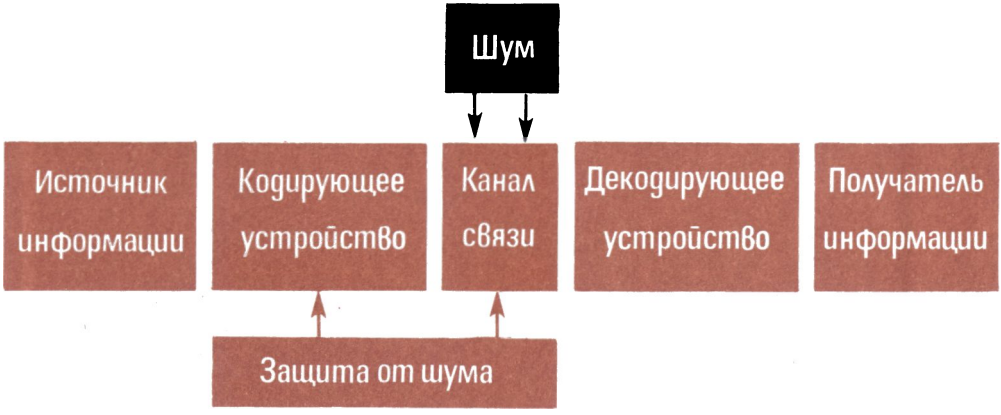
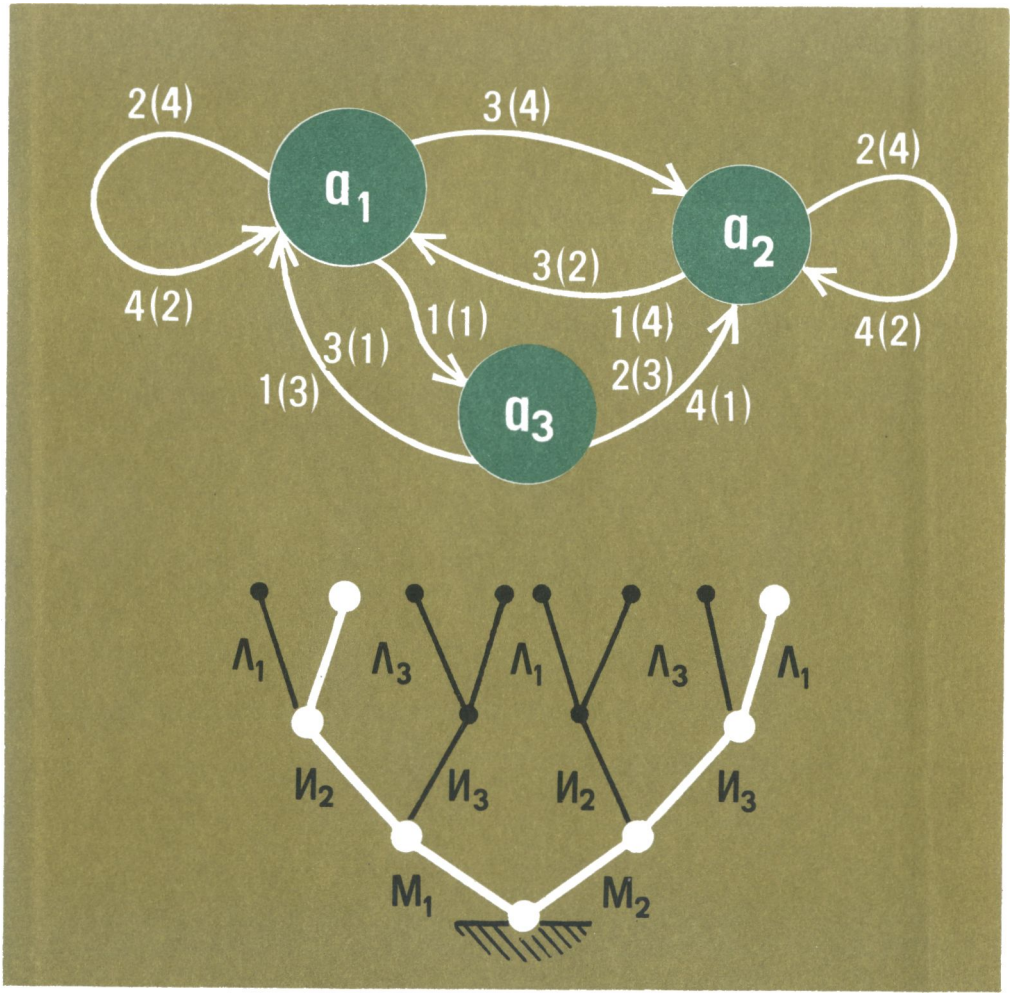
В. Н. КАСАТКИН

