЮЦИЯ ПО ЗАКАЗУ

Идея еще одной занимательной задачи на программирование подсказана работой американского специалиста Станислава Улама «Некоторые математические проблемы, связанные с процессом роста фигур и статьей известного популяризатора математики Мартина Гарднера «Игра «Жизнь».

Ниже дается краткое изложение идей, обсуждаемых авторами в вышеупомянутых статьях, а затем формулируется задача на программирование.

Авторы рассматривают воображаемую искусственную «Жизнь», протекающую по своим, не очень сложным, но чрезвычайно любопытным для наблюдателя, законам. Вот что пишет М. Гарднер: «...«Жизнь» можно отнести к быстро развивающейся в наши дни категории игр, имитирующих процессы, происходящие в живой природе. Возникающие в процессе игры ситуации очень похожи на реальные процессы, происходящие при зарождении, развитии и гибели колоний живых организмов».

Пространство, в котором протекает «жизнь», может быть двухмерным или трехмерным. Начнем рассмотрение с эволюции, протекающей в двухмерном пространстве. Жизненная среда, в которой протекает эволюция, есть бесконечная плоскость, разбитая на одинаковые квадратные поля так, как это показано на рис. 43. На каждом поле жизненного пространства может быть расположена только одна особь. Несколько рядом расположенных особей образуют структуру. Рядом — это значит расположена в соседнем по горизонтали, вертикали или диагонали поле. У каждой особи может быть не более восьми соседей. На рис. 43 слева изображена одна отдельная особь, а справа — две структуры: одна из трех особей, а другая — из шести.

Организовать эволюцию или начать процесс жизни структур — это значит:

поместить в жизненное пространство какую-либо начальную структуру, состоящую из нескольких особей;

ввести в действие «условные часы», которые такт за тактом, как метроном, будут вести отсчет времени начавшейся эволюции. Время в этой искусственной жизни течет скачками, подобно тому, как это мы видим на электрических часах;

сформулировать законы эволюции и ввести их в действие.

Еще раз процитируем М. Гарднера: «Основная идея состоит в том, чтобы начав с какого-нибудь простого расположения фишек (особей), расставленных по одной в клетке (поле), проследить за эволюцией исходной позиции (структуры) под действием «генетических законов», которые управляют рождением, гибелью и выживанием фишек (особей)».

От того, как мы сформулируем законы эволюции (генетические законы) и от того, к какой начальной структуре мы их применим, будет зависеть интенсивность, разнообразие и продолжительность эволюции. Начнем с рассмотрения простых законов эволюции.

Закон выживания особи — если особь имеет две или три соседних особи, то она сохраняет себя на следующем такте жизни.

Закон гибели особи — если особь имеет больше трех или меньше двух соседних особей — она гибнет. В первом случае — от перенаселенности, во втором — «от одиночества». Гибель особи означает снятие ее с поля.

Закон рождения особи — если какое-либо поле жизненного пространства окружено в точности тремя особями, т. е. имеет трех соседей, то к концу такта на нем рождается особь.

Заметим, что приведенные законы — это лишь один из вариантов генетических законов, можно задать и другие.

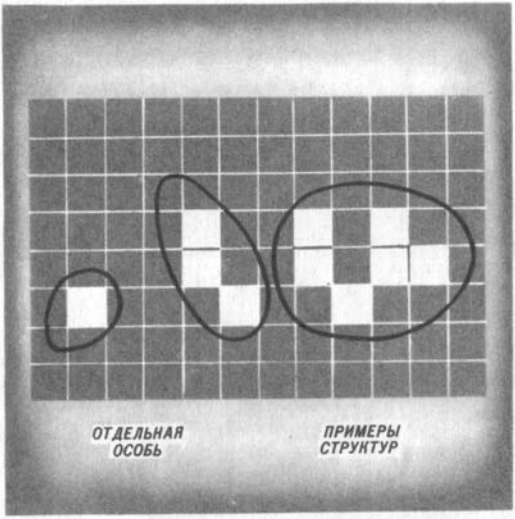
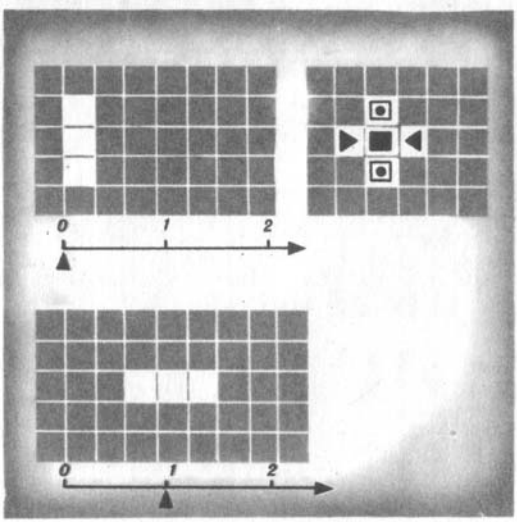


Рис. 43

Рис. 44 а б в



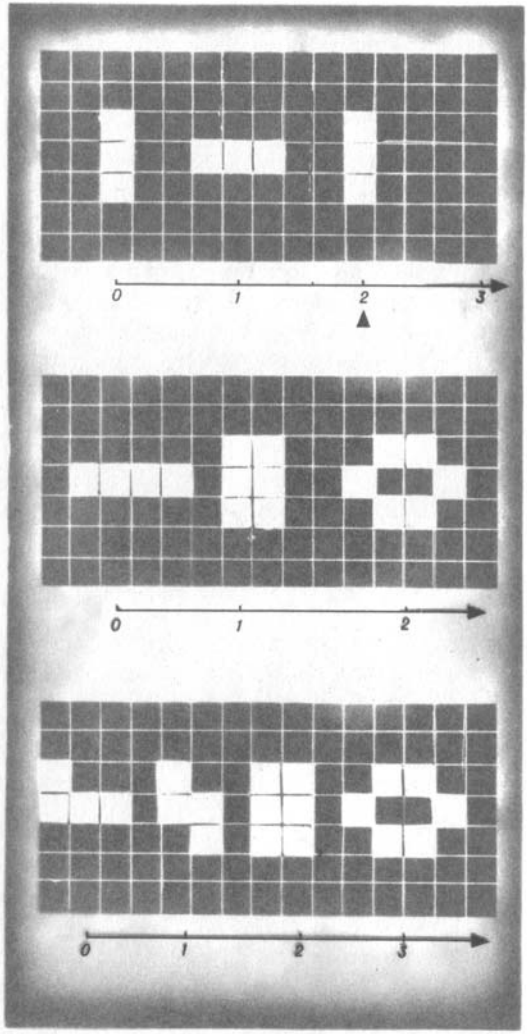


Рис. 45

Рис. 46

Рис. 47

Еще раз представим себе начало эволюции, которая должна протекать в соответствии с законами, сформулированными выше. Пусть начальная структура имеет вид, показанный на рис. 44, *а*.

Эволюционные часы показывают 0. Эволюция началась. Пока стрелка часов готовится к скачку в положение 1, действуют законы эволюции, применяемые к начальной структуре.

Спросим себя, не погибнет ли какая-нибудь особь? Из рисунка видно, что верхняя и нижняя особи исходной структуры должны подготовиться к гибели и погибнуть в тот момент, когда стрелка часов будет показывать 1. Эти особи погибнут потому, что у каждой из них по одному соседу. Чтобы это запомнить, отметим эти особи кружком, так как это показано на рис. 44, *б*. Средняя особь сохранит себя, так как у нее на этом этапе две соседних особи. Наконец, на полях, отмеченных треугольниками, возникнет по одной особи, так как эти пустые поля окружены тремя особями.

Пока мы обдумывали, какая особь погибнет, какая сохранит себя и на какой клетке будет рождена новая особь, время, отведенное на процесс одного такта эволюции, истекло. Стрелка часов перескакивает с 0 на 1 и мгновенно гибнут те особи, которым это «суждено», и появляются те, которым «суждено» появиться. Вот какой будет теперь структура: она, как и исходная, состоит из трех особей, но уже расположенных не вертикально, а горизонтально (рис. 44, *в*).

К этой структуре вновь применяем законы эволюции, отмечая кружком те особи, которые должны погибнуть, и треугольниками — поля, на которых будут возникать новые особи. Ясно, что в момент времени 2 структура будет такой, как показано на рис. 45 справа — структура возвратится в исходное состояние.

Этот очень простой пример разъясняет то, как осуществляется процесс эволюции, раскрывает понятие «такта» эволюции.

Мартин Гарднер пишет: «Иногда первоначальная колония организмов постепенно вымирает, все фишки (особи) исчезают, однако произойти это может не сразу, а лишь после того, как сменится очень много поколений. В большинстве своем исходные конфигурации либо переходят в устойчивые и перестают изменяться, либо навсегда переходят в колебательный режим. Конфигурации, не обладающие в начале игры симметрией, обнаруживают тенденцию к переходу в симметричные формы. Обретенные свойства симметрии в процессе эволюции не утрачиваются, симметрия конфигурации может лишь обогащаться».

Приведем примеры различных эволюций простых начальных структур.

На рис. 46 показано, как очень простая структура на втором такте превратилась в кристаллическую, которая уже измениться не может. Каждая особь в этой структуре имеет две соседних и ни одна пустая клетка не окружена в точности тремя особями.

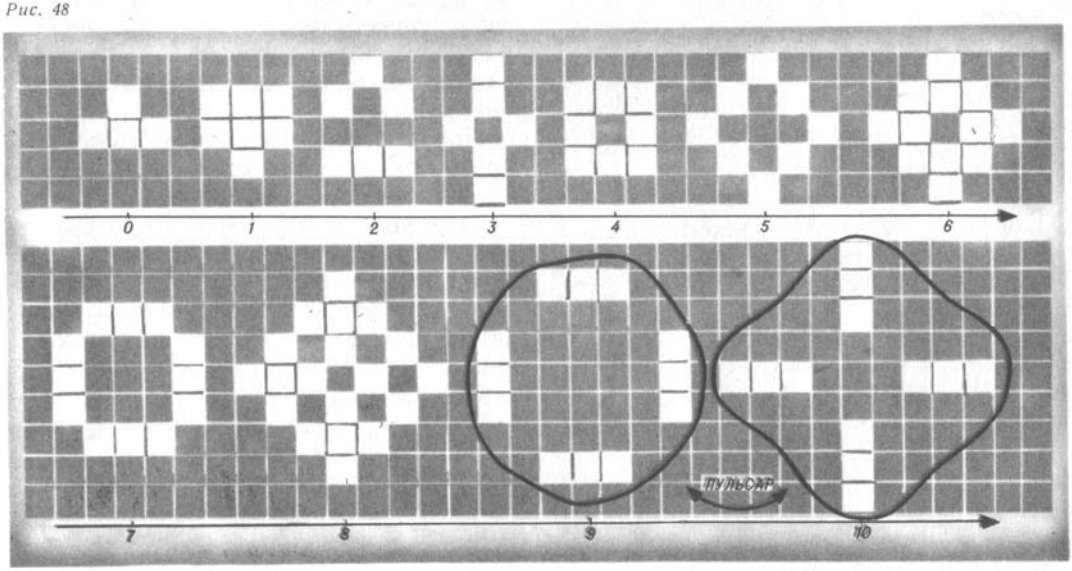
На рис. 47 изображена эволюция другой начальной структуры, которая превращается в такую же кристаллическую структуру, но уже не за два, а за три такта.

На рис. 48 показаны 10 тактов эволюции, которая приводит к появлению еще одной пульсирующей структуры. Любопытно, что столь сложная эволюция начата с очень простой исходной структуры, которая, как и две предшествующие, состояла всего из четырех особей.

Девятый и десятый такты переводят структуру в состояние «пульсара». Структура, возникшая к девятому такту, состоит из четырех элементов, каждый из которых, начиная с девятого такта, бесконечно пульсирует независимо от других. Пульсация осуществляется так же, как и для структуры, изображенной на рис. 45.

Очень интересно эволюционируют начальные структуры, имеющие формы фигур пентамино (фигуры пентамино состоят из пяти особей, связанных меж-

Рис. 48



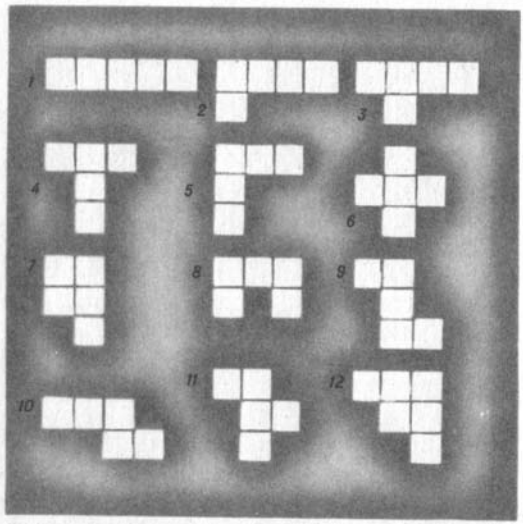


Рис. 49

ду собой так, что их можно обойти ходом ладьи). Фигур пентамино всего 12 (рис. 49).

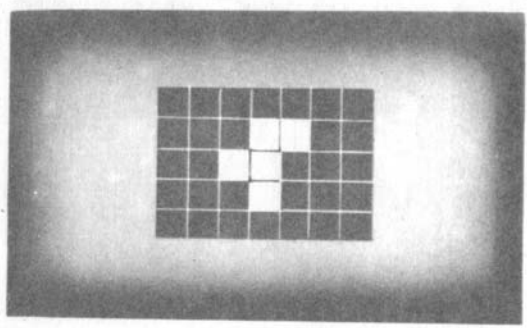
М. Гарднер утверждает, что пять из них гибнут на пятом ходу, две быстро переходят в устойчивые конфигурации из семи особей, а четыре — после небольшого числа тактов превращаются в пульсирующие структуры. Удивительно ведет себя структура, показанная на рис. 50. Ее эволюция была прослежена до 460 такта, когда развивающаяся структура распалась — «от фигуры осталось множество мертвых (не изменяющихся) обломков и лишь несколько областей, в которых еще теплилась жизнь...».

Из сказанного ясно, что идея использовать ЭВМ для слежения за процессом искусственной жизни, за эволюцией, представляется очень увлекательной. Ведь если столь простая начальная структура «живет» почти 500 тактов, то проследить вручную, вырисовывая каждый раз возникающую структуру, за такой эволюцией очень и очень сложно. Машина же может помочь. Моделируя эволюцию, машина должна печатать то, что она «видит» на каждом такте и делать это терпеливо, точно и очень долго. За эволюцией можно наблюдать и на телевизионном экране, которым снабжаются теперь ЭВМ.

После всего сказанного можно перейти к формулировке задачи на программирование:

разработать алгоритм и программу, работая по которой ЭВМ будет моделировать процесс эволюции для конкретных данных. Моделировать — это значит обеспечить применение всех законов эволюции на каждом такте и демонстрировать образовавшиеся структуры на экране дисплея или выдавать их изображение «на печать». При этом эволюция должна осуществляться только на плоском, ранее рассмотренном жизненном пространстве. В каче-

Рис. 50



стве исходных могут быть использованы любые структуры. Что же касается генетических законов, то в каждой конкретной эволюции создаваемый алгоритм должен обеспечить возможность задавать их каждый раз заново. При этом необходимо предусмотреть возможность изменения:

понятия «соседняя особь». Соседней может считаться, например, особь, расположенная в соседнем поле, в которое можно попасть ходом ладьи, соседней может считаться особь, находящаяся в любом из четырех соседних полей по диагонали, соседними могут считаться и особи, расположенные в любом из восьми полей, окружающих рассматриваемую особь;

числа соседей, обеспечивающих выживание рассматриваемой особи;

числа соседей у пустого поля жизненного пространства, вызывающего «возникновение» особи на этом поле.

Разрабатываемый алгоритм должен также включать возможность введения еще одного ограничения на существование особей, которое называется «продолжительность жизни» особи и измеряется в тактах. Особь умирает, если число тактов ее участия в эволюции превышает рубеж естественного старения.

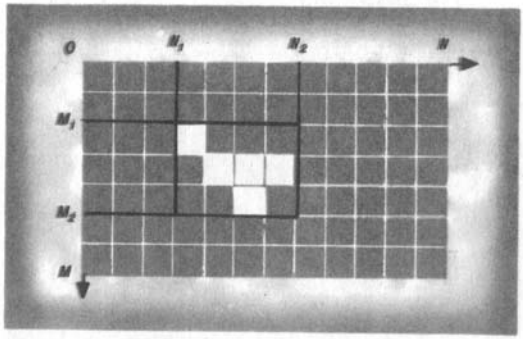
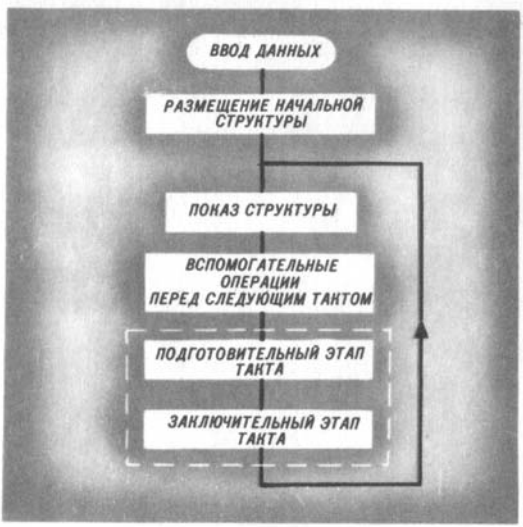


Рис. 51

Рис. 52



АЛГОРИТМ МОДЕЛИРОВАНИЯ ЭВОЛЮЦИИ НА ЭКРАНЕ

Приводим основные понятия и обозначения, необходимые для конструирования алгоритма.

Жизненное пространство представляется в памяти ЭВМ в виде двухмерного массива $A(M, N)$, а при выводе на печать или экран имеет такой вид, как это показано на рис. 51.

Каждая особь, входящая в начальную структуру, задается указанием

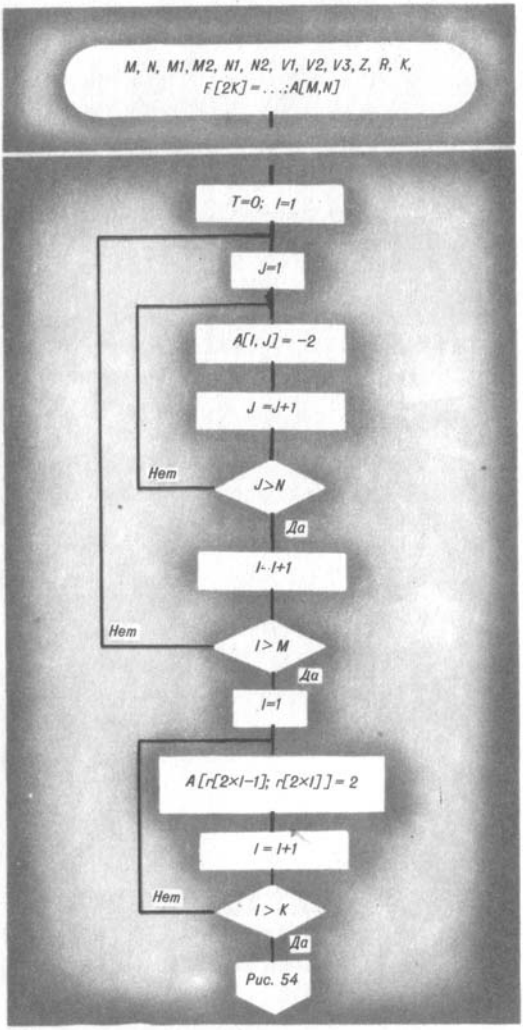


Рис. 53 а б

двух ее координат: $\Gamma(M, N)$. Задать начальную структуру — это значит ввести в память ЭВМ массив $\Gamma[2K]$, где K — число особей, образующих начальную структуру. Числа в массиве Γ рассматриваются пара за парой, каждая такая пара есть координаты конкретной особи.

Алгоритм моделирования эволюции в самом общем виде можно представить в виде последовательности блоков, изображенных на рис. 52.

Ниже назначение каждого из блоков комментируется с использованием более подробных схем. Рассмотрим блок ВВОД ДАННЫХ (рис. 53, *а*).

Перед началом работы алгоритма в память ЭВМ заносятся следующие величины:

M и N — размеры планируемого жизненного пространства, задаваемого в форме прямоугольника;

M_1, M_2, N_1 и N_2 — стороны прямоугольника, в который точно вписывается исходная структура. Этот прямоугольник в дальнейшем будем называть основным;

V_1, V_2 — число соседей, обеспечивающих выживание особи на данном такте;

V_3 — число соседей около пустого поля, необходимое и достаточное для того, чтобы на этом пустом поле к концу такта возникла особь;

R — число тактов, отводимое на эволюцию в целом;

Z — продолжительность жизни особи, измеряемая в тактах;

K — количество особей, образующих исходную структуру;

$F[2K]$ — массив координат особей, образующих исходную структуру;

$A[M, N]$ — массив, задающий размеры жизненного пространства, где $M \geqslant 5$ и $N \geqslant 5$.

Переходим к обсуждению блока РАЗМЕЩЕНИЕ НАЧАЛЬНОЙ СТРУКТУРЫ (рис. 53, *б*). В соответствии со схемой алгоритма в начале

работы всем элементам массива $A[M, N]$ присваивается значение — 2, что означает — поле жизненного пространства пусто. Для этого организуется два цикла: один с параметром I и вложенный в него цикл с параметром J.

Затем в жизненное пространство помещается начальная структура, для чего с помощью еще одного цикла с параметром I всем элементам массива A, координаты которых указаны в массиве $Г$, присваивается значение $+2$.

Рассмотрим в деталях блок ПОКАЗ СТРУКТУРЫ (рис. 54, *а*). Показать структуру — значит напечатать ее на бумаге или вывести на экран. Это осуществляется с помощью организации двух циклов, которые обеспечивают осмотр всех полей основного прямоугольника. Если на каком-либо рассматриваемом поле находится особь, т. е. $A[I, J]=+2$, то вступает в действие оператор вывода символа, которым изображается особь (*), либо на печать, либо на экран. В противном случае выводится символ «пусто» (0).

Обратимся к анализу блока ВСПОМОГАТЕЛЬНЫЕ ОПЕРАЦИИ ПЕРЕД СЛЕДУЮЩИМ ТАКТОМ (рис. 54, *б*). Перед каждым очередным тактом должны быть выполнены следующие вспомогательные операции:

увеличение значения счетчика тактов T на единицу. Счетчик T ведет подсчет числа тактов осуществляемой эволюции;

проверка условия $T>R$ и если оно выполнено, то на печать выводится текст «ВРЕМЯ, ОТВЕДЕННОЕ ДЛЯ ЭВОЛЮЦИИ, ИСТЕКЛО». В противном случае на печать выводится номер подготавливаемого такта;

изменение значений вспомогательных переменных $Л$, $В$, $Н$ и $П$. О назначении и использовании этих переменных сказано ниже. Рассмотрим блок ПОДГОТОВИТЕЛЬНЫЙ ЭТАП ТАКТА. Назначение изображенной на