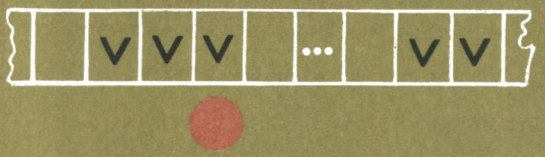
ИНФОРМАЦИЯ АЛГОРИТМЫ ЭВМ

«ПРОСВЕЩЕНИЕ»

В. Н. КАСАТКИН ИНФОРМАЦИЯ АЛГОРИТМЫ ЭВМ

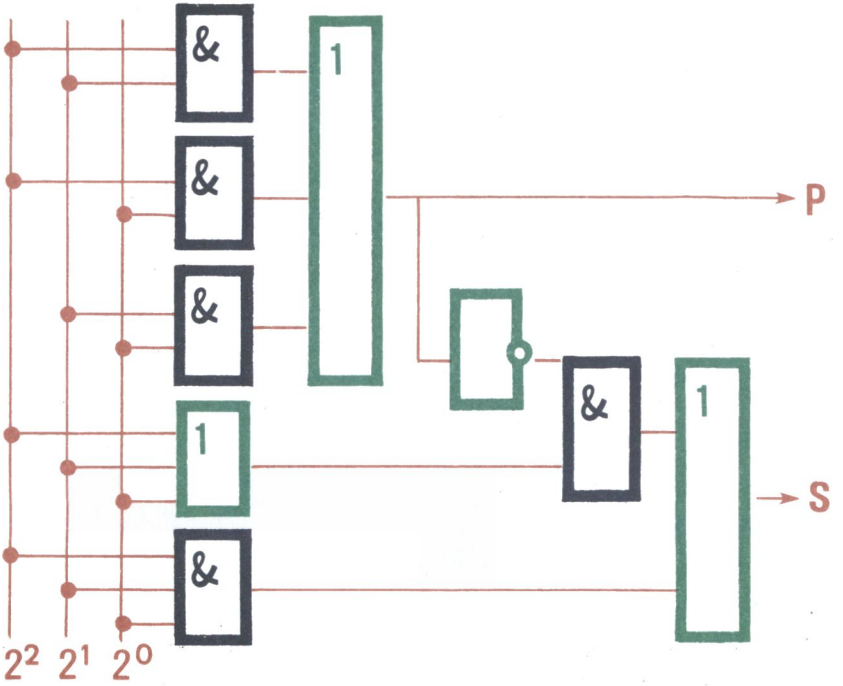

Шум
Источник информации
Кодирующее устройство
Канал связи
Декодирующее устройство
Получатель информации
Защита от шума


2(4)
3(4)
2(4)
a1
a2
3(2)
4(2)
1(1)
1(4)
4(2)
3(1)
2(3)
1(3)
a3
4(1)
Л1
Л3
Л1
Л3
Л1
И2
И3
И2
И3
М1
М2



1. ↕ 2
2. → 3
3. ? < 2, 4
4. ← 5
5. ∨ 6
6. !

	q_0	q_1	q_2	q_3
Δ			$1 \Pi q_3$	Λq_0
0	$1 \Lambda q_0$	Λ	$1 \Pi q_3$	Π
1	$0 \Lambda q_1$	Λ	$2 \Pi q_3$	Π
2			$0 \Lambda q_2$	Π
*	!	Λq_2		Π



В. Н. КАСАТКИН

ИНФОРМАЦИЯ АЛГОРИТМЫ ЭВМ

ПОСОБИЕ ДЛЯ УЧИТЕЛЯ

Рекомендовано
Главным учебно-методическим управлением
общего среднего образования Госкомитета СССР
по народному образованию

МОСКВА «ПРОСВЕЩЕНИЕ» 1991

ББК 74.262
К28

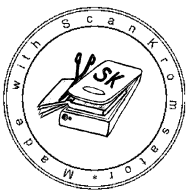
Рецензент:
кандидат физико-математических наук,
старший научный сотрудник МГУ им. М. В. Ломоносова
Г. В. Лебедев

Касаткин В. Н.

К28 Информация, алгоритмы, ЭВМ: Пособие для учителя.— М.: Просвещение, 1991.— 192 с.: ил.— ISBN 5-09-003398-6.

Информация, алгоритмы, ЭВМ — три основные части нового школьного курса «Основы информатики и вычислительной техники». Как лучше ввести учащихся в новый для них мир, показано в этой книге. Рекомендации, примеры и систему задач найдет молодой учитель в предлагаемой книге. В основе книги положен более чем 25-летний опыт работы автора в Крымской Малой академии наук «Искатель».