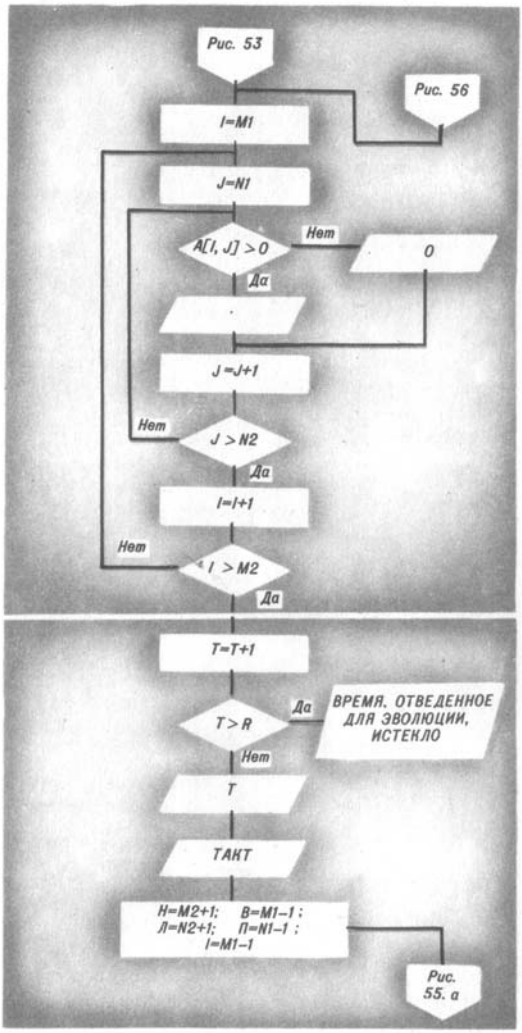


Рис. 54 а б

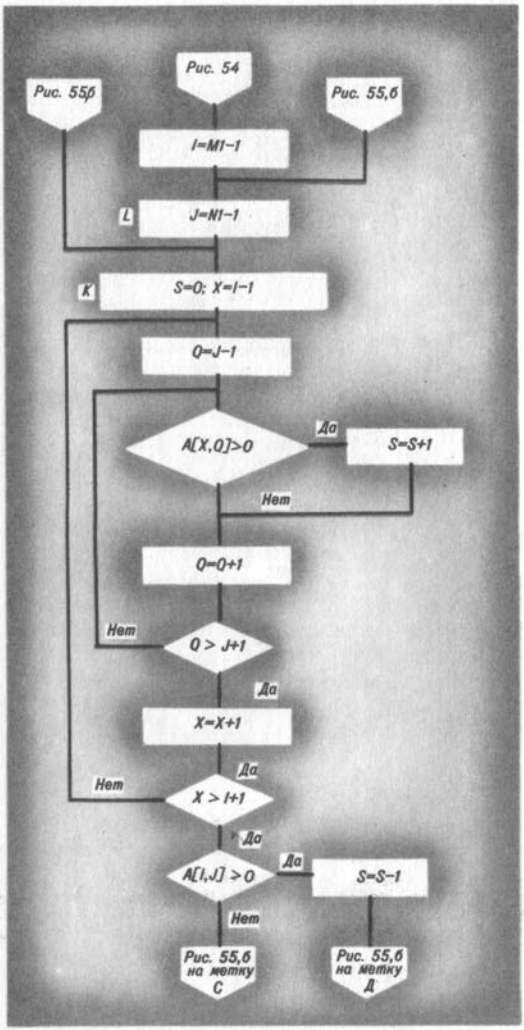


Рис. 54
Рис. 55,б
Рис. 55,б
I=M1-1
L
J=N1-1
K
S=0; X=I-1
Q=J-1
A[X,Q]>0
Да
S=S+1
Нет
Q=Q+1
Нет
Q > J+1
Да
X=X+1
Да
Нет
X > I+1
Да
A[I,J] > 0
Да
S=S-1
Нет
Рис. 55,б на метку С
Рис. 55,б на метку Д

Рис. 55 а б

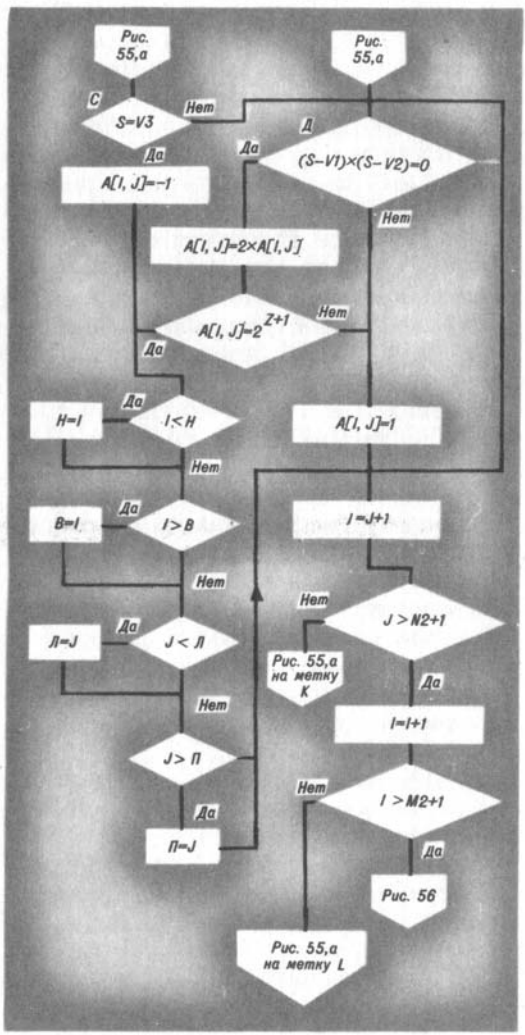


Рис. 55,а
Рис. 55,а
С
S=V3
Нет
Да
Д
Да
(S-V1)×(S-V2)=0
A[I, J]=-1
Нет
A[I, J]=2×A[I, J]
A[I, J]=2^Z+1
Нет
Да
H=I
Да
I<H
A[I, J]=1
Нет
B=I
Да
I>B
I=J+1
Нет
Нет
J > N2+1
Л=J
Да
J < Л
Рис. 55,а на метку К
Да
Нет
I=I+1
J > П
Нет
I > M2+1
Да
П=J
Да
Рис. 56
Рис. 55,а на метку L

рис. 55, *а* части рассматриваемого блока — образовать вспомогательный прямоугольник вокруг основного так, чтобы его размеры были на две единицы больше размера основного; подсчитать последовательно число соседей у всех полей образованного вспомогательного прямоугольника. Это осуществляется с помощью двух циклов, которые позволяют последовательно просмотреть все поля вспомогательного прямоугольника одно за другим. Еще два цикла обеспечивают просмотр всех полей квадрата 3×3, в центре которого расположено рассматриваемое поле вспомогательного прямоугольника.

Работа алгоритма по этой части блока взаимосвязана с работой по схеме части блока, изображенной на рис. 55, *б*. Обращение к схеме на рис. 55, *б* происходит каждый раз, как только будет подсчитано число соседей у очередного осматриваемого поля вспомогательного прямоугольника. Работая в соответствии с этой схемой алгоритма, ЭВМ обеспечит подготовку особи к смерти, для чего присвоит соответствующему элементу массива $A[I, J]$ значение 1. Здесь же осуществляется подготовка пустого поля ($A[I, J]=-2$) к возникновению на нем особи, для этого соответствующему элементу массива A будет присвоено значение — 1. А для тех особей, которые на этом такте выживают (сохраняют себя), соответствующие значения элементов массива A умножаются на 2. В этой же части блока осуществляется проверка условия: не превышает ли число тактов прожитой жизни данной особи числа Z. Это делается с помощью распознавателя $A[I, J]\leqslant 2^{z+1}$. В этом же блоке штрихами обведена группа операторов, выполнение которых обеспечивает вычисление временных границ основного прямоугольника, в который точно вписывается только что созданная структура. Эти границы и обозначены буквами *Л*, *Н*, *В* и *П*.

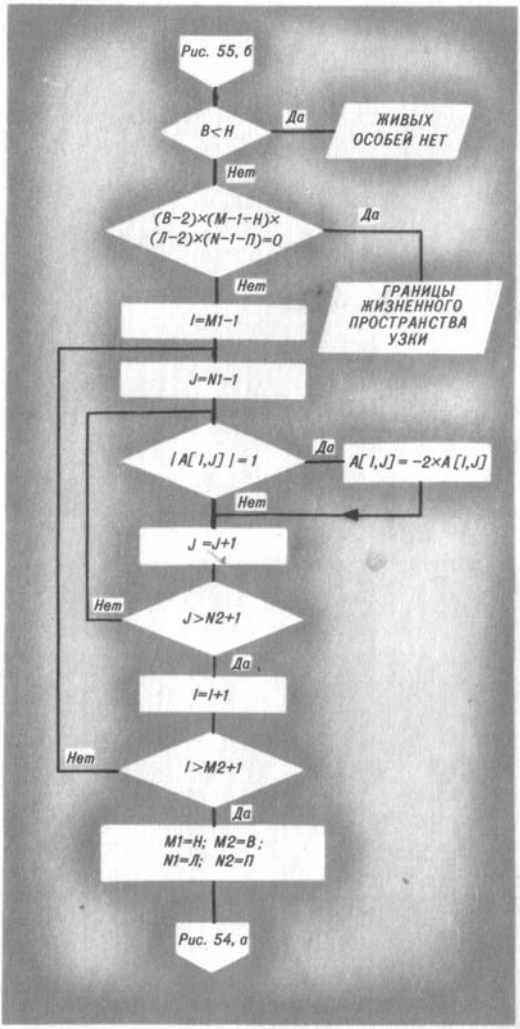


Рис. 56

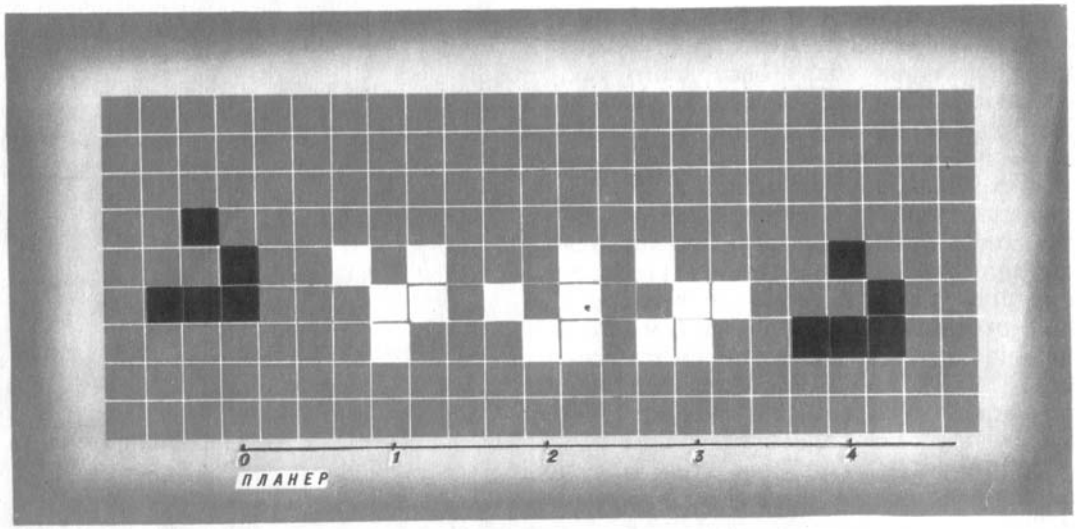
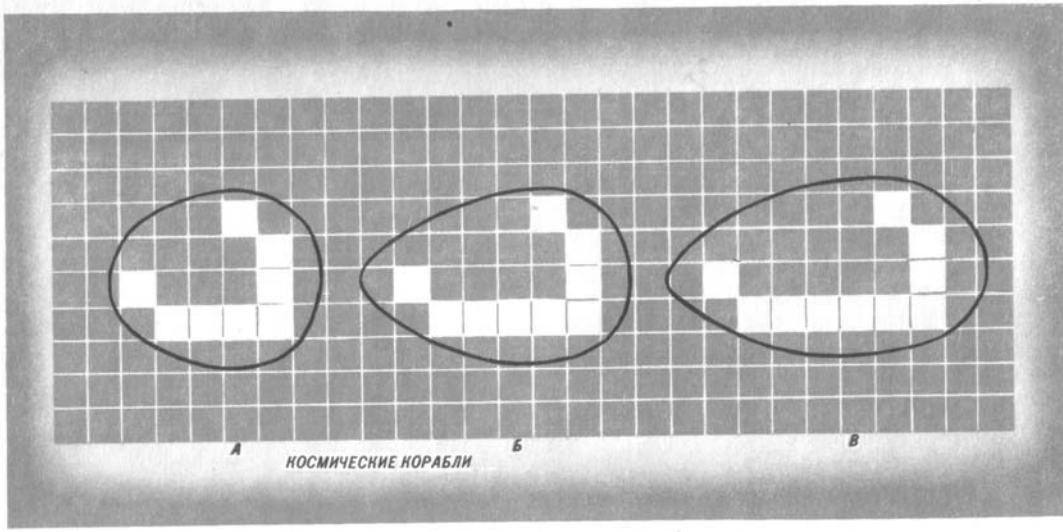


Рис. 57

Рис. 58



Анализируем блок ЗАКЛЮЧИТЕЛЬНЫЙ ЭТАП ТАКТА. Общее назначение блока — удалить все особи, подготовленные к смерти, и поместить в жизненное пространство новорожденные особи. В блоке предусмотрены два сообщения, выдаваемые ЭВМ о ходе эволюции. Одно — «ЖИВЫХ ОСОБЕЙ НЕТ» и второе — «ГРАНИЦЫ ЖИЗНЕННОГО ПРОСТРАНСТВА УЗКИ». После каждого из этих сообщений эволюция прекращается.