К $\frac{4306010000-578}{103(03)-91}$ 70—91 ББК 74.262



Учебное издание

Касаткин Валентин Николаевич

ИНФОРМАЦИЯ, АЛГОРИТМЫ, ЭВМ

Зав. редакцией *Т. А. Бурмистрова*
Редактор *А. К. Компанец*
Младший редактор *Л. И. Заседателева*
Художник *Е. М. Молчанов*
Художественный редактор *Ю. В. Пахомов*
Технический редактор *Л. М. Абрамова*
Корректор *И. А. Корогодина*

ИБ № 13381

Подписано к печати с диапозитивов 14. 10. 91. Формат $60 \times 90^1/_{16}$. Бум. офсетная. Гарнит. Литерат. Печать офсетная. Усл. печ. л. 12,0 + 0,31 форз. Усл. кр.-отт. 13,56. Уч.-изд. л. 11,87 + 0,35 форз. Цена 2 р. 10 к.

Ордена Трудового Красного Знамени издательство «Просвещение» Министерства печати и массовой информации РСФСР. 129846, Москва, 3-й проезд Марьиной рощи, 41.

Саратовский ордена Трудового Красного Знамени полиграфический комбинат Министерства печати и массовой информации РСФСР. 410004, Саратов, ул. Чернышевского, 59. Отпечатано при посредстве В/О «Внешторгиздат».
Изготовлено в типографии «Интердрук» ГмбХ, г. Лейпциг, ФРГ

ISBN 5-09-003398-6

ВВЕДЕНИЕ

Основные цели нового школьного курса «Основы информатики и вычислительной техники», его содержание и методы обучения сформулированы в программе. Однако программные установки, как правило, носят принципиальный характер, требуют развернутого методического комментария, потребность в котором сегодня, в период становления курса, особенно велика.

Появлению информатики и вычислительной техники в средней школе в форме самостоятельного учебного предмета предшествовал активный педагогический поиск, осуществлявшийся в течение многих лет в различных районах страны. Отдельные учителя и педагогические коллективы накопили интересный конкретный опыт ознакомления учащихся как с основами вычислительной техники, так и с основами информатики и программирования.

В предлагаемой учителю средней школы книге отражен опыт такой поисковой работы. Книга построена как сборник небольших эссе по основным вопросам программы. Следуя структуре программы, курс рассмаривается тема за темой; в каждой из тем выбраны наиболее интересные проблемы.

Конечно, сегодня такого рода работа не может претендовать на полноту рассмотрения всех проблем методики нового школьного курса. Предлагаемую работу следует рассматривать как обобщающую опыт, как некоторый вспомогательный методический материал. Книга не содержит методических указаний, а включает лишь методические соображения по отдельным вопросам преподавания курса «Основы информатики и вычислительной техники».

Существенной частью книги является подбор примеров и задач по всем основным разделам курса. Все предложенные задания проработаны на занятиях со школьниками.

Читатель книги — учитель приглашается к небольшой методической дискуссии. Ему решать, какие из методических позиций автора он приемлет, какие отвергает.

Автор надеется, что книга окажется полезной и послужит делу становления важной учебной дисциплины.

Глава I. ИНФОРМАЦИЯ

§ 1. ИНФОРМАЦИОННЫЙ АНАЛИЗ. ОБЩИЕ ПОДХОДЫ

В прошлом веке введение понятия «энергия» породило плодотворный методический подход к изучению явлений широкого класса, в том числе и таких, какие до того времени стояли в сознании людей весьма далеко друг от друга.

С единой энергетической точки зрения стало возможно обсуждать и сопоставлять явления электрические и механические, биологические и тепловые, явления, ранее совместно не изучавшиеся.

В настоящее время на такую же роль нового, революционизирующего методику научного познания претендует понятие «информация». Подчеркнем, что информационный анализ применим к еще большему классу явлений: удается, например, изучать в сопоставлении с информационной точки зрения явления социальные и экономические.

Изучение естественных биологических систем включает изучение их поведения под влиянием сигналов из внешней среды. Не понимая, как животное пользуется получаемой им информацией, нельзя глубоко понять основы его поведения.

Невозможно разобраться в работе управляющей системы любой природы, если не знать всех информационных процессов, в ней протекающих. Источником, основой для принятия решений является информация. Реализация управления осуществляется через команды, через подаваемые сигналы, т. е. через информацию.

Уточняя и раскрывая смысл этого понятия для школьников, учитель должен подчеркнуть, что такое качество, как информированность, не должно сегодня рассматриваться как исключительно человеческое качество. Информация сегодня воспринимается не только людьми или животными, но и самыми различными устройствами.

Школьники должны уметь приводить примеры, когда источником информации является прибор, а получателем ее — человек. Должны приводиться примеры передачи информации от человека к устройству и примеры обмена информацией, сигналами между машинами.

§ 2. ОБЪЕКТИВНОСТЬ ИНФОРМАЦИИ

Сопоставляя понятия «информация» и «энергия», подчеркнем, что мы при оценке подводимой энергии (например, тепловой) используем два подхода: оцениваем состояние среды лично, давая ей субъективную оценку (один человек может сказать, что ему уже жарко; другой — еще не жарко) или же объективно, применяя измерение, указывая температуру среды.