Последние операторы блока обеспечивают вычисление размеров основного прямоугольника, в котором помещена структура, возникшая после выполнения рассмотренного такта. Лишь

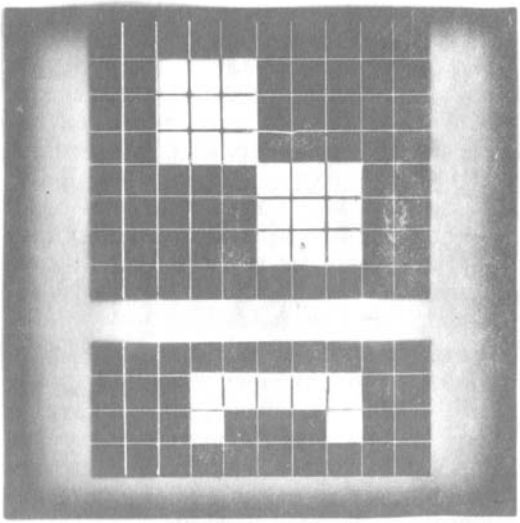
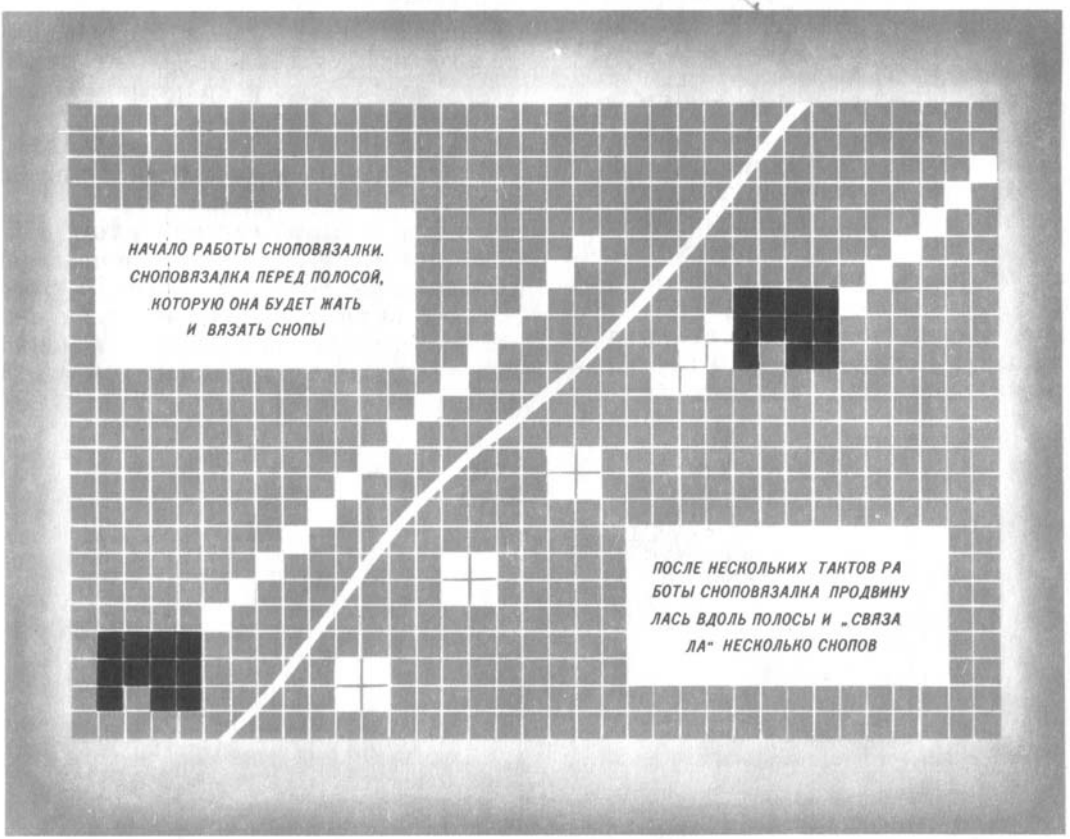


Рис. 59

Рис. 60

Рис. 61 *а б*



после этого начинается новый такт эволюции с участием уже измененной на предшествующем такте структуры. Эволюция будет продолжаться такт за тактом до тех пор, пока число тактов T не превысит отведенного для нее общего числа тактов R, т. е. пока $T<R$, или до тех пор, пока «хватает» отведенного для эволюции жизненного пространства (мы помним, что на всю эволюцию отводилось поле прямоугольной формы размером $M\times N$).

Естественно, что возможность моделирования различных «эволюций» на ЭВМ,— дело чрезвычайно увлекательное. Рекомендуем непременно научить ЭВМ руководству эволюцией и проверить некоторые очень любопытные варианты процесса эволюции.

В работе М. Гарднера, о которой мы упоминали выше, обращается внимание на структуру «Планер», которая изображена на рис. 57.

Если проследить за эволюцией «Планера» в течение четырех тактов, то можно подметить, что «Планер» за эти четыре такта успевает продвинуться вправо вниз по диагонали на одну клетку. Это пример перемещающейся структуры. Поиски таких скользящих структур — дело чрезвычайно сложное. Интересно, что создателю игры «Жизнь» Конуэю известны все четыре таких скользящих по полю изолированных структуры, которые он назвал «космическими кораблями». «Планер» — это космический корабль легчайшего веса, потому что все остальные корабли состоят из большего числа особей (рис. 58).

Любопытно живет структура, изображенная на рис. 59 и получившая наименование «восьмерка». Эта структура относится к пульсирующим. Попробуйте определить, за сколько тактов осуществляется одна пульсация.

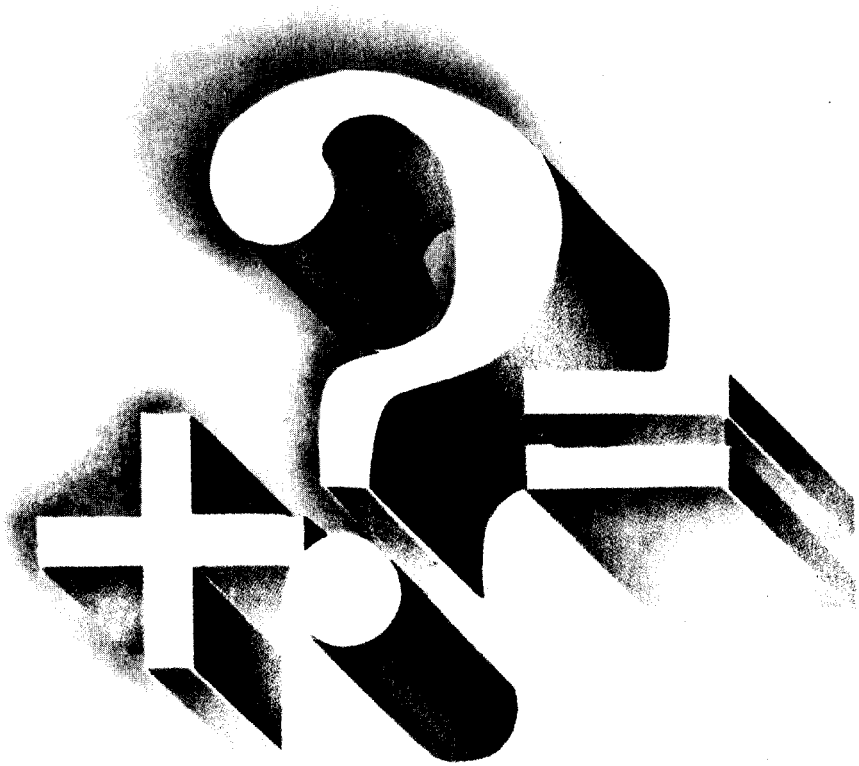
Чрезвычайно интересно, но вручную почти практически невозможно проследить за эволюцией структуры, изображенной на рис. 60.

В заключение остановимся еще на одном виде эволюционирования структур (речь идет о тех же самых законах эволюции). На рис. 61 изображена структура, которую мы называем «Сноповязалка». Эта структура не только движется вдоль рядка слева вверх направо, но и при этом «вяжет снопы» в виде кристаллических структур, состоящих из четырех особей. На рисунке показаны несколько тактов эволюции «Сноповязалки».

Этот небольшой рассказ о том, как ЭВМ помогает моделировать и следить за эволюцией, возможно пробудит у читателей интерес к моделированию на ЭВМ в принципе и, возможно, к моделированию элементов живого, в частности.

ЛОГИЧЕСКОЕ ПРОГРАММИРОВАНИЕ В ЗАНИМАТЕЛЬНЫХ ЗАДАЧАХ

ОТВЕТЫ И РЕШЕНИЯ. ПРИЛОЖЕНИЯ



ОТВЕТЫ
И
РЕШЕНИЯ
ПРИЛОЖЕНИЯ

Программа для игры «Гонки по вертикали

```
«РАЗРЯДНОСТЬ»3.
  0.T=0;P=0;
   «ДЛЯ»I=1«ШАГ»1«ДО»M«ВЫПОЛНИТЬ»
   A[N,I]=0;
   «ДЛЯ»I1=1«ШАГ»1«ДО»(N—1)«ВЫПОЛ
   НИТЬ»
«ДЛЯ»I2=1«ШАГ»1«ДО»M«ВЫПОЛНИТЬ»
   A[I1,I2]=2;S=N;
  I.S=S—1;
   «ЕСЛИ»S⩾0«ТО»(«НА»L);X+M+1;
  2.K=0;X=X—1;
   «ЕСЛИ»X<1«ТО»(«НА»1);R=0;
  3.X1=0;
  4.R=R+1;
   «ЕСЛИ»R=X«ТО»(«НА»4);B=S+R;
   «ЕСЛИ»B>N«ТО»(
  5.A[S,X]=0;«НА»2);
   X1=X1+R;
   «ЕСЛИ»A[B,X1]=0«ТО»(A[S,X]=1,«НА»2);
   K=K+1;
   «ЕСЛИ»K⩾M—1«ТО»(«НА»5);«НА»3;
  L.«ВЫВОД»01,[ИГРАГОНКИПОВЕРТИКАЛ
   И],
   «СТРОКА»,[ВАШСОПЕРНИК — ЭВММИ
   Р],«СТРОКА»;
   S=0;P=0;
  L1.X=1;
  L2.S=S+1;
   «ЕСЛИ»X=P«ТО»(
  L3.X=X+1;«НА»L2);
   «ЕСЛИ»X>M«ТО»(S=0;
   «ВЫВОД»01,[ВАШПЕРВЫЙХОД];«ВВОД»
   34);
   «ЕСЛИ»A[S,X]>0«ТО»(«НА»L3);T=S;
   P=X;
   «ВЫВОД»01,[ДЕЛАЮСВОЙХОД]«СТРОК
   А»,T
   «ПРОБЕЛ»2,P,«СТРОКА»;
   «ЕСЛИ»S=N«ТО»(
  L4.«ВЫВОД»01,[ВЫПРОИГРАЛИ]);
   «ВЫВОД»01,[ВАШ...ХОД];«ВВОД»34;
  L5.«ЕСЛИ»ABS(T—P—S)>0«ТО»(
  L6.«ВЫВОД»01,[ВЫОШИБЛИСЬ];«ВВОД»
   34);
   «ЕСЛИ»P=X«ТО»(«НА»L6);
   «ЕСЛИ»T>N«ТО»(«НА»L4);S=T;«НА»L1
  «ГДЕ»M=m; N=n; A[n, m]
   «КОНЕЦ»◇
```

Инструкция для пользователя

В программе использованы обозначения:
N — количество вертикальных клеток;
M — максимальное число клеток, на которое можно продвинуть вверх шашку за один ход;
A[N,M] — вспомогательный массив, содержащий особые и неособые точки;
P — число клеток, на которое один из соперников продвинул шашку за свой ход (P<M);
T — номер клетки, на которой находится шашка после последнего хода.

Сделать ход для человека, играющего против ЭВМ МИР,— это значит ввести директиву вида

```
«ВЫПОЛНИТЬ»T=...;P=...;«НА»L5»
 «КОН» ◇
```

Сделать ход в игре для ЭВМ — это значит напечатать

```
T=...;P=...;«ВАШХОД»◇
```

Программа для игры «Гонки по вертикали

(программа конкретного поединка между ЭВМ и человеком)

```
«РАЗРЯДНОСТЬ»3;0.T=0;P=0;«ДЛЯ»I=1«Ш
АГ»1«ДО»M«ВЫПОЛНИТЬ»A[N,I]=0;«ДЛЯ»I
1=1«ШАГ»1«ДО»(N—1)«ВЫПОЛНИТЬ»«ДЛ
Я»I2=1«ШАГ»1«ДО»M«ВЫПОЛНИТЬ»A[I1,
I2,]=2;S=N;1.S=S(1;«ЕСЛИ»S<0«ТО»(«Н
А»L);X=M+1;2.K=0;X=X—1;«ЕСЛИ»X<1«Т
О»(«НА»1);R=0;3.X1=0;4.R=R+1;«ЕСЛИ»R
=X«ТО»(«НА»4);B=S+R;«ЕСЛИ»B>N«ТО»
(5.A[S,X]=0;«НА»2);X1=X1+R;«ЕСЛИ»A[B,
X1]=0«ТО»(A[S,X]=1;«НА»2);K=K+1;«ЕСЛ
И»K⩾M—1«ТО»(«НА»5);«НА»3;L.«ВЫВОД»0
1,[ИГРАГОНКИПОВЕРТИКАЛИ],«СТРОКА»,
[ВАШСОПЕРНИК — ЭВММИР],«СТРОКА»;S
=0;P=0;L1.X=1;L2.S=S+1;«ЕСЛИ»X=P«ТО
»(L3.X=X+1;«НА»L2);«ЕСЛИ»X>M«ТО»(S=
0;«ВЫВОД»1,[ВАШ...ПЕРВЫЙ...ХОД];«ВВОД
»34);«ЕСЛИ»A[S,X]>0«ТО»(«НА»L3);T=S;P=
X;«ВЫВОД»01,[ДЕЛАЮСВОЙХОД],«СТРОК
А»,T,«ПРОБЕЛ»2,P,«СТРОКА»;«ЕСЛИ»S=N«
ТО»(L4.«ВЫВОД»1,[ВЫ...ПРОИГРАЛИ]);«В
ЫВОД»1,[ВАШ...ХОД];«ВВОД»34;L5.«ЕСЛИ»
ABS(T—P—S)>0«ТО»(L6.«ВЫВОД»1,[ВЫ...О
ШИБЛИСЬ];«ВВОД»34);«ЕСЛИ»P=X«ТО»(«
НА»L6);«ЕСЛИ»T>N«ТО»(«НА»L4);S=T;«НА
»L1«ГДЕ»M=6;A[15,6];N=15«КОНЕЦ»◇
```

Протокол поединка, проведенного между ЭВМ и человеком

```
ИГРА ГОНКИ ПО ВЕРТИКАЛИ
ВАШ СОПЕРНИК — ЭВМ МИР
ДЕЛАЮ СВОЙ ХОД
T=1 P=1
ВАШ ХОД◇
«ВЫП» T=6; P=5; «НА»L5 «КОН» ◇
ДЕЛАЮ СВОЙ ХОД
T=8 P=2
ВАШ ХОД ◇
«ВЫП» T=14; P=6; «НА»L5 «КОН» ◇
ДЕЛАЮ СВОЙ ХОД
T=15 P=1
ВЫ ПРОИГРАЛИ ◇
```

На рис. П1 изображено игровое поле, на котором проводился поединок, и все ходы, сделанные человеком и ЭВМ.

Машина, начиная игру, поместила шашку на поле номер 1 (T=1, P=1), человек передвинул шашку на пять клеток (T=6, P=5), затем ЭВМ сделала второй ход: T=8, P=2, т. е. передвинула шашку на две клетки и поместила ее в клетку 8, человек ответил ходом (T=14, P=6) и поставил шашку на поле 14, после этого ЭВМ передвинула шашку на одну клетку и выиграла.

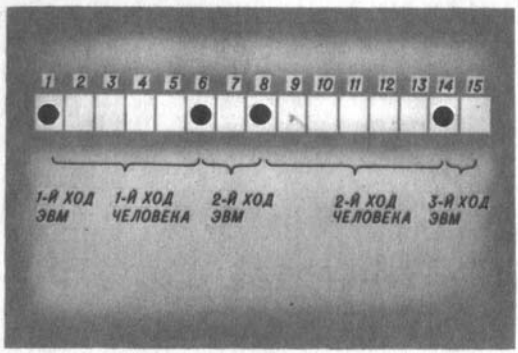


Рис. П 1

Программа для игры «Тригекс»

```
«РАЗРЯДНОСТЬ»6.
  0.«ВЫВОД»01,[ИГРАТРИГЕКС],«СТРОКА»,
   [ВАШСОПЕРНИК — ЭВММИР-2];
   H=0;L=0;N=M;
   «ДЛЯ»I=1«ШАГ»1«ДО»9«ВЫПОЛНИТЬ»
   A[I]=0;
  1.H+1;N=ε(N);
   «ВЫВОД»01«ЗНАЧЕНИЙ»«СТРОКА»,[ДЕ
   ЛАЮСВОЙ],
   «ПРОБЕЛ»2,H,«ПРОБЕЛ»2,[ХОД],«СТРО
   КА»,[N=],N,
   «СТРОКА»,[ВАШХОД];«ВВОД»34;
  2.A[N]=1;A[L]=5;
   «ЕСЛИ»H=1«ТО»(«ЕСЛИ»M=1«ТО»(
   N=3—(A[3]+A[5]+A[6])×.2+A[7]×.6;«НА»
   1);
   «ЕСЛИ»M=2«ТО»(
   N=1+(A[1]+A[7]+A[8])×.4+A[9]×.8«НА»
   1);
   N=2—(A[2]+A[4]+A[9])×.2+A[5]×.8;«НА
   »1);
   I=1;
  3.S=A[P[I,1]]+A[P[I,2]]+A[P[I,3]];
   «ЕСЛИ»(S—2)×(S—10)=0«ТО»(
   «ЕСЛИ»S=2«ТО»(
   «ДЛЯ»J=1«ШАГ»1«ДО»3«ВЫПОЛНИТЬ»(
   N=P[I,J];
   «ЕСЛИ»A[N]=0«ТО»(«НА»5));
  5.H=H+1;N=ε(N)
   «ВЫВОД»01«ЗНАЧЕНИЙ»«СТРОКА»,[ДЕ
   ЛАЮСВОЙ],
   «ПРОБЕЛ»2,H,«ПРОБЕЛ»2,[ХОД],«СТРО
   КА»,[N=],N,
   «СТРОКА»,[ВЫПРОИГРАЛИ];«СТОП»);
   L=0;K=I);I=I+1;
   «ЕСЛИ»I≤9«ТО»(«НА»3);
   «ЕСЛИ»L=0«ТО»(I=K;
   «ДЛЯ»J=1«ШАГ»1«ДО»3«ВЫПОЛНИТЬ»
   (N=P[I,J];
   «ЕСЛИ»A[N]=0«ТО»(«НА»1)));N=1;
  6.«ЕСЛИ»A[N]>0«ТО»(N=N+1;«НА»6);
   A[N]=1;T=0;I=1;
  7.S=A[P[I,1]]+A[P[I,2]]+A[P[I,3]];
   «ЕСЛИ»S=2«ТО»(T=T+1;
   «ЕСЛИ»T=2«ТО»(«НА»1));I=I+1;
   «ЕСЛИ»I>9«ТО»(A[N]=0;N=N+1;«НА»
   6);«НА»7
   «ГДЕ»M=1;P[9,3]=1,2,8,1,3,6,1,4,7,2,6,9,2,3,4,
   3,5,8,5,4,9,5,6,7,7,8,9;A[9]
   «КОНЕЦ» ◇
```

Инструкция для пользователя

В программе использованы обозначения:
N — номер поля, на которое поставила шашку своего цвета ЭВМ;
L — номер поля, на которое ставит шашку соперник — человек.

Сделать ход для человека, играющего против ЭВМ МИР,— это значит ввести директиву вида

«ВЫПОЛНИТЬ»L=...;«НА»2«КОНЕЦ» ◇

Сделав очередной ход в игре, ЭВМ МИР напечатает текст

ДЕЛАЮ СВОЙ ХОД
N=...
ВАШ ХОД ◇

Если играются подряд несколько партий, то возможно изменить первый ход ЭВМ. Для этого необходимо ввести программу с перфоленты, а затем напечатать текст

«ВМЕСТО» М=1 «ЗАПИСАТЬ» М=2◇

или

«ВМЕСТО» М=1 «ЗАПИСАТЬ» М=3◇

после чего пустить ЭВМ на счет.

Программа
для игры
«Тригекс»
(программа
конкретного поединка
между ЭВМ и человеком)

```
«РАЗР»6.0.«ВЫВ»01,[ИГРАТРИГЕКС],«СТРО
»,[ВАШСОПЕРНИК—ЭВММИР-2];H=0;L=0;
N=M;«ДЛЯ»I=1«ШАГ»1«ДО»9«ВЫПО»A[I]
=0;1.H=H+1;N=ε(N);«ВЫВО»01«ЗНАЧ»«С
ТРО»,[ДЕЛАЮСВОЙ],«ПРОБ»2,H,«ПРОБ»2,
[ХОД],«СТРО»,[N=],N,«СТРО»,[ВАШХОД];«В
ВОД»34;2.A[N]=1;A[L]=5;«ЕСЛИ»H=1«ТО»
(«ЕСЛИ»M=1«ТО»(N=3—(A[3]+A[5]+A[6])
×.2+A[7]×.6;«НА»1);«ЕСЛИ»M=2«ТО»(N=1
+(A[1]+A[7]+A[8])×.4+A[9]×.8;«НА»1);N=2
—(A[2]+A[4]+A[9])×.2+A[5]×.8;«НА»1);I=1;
3.S=A[P[I,1]]+A[P[I,2]]+A[P[I,3]];«ЕСЛИ»(S—
2)×(S—10)=0«ТО»(«ЕСЛИ»S=2«ТО»(«ДЛЯ
»J=1«ШАГ»1«ДО»3«ВЫПО»(N=P[I,J];«ЕСЛ
И»A[N]=0«ТО»(«НА»5));5.H=H+I;N=ε(N);
«ВЫВО»01«ЗНАЧ»«СТРО»,[ДЕЛАЮСВОЙ],
«ПРОБ»2,H,«ПРОБ»2,[ХОД],«СТРО»,[N=],N,
«СТРО»,[ВЫПРОИГРАЛИ];«СТОП»);L=0;K=
I);I=I+1;«ЕСЛИ»I≤9«ТО»(«НА»3);«ЕСЛИ»L
=0«ТО»(I=K;«ДЛЯ»J=1«ШАГ»1«ДО»3«ВЫП
О»(N=P[I,J];«ЕСЛИ»A[N]=0«ТО»(«НА»1)));
N=1;6.«ЕСЛИ»A[N]>0«ТО»(N=N+1;«НА»6);
A[N]=1;T=0;I=1;7.S=A[P[I,1]]+A[P[I,2]]+A[P
[I,3]];«ЕСЛИ»S=2«ТО»(T=T+1;«ЕСЛИ»T=2«
ТО»(«НА»1));I=I+1;«ЕСЛИ»I>9«ТО»(A[N]=
0;N=N+1;«НА»6);«НА»7«ГДЕ»M=1;P[9,3]=1,
2,8,1,3,6,1,4,7,2,6,9,2,3,4,3,5,8,5,4,9,5,6,7,7,8,9;A[9]«
КОН»◇
```

Протокол
одного
поединка,
проведенного
между
ЭВМ МИР-2 и человеком

```
ИГРА «ТРИГЕКС»
ВАШ СОПЕРНИК — ЭВМ МИР-2
ДЕЛАЮ СВОЙ 1 ХОД
N=1
ВАШ ХОД◇
«ВЫП»L=3; «НА»2 «КОН»◇
ДЕЛАЮ СВОЙ 2 ХОД
N=2
ВАШ ХОД◇
«ВЫП»L=8; «НА»2 «КОН»◇
ДЕЛАЮ СВОЙ 3 ХОД
N=5
ВАШ ХОД◇
«ВЫП»L=9; «НА»2 «КОН»◇
ДЕЛАЮ СВОЙ 4 ХОД
N=7
ВАШ ХОД◇
«ВЫП»L=4;«НА»2«КОН»◇
ДЕЛАЮ СВОЙ 5 ХОД
N=6
ВЫ ПРОИГРАЛИ◇
```