Так и при оценке поступающей информации мы можем оценивать ее субъективно. В этом случае на первое место выдвигается смысл — содержание полученного сообщения. Можно прибегнуть и к ее измерению — в этом случае, конечно, следует отказаться от содержания, от человеческой «важности» для кого-то.

Информация, воспринимаемая каким-нибудь устройством из внешней среды, может иметь для получателя конкретный смысл, может вызывать ответную реакцию. Говорить о «важности» информации для машины, получающей информацию, бессмысленно. В этом проявляется объективный характер понятия «информация».

Объективность оценки информационного состояния среды достигается самым активным использованием искусственных органов чувств-датчиков. Датчики не только расширяют диапазон слышимого и видимого, но и могут получать сигналы, не воспринимаемые органами чувств человека. Речь идет, например, о сигналах, информирующих об уровне радиации и т. д.

Учитель должен подвести учащихся к вопросу о соизмеримости информации, получаемой через разнообразные по своей природе датчики. Подобно тому как существует такое качество энергии, как ее измеримость, следует поставить вопрос об измеримости информации.

Для оценки тепловой, механической или электрической энергии мы используем сравнимые между собой единицы.

Учитель ставит вопрос: можно ли сравнивать величину информации, получаемой зрительно, с информацией, получаемой на слух или как-то иначе.

§ 3. ИЗМЕРИМОСТЬ ИНФОРМАЦИИ. ОСНОВНЫЕ ЕДИНИЦЫ

Ниже предлагается методическая схема введения единицы информации. Данная схема должна рассматриваться лишь как возможный вариант изложения этого вопроса школьникам. Учитель может вводить понятие о единице информации с учетом своего стиля и подготовленности учащихся.

Обращаем внимание на то, что понятие о количестве информации, введенное в науке, неадекватно интуитивному представлению о количестве информации. Это обнаруживается тогда, когда осуществляется увязывание количества получаемой информации с

поведением ее получателя. В этом случае на первое место выступает «полезность» или «важность» полученной информации для конкретного получателя. Ясно, что при таком подходе будет иметь место различная оценка одного и того же сообщения. Такой подход наукой отвергается.

Введение понятия об измеримости информации можно осуществить по схеме:

— Учитель разъясняет, что получение информации, ее увеличение одновременно означает уменьшение незнания или информационной неопределенности.

Например, известно, что Иванов живет на улице Весенней. Сообщение о том, что номер его дома есть число четное, уменьшило неопределенность. Получив такую информацию, мы стали знать больше, но информационная неопределенность осталась, хотя и уменьшилась.

— Учитель вводит понятие об информационной неопределенности на примерах.

Пример 1. Шарик находится в одном из восьми ящичков — информационная неопределенность равна восьми.

Ясно, что учителю нужно ограничиться примерами только таких неопределенностей, в которых она создается равновероятными возможностями.

— Вводится основное определение: сообщение, уменьшающее неопределенность ровно вдвое, содержит единицу информации — бит.

Пример 2. Книга лежит на одной из двух полок — верхней или нижней. Сообщение о том, что книга лежит на верхней полке, несет один бит информации.

Пример 3. Шарик находится в одной из трех урн: A, B или C. Сообщение о том, что шарик находится в урне A, несет в себе информации больше, чем бит.

Число примеров, закрепляющих понятие о единице информации, должно быть значительным — желательно, чтобы примеры придумывали и школьники.

— Далее учитель вводит понятие о составном сообщении. Это можно сделать на примере.

Пример 4. Пусть имеется колода из 32 игральных карт (в колоде отсутствуют все четыре шестерки). Задумывается одна из карт. Необходимо, задавая вопросы, на которые будут даны ответы «Да» или «Нет», угадать задуманную карту.

Первый вопрос: «Задумана карта черной масти?»

На этот вопрос получен ответ: «Нет»,— что уменьшило неопределенность ровно вдвое и принесло отгадывающему один бит информации.

Второй вопрос: «Задумана карта бубновой масти?»

Ответ: «Да». (Это еще один бит информации, исходная неопределенность уменьшилась уже в четыре раза.)

Третий вопрос: «Задумана карта-картинка?»

Ответ: «Нет». (Еще один бит информации.)

Четвертый вопрос: «Задумана семерка или девятка бубновые?»

Ответ: «Да». (Еще один бит информации.)

Пятый вопрос: «Задумана семерка бубновая?»

Ответ: «Да». (Еще один, пятый, бит информации.)

Чтобы выяснить, какая карта была задумана, пришлось задать пять вопросов, каждый ответ на умело поставленный вопрос давал один бит информации. Следовательно, в сообщении о любой из задуманных карт будет содержаться пять бит.

Пример 5. Шарик находится в каком-то одном из 64 ящичков. Сколько единиц информации будет содержать сообщение о том, где находится шарик?

Пример 6. Сколько следует задать вопросов и как их следует формулировать, чтобы оценить сообщение о том, что вагон стоит на одном из 16 путей?

Научный подход к оценке сообщений был предложен еще в 1928 г. американским инженером Р. Хартли.

Расчетная формула имеет вид

$$И = \log_2 N, \qquad (1)$$

где N — количество равновероятных событий; И — количество бит в сообщении, такое, что любое из N событий произошло.

Пример 7. Сообщение о том, что шарик находится в одной из трех урн, содержит $И = \log_2 3 = 1{,}585$ бита информации. Иногда формула Хартли (1) записывается иначе. Так как каждое из N возможных событий имеет одинаковую вероятность: $p = 1/N$, то $N = 1/p$ и формула имеет вид

$$И = \log_2 N = \log_2 (1/p) = -\log_2 p.$$

Познакомимся с более общим случаем вычисления количества информации в сообщении об одном из N, но уже не равновероятных событий.

Рассмотрим некоторый умозрительный эксперимент. Пусть имеется генератор, который на своем экране может демонстрировать любую из букв некоего алфавита, состоящего из k букв. Генерирование осуществляется в соответствии с заданным законом распределения.

A_i	A_1	A_2	A_3	...	A_{k-1}	A_k
p_i	p_1	p_2	p_3	...	p_{k-1}	p_k

Каждая из букв появляется на экране в соответствии с вероятностью ее появления.

За экраном ведется наблюдение: пусть на экране уже появлялось N букв (N — достаточно большое число). Если мы заинтересуемся буквой A_i, то она на экране появится приблизительно

$(N \cdot p_i)$ раз. Каждое появление буквы A_i дает $(-\log_2 p_i)$ бит информации. Всего за все ее появления будет получено $(-Np_i \log_2 p_i)$ бит информации.

Общее количество информации, которое следует суммировать после демонстрации всех N букв, равно:

$$И = -N \cdot \sum_{i=1}^{k} p_i \log_2 (p_i).$$

На одну букву в среднем приходится

$$И_{ср} = -\sum_{i=1}^{k} p_i \log_2 (p_i) \text{ бит информации.}$$

Эту формулу впервые вывел американский инженер и математик К. Шеннон в 1948 г.

Рассказывая о единицах информации: байт, килобайт, мегабайт и гигабайт,— уместно привести примеры достижений в создании запоминающих устройств.

Так, на конец 1987 г. имеются результаты:

плотность монтажа электронных элементов на одну микросхему доведена до 20 млн. единиц;

один кристаллик со стороной менее 38 мм содержит 275 000 транзисторов и может хранить 4 млрд. байт памяти. Напомним, что для хранения одного символа, одной буквы или знака действия достаточно одного байта;

емкость ЗУ на магнитных дисках с 1967 г. к 1987 г. выросла с 30 000 до 3 000 000 бит на 1 см. В 1 $см^3$ можно разместить такое количество элементов памяти, которых хватит для записи 15 млн. байт информации (почти 300 романов среднего объема);

пропускная способность оптико-волоконного канала доведена почти до 10 млрд. бит/с;

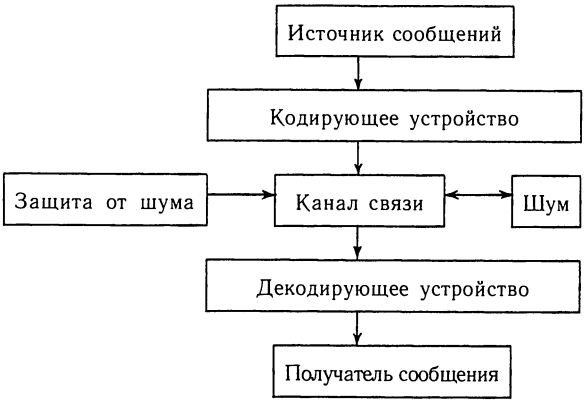
скорость формирования изображений на экране доведена до 4 млн. пикселей (экранных точек) за секунду, при этом возможно окрашивание каждой точки от 16 до 256 различных цветов;

видеодиски с лазерным способом записи и чтения информации могут содержать до 50 тыс. изображений на каждой стороне диска.

Эти краткие сведения служат лишь ориентиром. Учитель, конечно, будет накапливать такого рода материалы, включать в свой автоматизированный педагогический архив.

§ 4. ПЕРЕДАЧА ИНФОРМАЦИИ

Широкое использование информации требует умелой ее передачи, особенно на большие расстояния. При передаче информации возникает немало проблем. При ознакомлении школьников с основными вопросами передачи информации учитель может воспользоваться схемой, предложенной К. Шенноном.