Программа
для игры
«НИМ»

```
«РАЗРЯДНОСТЬ»2.
  0.«ВЫВОД»01,[ИГРАНИМ],«СТРОКА»,
    [ВАШСОПЕРНИК — ЭВММИР],«СТРОКА»;
    Y=0;P1=1;T=1;
  1.«ЕСЛИ»M[T]>Y«ТО»(Y=M[T]);T=T+1;
    «ЕСЛИ»T≤K«ТО»(«НА»1);N=2;
  2.«ЕСЛИ»Y≥N«ТО»(N=N×2;P1=P1+1;«Н
    А»2);
    LL.T=1;
  3.P=P1;A=M[T];
  4.Q=ε(A/2);L[T,P]=A—2×Q;
    «ЕСЛИ»Q>0«ТО»(A=Q;P=P—1;«НА»4);
  5.«ЕСЛИ»P>1«ТО»(P=P—1;L[T,P]=0;«НА»
    5);T=T+1;
    «ЕСЛИ»T≤K«ТО»(«НА»3);
  6.R=ε(T=1,K,L[T,P];
    «ЕСЛИ»F(R/2)=0«ТО»(P=P+1;
    «ЕСЛИ»P<P1«ТО»(«НА»6);«НА»8);T=1;
  7.«ДЛЯ»P=1«ШАГ»1«ДО»P1«ВЫПОЛНИТ
    Ь»
    (SUM=Σ(I=1,T—1,L[I,P])+Σ(I=T+1,K,L
    [I,P]));
    L[K+1,P]=SUM—ε(SUM/2×2);Л=Σ(I=
    0,P1—1,L[K+1,P1—I]×2↑I);
    «ЕСЛИ»Л≥M[T]«ТО»(T=T+1;«НА»7);M[
    T]=Л;
    «ВЫВОД»01;[ДЕЛАЮСВОЙХОД],«СТРОК
    А»,T,«ПРОБЕЛ»6,
    Л,«СТРОКА»;
    «ЕСЛИ»Σ(I=1,K,M[I])=0«ТО»(
    «ВЫВОД»01,[ВЫПРОИГРАЛИ];«СТОП»);
  8.«ВЫВОД»01,[ВАШХОД];«ВВОД»34;
  9.«ЕСЛИ»M[T]>Л«ТО»(
    «ЕСЛИ»T≤K«ТО»(M[T]=Л;«НА»LL));
    «ВЫВОД»01,[ВАШХОДСДЕЛАННЕПОПР
    АВИЛАМ];
    «ВВОД»34
    «ГДЕ»K=k;M[k]=...;L[k+1,P1]
    «КОНЕЦ»◇
```

Инструкция для пользователя

В программе использованы обозначения:
К — число групп предметов;
М[К] — число предметов в группе с номером К;
L[K+1,P1] — вспомогательный массив, в котором число строк К+1, а число столбцов Р1, где Р1 — число двоичных разрядов в наибольшем числе М[К];
Т — номер группы, из которой взяты предметы за очередной ход;
Л — количество предметов, оставшихся в группе с номером Т после хода одного из играющих.

Сделать ход для человека, играющего против ЭВМ,— это значит ввести директиву вида

«ВЫПОЛНИТЬ» Т=...; Л=...; «НА»9 «КОНЕЦ» ◊

Сделав свой очередной ход в игре, ЭВМ МИР напечатает текст

Т=...; Л=...; «ВАШ ХОД» ◊

Увидев этот текст, человек в группе с номером Т должен взять предметы так, чтобы число оставшихся равнялось Л.

Программа для игры «НИМ»

(программа конкретного поединка между ЭВМ и человеком)

```
«РАЗРЯДНОСТЬ»2;0.«ВЫВОД»01,[ИГРАНИ
М],«СТРОКА»,[ВАШСОПЕРНИК — ЭВММИР
],«СТРОКА»;Y=0;P1=1;T=1;1.«ЕСЛИ»M[T]>
Y«ТО»(Y=M[T]);T=T+1;«ЕСЛИ»T≤K«ТО»(«
НА»1);N=2;2.«ЕСЛИ»Y≥N«ТО»(N=N×2;P1
=P1+1;«НА»2);LL.T=1;3.P=P1;A=M[T];4.Q
ε(A/2);L[T,P]=A—2×Q;«ЕСЛИ»0>0«ТО»(A=
Q;P=P—1;«НА»4);5.«ЕСЛИ»P>1«ТО»(P=P—
1;L[T,P]=0;«НА»5);T=T+1;«ЕСЛИ»T≤K«ТО»
(«НА»3);6.R=Σ(T=1,K,L[T,P]);«ЕСЛИ»F(R/2)
=0«ТО»(P=P+1;«ЕСЛИ»P≤P1«ТО»(«НА»6);
«НА»8);T=1;7.«ДЛЯ»P=1«ШАГ»1«ДО»P1«ВЫ
ПОЛНИТЬ»(SUM=Σ(I=1,T—1,L[I,P])+Σ(I=
T+1,K,L[I,P];L[K+1,P]=SUM—ε(SUM/2)×
2);Л=Σ(I=0,P1—1,L[K+1,P1—I]×2↑ I);«ЕС
ЛИ»Л≥M[T]«ТО»(T=T+1;«НА»7);M[T]=Л;«
ВЫВОД»01,[ДЕЛАЮСВОЙХОД],«СТРОКА»,T,
«ПРОБЕЛ»6,Л,«СТРОКА»;«ЕСЛИ»Σ(I=1,K,M
[I]=0«ТО»(«ВЫВОД»01,[ВЫ...ПРОИГРАЛИ];
«СТОП»);8.«ВЫВОД»01,[ВАШ...ХОД];«ВВОД»
34;9.«ЕСЛИ»M[T]>Л«ТО»(«ЕСЛИ»T≤K«ТО»
(M[T]=Л;«НА»LL));«ВЫВОД»01,[ВАШ...ХОД
...СДЕЛАН...НЕ...ПО...ПРАВИЛАМ];«ВВОД»
34«ГДЕ»M[3]=7,8,4;K=3;L[4,4]«КОНЕЦ» ◊
```

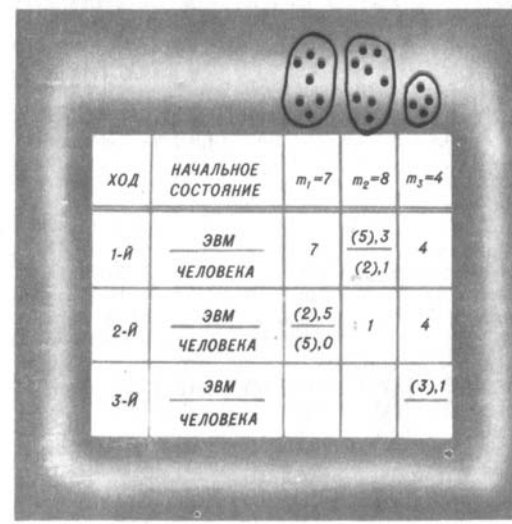
Протокол одного поединка, проведенного между ЭВМ МИР-2 и человеком

```
ИГРА НИМ
ВАШ СОПЕРНИК — ЭВМ МИР
ДЕЛАЮ СВОЙ ХОД
Т=2 Л=3
ВАШ ХОД ◊
«ВЫП» Т=2; Л=1; «НА» 9 «КОН» ◊
ДЕЛАЮ СВОЙ ХОД
Т=1 Л=5
ВАШ ХОД ◊
«ВЫП» Т=1; Л=0; «НА»9 «КОН» ◊
ДЕЛАЮ СВОЙ ХОД
Т=3 Л=1
ВАШ ХОД ◊
«ВЫП» Т=2; Л=0; «НА» 9 «КОН» ◊
ДЕЛАЮ СВОЙ ХОД
Т=3 Л=0
ВЫ ПРОИГРАЛИ ◊
```

На рис. П2 изображено состояние перед началом поединка: имеется три группы, содержащих соответственно 7, 8 и 4 предмета. В таблице показано, из какой группы ЭВМ и человек брали себе предметы на каждом ходе. Число в скобках — число взятых за этот ход предметов, число вне скобок — число предметов, остающихся в группе на данный момент игры.

Первым своим ходом ЭВМ взяла пять предметов из второй группы, она напечатала: Т=2 (вторая группа), Л=3 (оставляю три предмета). Человек, делая свой первый ход, также обратился ко второй группе и взял из нее два предмета, оставив в ней один. Вторым ходом

Рис. П 2



ЭВМ взяла два предмета из первой группы (Т=1, Л=5), отвечая ЭВМ, человек взял из первой группы все предметы (Т=1, Л=0). На третьем ходе (Т=3, Л=1) ЭВМ из третьей группы взяла три предмета, оставила один. Перед третьим ходом человека имеется две группы: вторая и третья, в каждой из которых по одному предмету, и очевидно, что человек проигрывает.

Программа для решения логических задач

```
«РАЗРЯДНОСТЬ»2.
  0.«ДЛЯ»A=0«ШАГ»1«ДО»I«ВЫПОЛНИТЬ»
    «ДЛЯ»B=0«ШАГ»1«ДО»I«ВЫПОЛНИТЬ»
    «ДЛЯ»C=0«ШАГ»1«ДО»I«ВЫПОЛНИТЬ»
    (X=F(A,B,C);
    «ЕСЛИ»X=1«ТО»(
«ВЫВОД»A,«ПР»2,B,«ПР»2,C,«СТРОКА))
«ГДЕ»F(X,Y,Z)=...
    «КОНЕЦ»
```

Инструкция по применению программы

1. Программа, приведенная выше, обеспечивает решение задач, если в формуле, задающей все логические условия, содержится не более трех переменных. В программе эти переменные обозначены буквами A, B и C.

Если в какой-либо конкретной задаче логических переменных окажется больше, то число циклов, аналогичных тем, которые записаны во второй — четвертой строках программы, соответственно должно быть увеличено.

2. Логическая функция, составленная по условиям задачи, должна быть предварительно вручную преобразована так, чтобы каждая логическая операция была заменена моделирующей ее арифметической функцией.

Описание полученной арифметической функции F(X,Y,...,Z) дается в описательной части текста программы.

Пример решения логической задачи

Логическая задача «КТО ЕСТЬ КТО?»

Убийство совершено одиночкой. Подозреваются трое: Браун, Джонс и Смит. Каждый из них сделал по два заявления:

Браун: «Я не убивал и Джонс не убивал»,
Джонс: «Браун не убивал, это сделал Смит»,
Смит: «Я не убивал, это сделал Браун».

Позже выяснилось, что один из подозреваемых оба раза сказал правду, кто-то оба раза солгал и один из них один раз сказал правду, а другой — солгал. Спрашивается «КТО ЕСТЬ КТО?»

Кто убийца?

Как звали человека, оба раза сказавшего правду?

Как звали человека, оба раза солгавшего?

Как звали человека, который один раз сказал правду, а в другой — солгал?

Решение начинаем с того, что вводим обозначения отдельных высказываний:

Б — «Браун убийца»
Д — «Джонс убийца»
С — «Смит убийца»
$\overline{Б}$ — «Браун не убийца»
$\overline{Д}$ — «Джонс не убийца»
$\overline{С}$ — «Смит не убийца»

Составляем формулы сложных высказываний, соответствующих показаниям подозреваемых:

показания Брауна: «Я не убивал и Джонс не убивал» — $Б\overline{Д}$;

показания Джонса: «Браун не убивал, это сделал Смит» — $\overline{Б}С$;

показания Смита: «Я не убивал, это сделал Браун» — $Б\overline{С}$.

Каждое из полученных высказываний — есть логическое произведение и только одно из них истинно, так как в него входят два истинных сомножителя и два других ложны. Одно из произведений ложно потому, что оба сомножителя ложны, а другое потому, что в него входит только один истинный сомножитель. Отсюда следует, что возможны три случая распределения истинности сложных высказываний, что и показано в таблице:

$\overline{Б}\overline{Д}$	1	0	0
$\overline{Б}С$	0	1	0
$Б\overline{С}$	0	0	1

Располагая этой таблицей, составляем три логических формулы:

$$\begin{cases}\overline{Б}\overline{Д}=1\\ \overline{Б}С=0\\ Б\overline{С}=0\end{cases}\text{ или }\begin{cases}\overline{Б}\overline{Д}=1\\ Б+\overline{С}=1\\ \overline{Б}+С=1\end{cases}\text{ или }\overline{Б}\overline{Д}(Б+\overline{С})(\overline{Б}+С)=1;$$

$$\begin{cases}\overline{Б}\overline{Д}=0\\ \overline{Б}С=1\\ Б\overline{С}=0\end{cases}\text{ или }\begin{cases}\overline{Б}\overline{Д}=1\\ \overline{Б}С=1\\ Б+С=1\end{cases}\text{ или }\overline{Б}С(Б+Д)(\overline{Б}+С)=1;$$

$$\begin{cases}\overline{Б}\overline{Д}=0\\ \overline{Б}С=0\\ Б\overline{С}=1\end{cases}\text{ или }\begin{cases}\overline{Б}\overline{Д}=1\\ Б+Д=1\\ Б+\overline{С}=1\end{cases}\text{ или }Б\overline{С}(Б+\overline{С})(\overline{Б}+\overline{Д})=1.$$

Только одна из этих формул при некоторых значениях переменных Б, Д и С даст значение, равное единице: это может быть любая из них.