Рассказывая об общей задаче передачи информации, учитель должен ввести понятие о кодировании символов, разъяснить широкий смысл понятия «канал связи» (привести примеры современных каналов связи), объяснить, что такое шум и как от него защищаются. Желательно упомянуть о понятии «пропускная способность канала связи». Все эти вопросы следует иллюстрировать примерами. Приведем несколько примеров конца 1987 г.:

использование световодов (сверхтонкие силиконовые волокна); повышение эффективности лазеров, посылающих импульсы по таким каналам связи, позволяет передавать отдельные сигналы на расстояние до 135 км без усиления;

пропускная способность световодов доведена до 420 Мбит/с, т. е. около 50 Мбайт, или 50 млн. букв. Это значит за секунду 125 мегабайтных листов или почти 10 школьных учебников;

для оптических систем связи (в основе — световоды) характерны как высокая скорость передачи информации, так и большие расстояния между усиливающими подстанциями. В 1987 г. уже действовал трансатлантический световодный кабель;

завершается проектирование канала связи с пропускной способностью в 1,1 Гбит/с.

Учитель должен сообщить школьникам, что в разработке основ теории связи и теории информации большой вклад принадлежит советским математикам А. Н. Колмогорову (1903—1987), А. Я. Хинчину (1894—1959) и советским специалистам по радиотехнике В. А. Котельникову (1908), А. А. Харкевичу (1904—1965) и др.

Полезно школьникам упомянуть о том, что возможности человека по восприятию информации заметно уступают возможностям технических устройств. Так, известно, что количество информации, которое нервная система человека способна передать в мозг при чтении текстов, составляет примерно 16 бит/с. Эта порция удерживается в сознании примерно 10 с, так что одновременно в сознании удерживается 160 бит.

§ 5. ПРЕОБРАЗОВАНИЕ ИНФОРМАЦИИ

Одной из центральных учебных целей в курсе информатики является ознакомление учащихся с задачей содержательного преобразования информации. Учитель разъясняет на ярких примерах суть физического преобразования информации: усиление речи, сжатие снимаемого на пленку изображения во времени и т. д.

Учитель должен подвести школьников к задаче о преобразовании информации «по существу», т. е. о преобразовании, которое затрагивает «смысл» исходной информации и порождает новую информацию: например, нахождение корня из числа, нахождение диаметра окружности по ее длине, расстановка окончаний слов при склонении слов по падежам, подбор слов в рифму и т. д.

Каждый такой пример обсуждается со школьниками, они должны уметь находить примеры сами.

Особенно важно рассмотреть и обсудить с учениками примеры технических устройств, выполняющих такого типа преобразование информации: например, кассовый аппарат в магазине, справочная ЭВМ в авиакассах, ученический микрокалькулятор, шахматный автомат.

При обсуждении каждого такого примера нужно выяснить:

какая информация является исходной;

как осуществляется преобразование исходной информации в результат (этот вопрос требует от учителя находчивости, нужно подчеркивать роль человеческого фактора во всей деятельности ЭВМ и других преобразователей информации);

какая информация получена в результате преобразования.

§ 6. АЛФАВИТНЫЙ СПОСОБ ПРЕДСТАВЛЕНИЯ ИНФОРМАЦИИ

Современный отечественный курс информатики строится с учетом того, что во всех информационных процессах, с которыми школьники знакомятся, информация передается только с помощью дискретных сигналов. Это означает, что в школьной информатике, как правило, изучается алфавитный способ представления информации.

Учитель рассказывает, что алфавитный способ записи информации известен давно. Множество народностей на Земле имеют языки, построенные по алфавитному принципу. Всякая информация в таких языках записывается в виде слов, составленных из букв и символов алфавита, и в виде предложений, составленных из слов.

Алфавит русского языка содержит 33 буквы, немецкого — 26, древнеиндийский язык дева-нагари имеет 50 букв, а классический греческий — 24 буквы. Самый маленький алфавит — алфавит внутреннего языка ЭВМ: в нем всего две буквы — 0 и 1.

В курсе информатики решается важная образовательная задача — расширяется понятие об алфавите. Школьники должны ясно понимать, что понятие «алфавит» принадлежит не только грамматике, но и человеческой культуре в целом.

Можно говорить, например, об алфавите арифметики — тогда в роли «букв» выступят цифры, знаки арифметических действий, скобки и знаки арифметических отношений, а привычные формулы будут словами этого языка. Сегодня школьник и учитель могут встретиться с созданием новых алфавитов и даже новых языков.

Учителю рекомендуется расширить свои личные представления о языках, используемых в общении людей: в частности, целесообразно узнать подробности о создании такого искусственного языка, как эсперанто, и аналогичных ему.

Учителю предстоит вводить в мир своих учеников понятие об алгоритмических языках и о языках программирования. Это языки искусственные. Очень хорошо, если преподаватель сумеет провести сравнение современных искусственных очень сильно целеустремленных языков с языками типа эсперанто. Разве не будет интересно узнать его ученикам, что в эсперанто всего 16 правил и ни одного исключения. Полезно рассказать школьникам и о причинах, вызывавших изобретение искусственных языков человеческого общения, о том, что люди изобрели более 300 таких языков.

Несомненно, интересным в глазах учащихся будет такой факт, что язык эсперанто назван так в честь польского врача Л. Заменхофа (1859—1917), который опубликовал свою лингвистическую систему под псевдонимом доктор Эсперанто, а язык «Ада» — современный язык программирования — назван в честь Ады Лавлейс — дочери поэта Байрона, которая одна из первых научилась писать программы.

Более широкое представление о языках в связи с тем, что общение человека с человеком в наше время дополняется общением человека с машиной, в частности с ЭВМ, становится необходимым элементом культуры, необходимым условием компьютерной грамотности.

§ 7. КОДИРОВАНИЕ ИНФОРМАЦИИ. ОБЩИЕ ПРОБЛЕМЫ

В различных вариантах школьных курсов информатики вопросам кодирования информации может уделяться неодинаковое внимание. Для учителя, ведущего курс, знание проблем кодирования информации существенно важно.

Изучая цели и методы кодирования информации, учитель коснется самых общих проблем преобразования информации, узнает о глубинных проблемах преобразования информации из одного вида в другой. Примером могут быть вопросы преобразования

звуковой информации в информацию, представленную в алфавитной форме. Для учителя важно знать, как непрерывный сигнал квантуют, т. е. представляют в дискретной форме, в виде последовательности идущих один за другим сигналов.

Учителю желательно иметь представление о том, что перед передачей информации ее можно закодировать так, что получатель узнает, произошел ли «сбой» при ее передаче. Необходимо свой багаж знаний, связанный с информатикой, пополнять сведениями о проблемах и достижениях теории кодирования.

Теория кодирования и древнейшее искусство тайнописи — искусство криптографии — близки друг другу. Над разработкой различных шифров трудились многие известные ученые: философ Ф. Бэкон, математики Д. Кардано, Д. Валлис. Естественно, что одновременно с развитием методов шифровки развивались приемы расшифровки, или криптоанализа. Например, французский математик Ф. Виет (1540—1603) нашел ключ к шифру, которым пользовались испанцы во время войны с французами, и даже сумел проследить за всеми его изменениями.

В середине XIX в. ситуация изменилась. Изобретение телефона и искрового телеграфа поставило перед учеными и инженерами проблему создания теории связи, и, главное, в первую очередь новой теории кодирования. В теории кодирования связь между людьми была потеснена интересом к проблеме связи между устройствами.

Упомянем, что первой ориентированной на технику системой кодирования оказалась азбука Морзе. Это подчеркивается потому, что в азбуке Морзе принято двоичное кодирование, которое сегодня играет огромную роль в мире вычислительных машин.

§ 8. ДВОИЧНОЕ КОДИРОВАНИЕ, ЕГО УНИВЕРСАЛЬНОСТЬ

Если учитель знакомит учащихся с двоичным кодированием, то план занятия может быть таким:

1. Определение двоичного алфавита: $A=\{0, 1\}$.

2. Правила двоичного кодирования:

а) двоичные коды букв данного алфавита B должны иметь одинаковое число двоичных букв;

б) коды различных букв алфавита B должны быть различными;

в) кодируются только буквы алфавита B, каждой ставится в соответствие отдельный код.

При соблюдении этих правил по имеющемуся двоичному коду можно всегда восстановить исходный символ. Для этого достаточно разбить двоичное слово на части, на подслова, каждое длиной n двоичных букв, затем каждое полученное двоичное — буквенное — подслово заменить соответствующим символом исходного алфавита B.

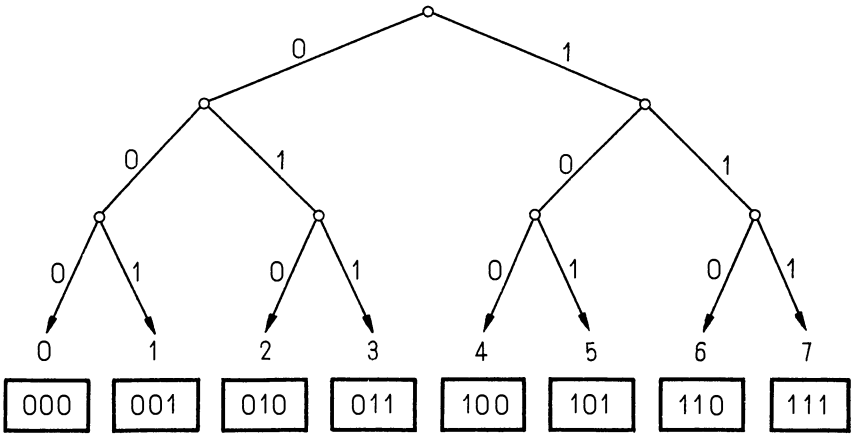


Рис. 1

Пример. Пусть дан алфавит $B=\{0, 1, 2, 3, 4, 5, 6, 7\}$. Необходимо закодировать в двоичном алфавите $A=\{0, 1\}$ каждый символ алфавита B.