Отсюда следует, что формула, связывающая все логические условия, имеет вид:

$$\bar{Б}\bar{Д}(Б+\bar{С})(\bar{Б}+С)+\bar{Б}С(Б+Д)(\bar{Б}+С)+$$
$$+Б\bar{С}(Б+Д)(Б+\bar{С})=1 \qquad (1)$$

Необходимо обязательно учесть то, что убийца был одинок, что задается условием

$$Б\bar{С}\bar{Д}+\bar{Б}С\bar{Д}+\bar{Б}\bar{С}Д=1. \qquad (2)$$

После этого можно записать формулу F(Б,С,Д), связывающую уже все логические условия задачи, для чего логически перемножаются формулы (1) и (2). После перемножения и элементарных упрощений имеем

$$F(Б,С,Д)=(\bar{Б}\bar{Д}\bar{С}+\bar{Б}СД+Б\bar{С})\times$$
$$\times(Б\bar{С}\bar{Д}+\bar{Б}С\bar{Д}+\bar{Б}\bar{С}Д). \qquad (3)$$

Последний этап, проводимый на бумаге, состоит в том, что формула (3) заменяется ее арифметической моделью.

Ниже приводится конкретная программа, по которой машина решила задачу:

«РАЗРЯДНОСТЬ»2.0.«ДЛ»Б=0«Ш»1«ДО»1«ВЫП»«ДЛ»Д=0«Ш»1«ДО»1«ВЫП»«ДЛ»С=0«Ш»1«ДО»1«ВЫП»(X=F(Б,Д,С);«Е»X=1«ТО»(ВЫВ»Б,«ПРОБЕЛ»2,Д,«ПРОБЕЛ»2,С,«СТР»))
«ГДЕ»F(X,Y,Z)=(((1—X)×(1—Y)×(1—Z+X×Z)×(1—X+X×Z)+(X+Y—X×Y)×(1—X)×(1—X+X×Z)×Z+(X+Y—X×Y)×(1—Z+X×Z)×(1—Z)×X)×(1—Y)×(1—Z)×X+Y×(1—X)×(1—Z)+Z×(1—X)×(1—Y)—X×Y×(1—Y)×(1—X)×(1—Z)↑2—(1—X)↑2×(1—Z)(1—Y)×Y×Z—X×Z×(1—X)×(1—Z)×(1—Y)↑2)«КОН» ◇

Ответ машина напечатает в виде: Б=1, С=0, Д=0, т. е. «Убийца Браун».

Программа для вычисления «Расстояний» между двумя датами и дней недели

«РАЗРЯДНОСТЬ»6.
0.N=1
1.«ЕСЛИ»M[N]<2«ТО»(
I[N]=365×Г[N]+Д[N]+31×(M[N]—1)+ε(0.25×(Г[N]—1))—ε(0.75×ε((Г[N]—1)×0.01)+1));«НА»2);
I[N]=365×Г[N]+Д[N]+31×(M[N]—1)—ε(0.4×M[N]+2.3)+ε(0.25×Г[N])—ε(0.75×(ε(0.01×Г[N]+1));
2.ДН[N]=I[N]+ε(—I[N]/7)×7+6;
N=N+1;
«ЕСЛИ» N<2«ТО»(«НА»1);
S=I[2]—I[1];
«ВЫВОД»S,«МАССИВА»ДН
«ГДЕ»Г[2]=...;M[2]=...;Д[2]=...;I[2];ДН[2]
«КОНЕЦ» ◇

Инструкция для пользователя

В программе использованы обозначения:
Г[2] — массив, элементами которого являются год более ранней и более поздней даты;
М[2] — массив, элементами которого являются месяц более ранней и более поздней даты;
Д[2] — массив, элементами которого являются число месяца более ранней даты и число месяца более поздней даты;
ДН[2] — рабочий массив, элементами которого по окончании работы будут дни недели, соответствующие первой и второй датам;
I[2] — рабочий массив, элементами которого являются индексы дат.

Подготовить программу для работы — это значит ввести программу с перфоленты и допечатать следующие данные:
Г[2]=...;M[2]=...;Д[2]=...;I[2];ДН[2]«КОНЕЦ» ◇

Пример. Необходимо вычислить расстояние между 12 апреля 1968 г. и 15 июня 1978 г. и определить дни недели каждой из этих дат.

«РАЗР»6....«ГДЕ»Г[2]=1968,1978;M[2]=4,6;Д[2]=12,15;I[2];ДН[2]«КОНЕЦ» ◇

Результатом работы машины по этой программе будет выдача текста

Текст описательной части программы к примеру

S=3716
ДН[2]
5 4

что означает следующее: расстояние между 12 апреля 1968 г. и 15 июня 1978 г.— 3716 дней; 12 апреля 1968 г. была пятница, а 15 июня 1978 г.— четверг.

Программа обучения распознаванию карточек

«РАЗРЯДНОСТЬ»2.
0.T=K;S=2;
1.«ДЛЯ»I=1«ШАГ»1«ДО»K«ВЫПОЛНИТЬ»(«ЕСЛИ»ABS(X[I]—X[T+I])>0«ТО»(X[I]=5));
S=S+1;
«ЕСЛИ»S≤N«ТО»(T=T+K;«НА»1);
«ВЫВОД»[ЭТАПОБУЧЕНИЯЗАВЕРШЕН],«СТРОКА»,
[ЯГОТОВАОТВЕТИТЬНАВОПРОС,ЯВЛЯЕТСЯЛИКАРТАНАШЕЙ];«ВВОД»34;
2.«ДЛЯ»I=1«ШАГ»1«ДО»K«ВЫПОЛНИТЬ»(«ЕСЛИ»ABS(X[I]—X1[I])=1«ТО»(«ВЫВОД»[НЕНАША];«СТОП»);L.);
«ВЫВОД»[НАША]
«ГДЕ»N=...;K=...;X[n×k]=...;X1[K]
«КОНЕЦ» ◇

Инструкция для пользователя

В программе использованы обозначения:
N — количество карт обучающего массива;
К — количество клеток в карте, что позволяет использовать карты с различным количеством клеток;
X[N×K] — массив, элементами которого являются коды каждой карты обучающего массива, перечисленные подряд;
X1[K] — рабочий массив; резервирует место в памяти машины для кодирования исследуемой карты, причем в машину вводится либо каждый элемент массива, либо только те элементы, которые отличаются от элементов предшествующей исследуемой карты.

Результатом работы машины по программе будет выдача текста «ЭТАП ОБУЧЕНИЯ ЗАВЕРШЕН. Я ГОТОВА ОТВЕТИТЬ НА ВОПРОС ЯВЛЯЕТСЯ ЛИ КАРТА НАШЕЙ».

Ввести карту для распознавания — это значит ввести директиву вида

«ВЫПОЛНИТЬ»A=A;«НА»2«ГДЕ»X1[1К]=...«КОНЕЦ»◇

Результатом работы машины на этапе распознавания будет выдача текстов «НАША» или «НЕ НАША».

Примеры распознавания карточек

Ниже приведено пять примеров работы машины по следующей программе обучения распознаванию:

```
«РАЗР»2.0.T=K;S=2;1.«ДЛ»I=1«Ш»1«ДО»K
«ВЫП»(«Е»ABS(X[I]—X[T+I])>0«ТО»(X[I]
=5));S=S+1;«Е»S≤N«ТО»(T=T+K;«НА»1);
«ВЫВ»[ЭТАПОБУЧЕНИЯЗАВЕРШЕН],«СТР
»,[ЯГОТОВАОТВЕТИТЬНАВОПРОС,ЯВЛЯЕТ
СЯЛИКАРТАНАШЕЙ];«ВВОД»34;2.«ДЛ»I=1
«Ш»1«ДО»K«ВЫП»(«Е»ABS(X[I]—X1[I])=1«
ТО»(«ВЫВ»[НЕНАША];«СТОП»);L.);«ВЫВ»[
НАША]«ГДЕ»N=7;K=12;X[84]=0,1,1,0,0,1,0,1,
1,0,1,0,0,0,1,0,0,1,1,1,1,0,0,1,0,1,1,1,1,0,1,0,1,1,0,
0,1,0,0,1,0,1,1,0,1,0,1,1,1,0,0,1,0,1,1,1,0,0,0,1,1,1,1,1,1,
1,0,1,1,1,0,0,0,1,0,0,1,1,1,1,1,0,1,1;X1[12]«КОН»◇
```

На рис. П3 изображено семь карточек обучающего массива

```
ЭТАПОБУЧЕНИЯЗАВЕРШЕН
ЯГОТОВАОТВЕТИТЬНАВОПРОС,ЯВЛЯЕТСЯ
ЛИКАРТАНАШЕЙ◇
«ВЫП»A=A;«НА»2«ГДЕ»X1[12]=1,0,1,0,0,1,0,1,
0,1,0,0«КОН»◇
НЕНАША◇
«ВЫП»A=A;«НА»2«ГДЕ»X1[12]=1,1,1,1,1,1,0,
0,1,0,0,1«КОН»◇
НАША◇
«ВЫП»A=A;«НА»2«ГДЕ»X1[12]=0,0,0,0,1,1,0,
0,1,0,1,0,«КОН»◇
НЕНАША◇
«ВЫП»A=A;«НА»2«ГДЕ»X1[12]=0,0,1,0,0,1,0,
0,1,0,0,0«КОН»◇
НАША◇
«ВЫП»A=A;«НА»2«ГДЕ»X1[12]=1,0,1,0,0,0,1,
1,1,1,1,0«КОН»◇
НЕНАША◇
```

В приведенных примерах ЭВМ были предъявлены для распознавания карточки, изображенные на рис. П4, и она назвала «нашими» карточки 2 и 4.

Программа моделирования процесса эволюции на ЭВМ МИР-2

```
«РАЗРЯДНОСТЬ»6.
 0.«ВЫВОД»[ЭВОЛЮЦИЯ];«СТРОКА»;T=0;
  «ДЛЯ»I=1«ШАГ»1«ДО»M«ВЫПОЛНИТЬ»
  «ДЛЯ»J=1«ШАГ»1«ДО»N«ВЫПОЛНИТЬ»
  A[I,J]=—2;
  «ДЛЯ»I=1«ШАГ»1«ДО»K«ВЫПОЛНИТЬ»
  A[Г[2×I—1],Г[2×I]]=2;
 1.«ДЛЯ»I=M1«ШАГ»1«ДО»M2«ВЫПОЛНИ
  ТЬ»(
  «ВЫВОД»«СТРОКА»;
  «ДЛЯ»J=N1«ШАГ»1«ДО»N2«ВЫПОЛНИ
  ТЬ»
  «ЕСЛИ»A[I,J]>0«ТО»(«ВЫВОД»[0])
  «ИНАЧЕ»(«ВЫВОД»[.]));T=T+1;
  «ЕСЛИ»T>R«ТО»(«СТОП»);
  «ВЫВОД»«ЗНАЧЕНИЙ»«СТРОКА»2,T,«П
  РОБЕЛ»2,[ТАКТ];
  H=M2+1;B=M1—1;Л=N2+1;П=N1—1;
  «ДЛЯ»I=M1—1«ШАГ»1«ДО»M2+1«ВЫП
  ОЛНИТЬ»(
  «ДЛЯ»J=N1—1«ШАГ»1«ДО»N2+1«ВЫПО
  ЛНИТЬ»(S=0;
  «ДЛЯ»X=I—1«ШАГ»1«ДО»I+1«ВЫПОЛ
  НИТЬ»(
  «ДЛЯ»Q=J—1«ШАГ»1«ДО»J+1«ВЫПОЛ
  НИТЬ»(
  «ЕСЛИ»A[X,Q]>0«ТО»(S=S+1)));
  «ЕСЛИ»A[I,J]>0«ТО»(S=S—1;
 «ЕСЛИ»(S—V1)×(S—V2)=0«ТО»(A[I,J]=2
  ×A[I,J];
  «ЕСЛИ»A[I,J]≤2↑(Z+1)«ТО»(«НА»2));
  A[I,J]=1;«НА»3);
  «ЕСЛИ»S=V3«ТО»(A[I,J]=—1;
 2.«ЕСЛИ»I<H«ТО»(H=I);
  «ЕСЛИ»I>B«ТО»(B=I);
  «ЕСЛИ»J<Л«ТО»(Л=J);
  «ЕСЛИ»J>П«ТО»(П=J););3.));
  «ЕСЛИ»B<H«ТО»(
  «ВЫВОД»«СТРОКА»,[ЖИВЫХ ОСОБЕЙ
  НЕТ];«СТОП»);
  «ЕСЛИ»(B—2)×(M—1—H)×(Л—2)×(N—
  1—П)=0«ТО»(
  «ВЫВОД»«СТРОКА»,[ГРАНИЦЫЖИЗНЕ
  ННОГОПРОСТРАНСТВАУЗКИ];«СТОП»);
  «ДЛЯ»I=M1—1«ШАГ»1«ДО»M2+1«ВЫП
```

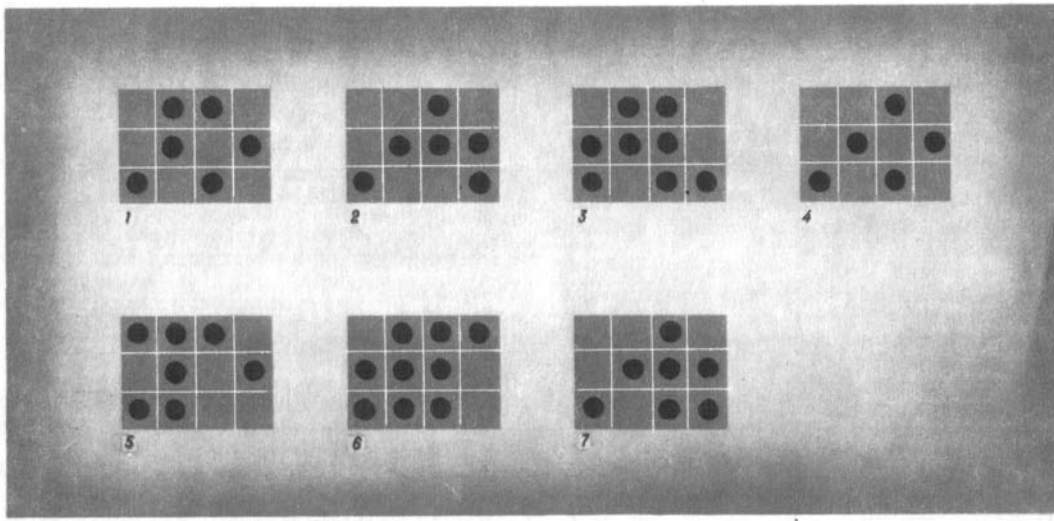


Рис. П 3

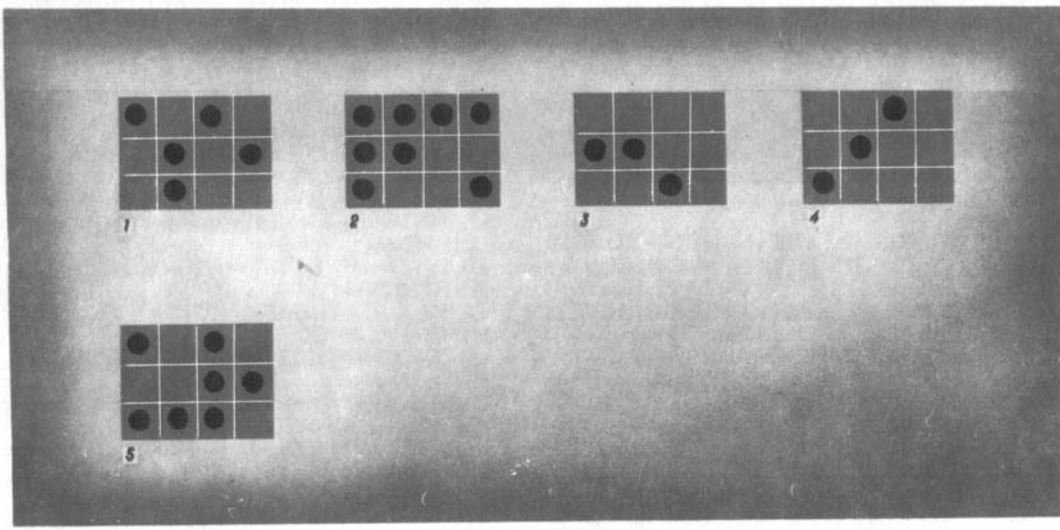


Рис. П 4

```
ОЛНИТЬ»(
«ДЛЯ»J=N1—1«ШАГ»1«ДО»N2+1«ВЫПО
ЛНИТЬ»(
«ЕСЛИ»ABS(A[I, J])=1«ТО»(A[I,J]=—2×
A[I,J]);));
M1=Н;M2=В;N1=Л;N2=П;«НА»1
«ГДЕ»M=...;N=...;M1=...;M2=...;N1=...;N
2=...;
V1=...;V2=...;V3=...;Z=...;R=...;K=...;
Г[2×K]=...;A[M,N]
«КОНЕЦ»
```

Инструкция для пользователя

В программе использованы обозначения:
M, N — размеры жизненного пространства (M — количество строк,
N — количество столбцов);
M1, M2, N1, N2 — координаты вершин прямоугольника, в который точно вписывается начальная структура. M1 — номер верхней горизонтальной границы, M2 — номер нижней горизонтальной границы, N1 — номер левой вертикальной границы, N2 — номер правой верти-

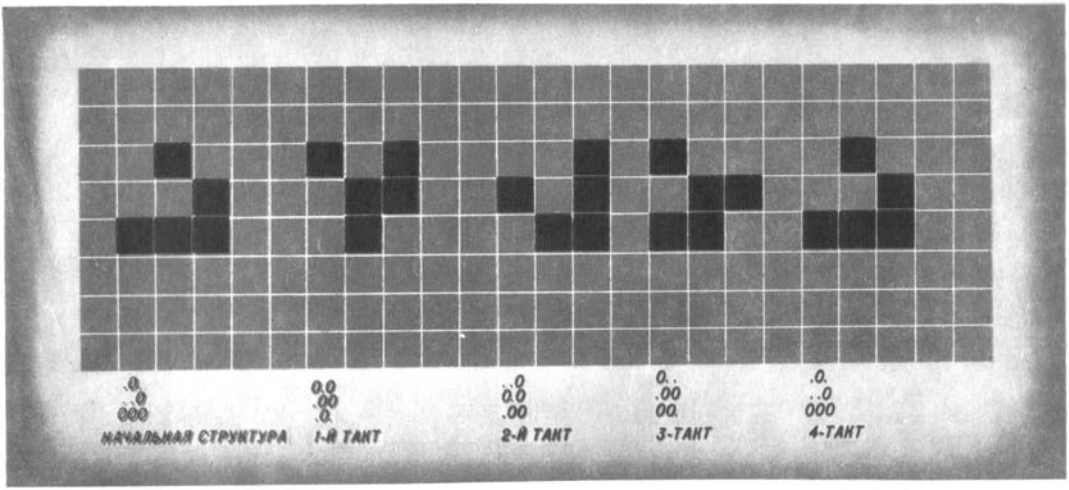


Рис. П 5

кальной границы. Иначе говоря: M1, N1 — координаты левой верхней вершины прямоугольника, M1, N2 — координаты верхней правой вершины, M2, N1 — координаты нижней левой вершины и M2, N2 — координаты нижней правой вершины;

V1,V2 — числа соседних особей, необходимые для выживания особи на данном такте;

V3 — число соседей около пустого поля пространства, необходимое для рождения на нем новой особи;

Z — максимальная продолжительность жизни особи в тактах;

R — число тактов, отведенное на весь процесс эволюции;

K — количество особей в начальной структуре;

Г[2×K] — массив, задающий начальную структуру, в котором каждая пара чисел есть координаты особей начальной структуры;

A[M,N] — массив, задающий жизненное пространство.

Начать моделирование процесса эволюции — это значит внести конкретные данные и пустить программу.

После этого ЭВМ будет на каждом такте выдавать изображение возникающих структур либо «на печать» либо «на экран» в зависимости от использованного режима «ВЫВОД» в конкретной программе.

Программа моделирования конкретной эволюции на ЭВМ МИР-2

```
«РАЗР»6.0.«ВЫВ»[ЭВОЛЮЦИЯ],«СТР»;Т=0;
«ДЛ»I=1«Ш»1«ДО»M«ВЫП»«ДЛ»J=1«Ш»1
«ДО»N«ВЫП»A[I,J]:=—2;«ДЛ»I=1«Ш»1«ДО»
K«ВЫП»A[Г[2×I—1],Г[2×I]]=2;1.«ДЛ»I=M1
«Ш»1«ДО»M2«ВЫП»(«ВЫВ»«СТР»;«ДЛ»J=N
1«Ш»1«ДО»N2«ВЫП»«Е»A[I,J]>0«ТО»(«ВЫВ
»[0])«ИНАЧ»(«ВЫВ»[.]));Т=Т+1;«Е»Т>R«ТО
»(«СТОП»); «ВЫВ» «ЗНАЧ»«СТР»2,Т,«ПРОБ
»2,[ТАКТ];Н=M2+1;В=M1—1;Л=N2+1;П=
N1—1;«ДЛ»I=M1—1«Ш»1«ДО»M2+1«ВЫП»
(«ДЛ»J=N1—1«Ш»1«ДО»N2+1«ВЫП»(S=0;
«ДЛ»X=I—1«Ш»1«ДО»I+1«ВЫП»(«ДЛ»
Q=J—1«Ш»1«ДО»J+1«ВЫП»(«Е»A[X,Q]>0«Т
О»( S=S+1)));«Е»A[I,J]>0«ТО»(S=S—1;«Е»
(S—V1)×(S—V2)=0«ТО»(A[I,J]=2×A[I,J];«
Е»A[I,J]<2↑(Z+1)«ТО»(«НА»2));A[I,J]=1;«
НА»3);«Е»S=V3«ТО»(A[I,J]=—1;2.«Е»I<H«Т
О»(H=I);«Е»I>B«ТО»(B=I);«Е»J<Л«ТО»(Л
=J;«Е»J>П«ТО»(П=J););3.));«Е»B<H«ТО»(«
ВЫВ»«СТР»,[ЖИВЫХОСОБЕЙНЕТ];«СТОП»
);«ЕСЛ»(В—2)×(М—1—Н)×(Л—2)×(N—1—
П)=0«ТО»(«ВЫВ»«СТР»,[ГРАНИЦЫЖИЗНЕ
ННОГОПРОСТРАНСТВАУЗКИ];«СТОП»);«Д
Л»I=M1—1«Ш»1«ДО»M2+1«ВЫП»(«ДЛ»J=
N1—1«Ш»1«ДО»N2+1«ВЫП»(«ЕСЛ»ABS(A[I,
J])=1«ТО»(A[I,J]=—2×A[I,J]);));M1=H;M2=
B;N1=Л;N2=П;«НА»1«ГДЕ»M=15;N=15;M1
=4;M2=6;N1=4;N2=6;V1=2;V2=3;V3=3;Z
=6;R=5;K=5;Г[10]=4,5,5,6,6,4,6,5,6,6;A[15,15]
«КОН» ◇
```

По приведенной конкретной программе машина обеспечит эволюцию только заданной на рис. П5 структуры из пяти особей в соответствии с законами эволюции, описанными выше.

Ход эволюции, изображения возникающих на каждом такте структур демонстрируется на экране, которым снабжена ЭВМ МИР-2, или печатаются на бумаге.

На рис. П5 показано, какой текст печатается на бумаге и каким структурам он соответствует.

Валентин Николаевич КАСАТКИН

ЛОГИЧЕСКОЕ ПРОГРАММИРОВАНИЕ В ЗАНИМАТЕЛЬНЫХ ЗАДАЧАХ

Редактор *Л. О. Полянская*
Оформление художника *Л. А. Дикарева*
Художественный редактор *Л. А. Дикарев*
Технический редактор *Н. А. Бондарчук*
Корректор *М. Г. Гаркавенко*

Информ. бланк № 1798

Сдано в набор 06.02.80.
Подписано в печать 13.08.80.
БФ 04553. Формат 70×90¹/₁₆.
Бумага люксоарт. Гарн. лит.
Печ. офс. Усл. печ. л. 5,85.
Уч.-изд. л. 6,24. Тираж 26 000.
Зак. 59. Цена 1 р. 10 к.
Издательство «Техника»,
252601, Киев, 1, ГСП, Крещатик, 5.
Киевская книжная фабрика «Жовтень»
республиканского производственного
объединения «Полиграфкнига»
Госкомиздата УССР, Киев, Артема, 25.