Учитель может ознакомить своих учеников с такой, например, системой выбора двоичных кодов. Он рисует на доске двоичное дерево (рис. 1). Двигаясь по ветви, ведущей к требуемому символу, и выписывая подряд двоичные буквы 0 и 1, получают искомый код любого из символов. Эти коды на рисунке выписаны внизу в рамках. В данном примере велось кодирование цифр восьмеричной системы счисления числами двоичной системы счисления.

В качестве упражнения можно дать задание школьникам закодировать цифры системы с основанием $q=16$, т. е. написать коды для каждого символа алфавита:

$$B=\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F\}.$$

Учитель может нарисовать на доске нижнюю часть двоичного дерева, например так, как это показано на рисунке 2. Предложенная схема формирования кодов — одна из возможных.

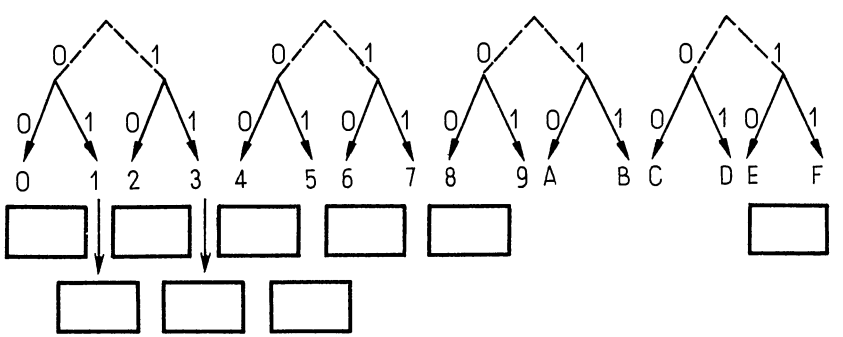


Рис. 2

Двоичное кодирование — один из распространенных способов представления информации. Важно подчеркнуть, что в вычислительных машинах, в роботах и станках с числовым программным управлением, как правило, вся информация, с которой имеет дело устройство, кодируется в виде слов двоичного алфавита.

Главное достоинство двоичного кодирования, которое привлекает конструкторов устройств, преобразующих информацию, в том, что физически очень просто можно «изобразить» и букву 1, и букву 0. Буква 1 может изображаться зажженной лампочкой, а буква 0 — потушенной. Буква 1 может изображаться намагниченной точкой, а буква 0 — размагниченной. Можно рассказать и о других возможностях физического моделирования букв 0 и 1.

К недостаткам двоичного кодирования относят то обстоятельство, что двоичные коды очень длинные. С такими кодами трудно работать.

Обратим внимание учителя на один из интересных вопросов, возникающих при изучении двоичного кодирования. Речь идет о том, что об информационной ценности двоичного кода судить очень просто: каждый двоичный код несет столько бит информации, сколько цифр в нем. Если в коде 101 три буквы, то этот код или слово несет три бита информации. Такой подход хорошо согласуется с ранее данным определением понятия «единица информации» — *бит*.

Разъяснить этот подход можно такими соображениями: пусть некто задумал одну из восьми букв алфавита $B=\{0, 1, 2, 3, 4, 5, 6, 7\}$.

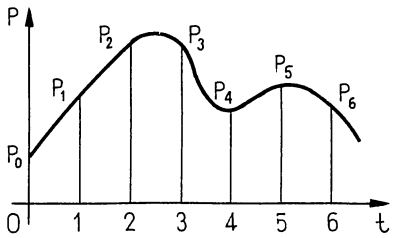
Если вернуться к рисунку 1, то, с одной стороны можно увидеть, что любой символ b_i кодируется трехбуквенным двоичным кодом. С другой стороны, по теореме Хартли имеем $\log_2 8=3$.

Для кодирования символов в алфавите, содержащем 16 букв, потребуется привлечь четырехбуквенные двоичные коды. Если количество кодируемых символов $n=2^k$, то для кодирования следует избрать k-разрядные двоичные коды.

Весьма важным вопросом, над изложением которого должен подумать учитель, является проблема разъяснения понятия *универсальность двоичного кодирования*.

Из рассказанного выше понятно, как следует кодировать буквы самых разнообразных алфавитов. Для теории информации принципиально важным является тезис о том, что двоичное кодирование можно успешно применять при записи не только дискретной, но и непрерывной информации. Этот принцип можно разъяснить следующим образом.

На рисунке 3 изображена барограмма, вычерченная прибором в течение шести часов проведения опыта. Рассматривая барограмму, можно сделать ряд выводов о том, как менялось давление в ходе опыта. Пусть требуется эту барограмму запомнить, занести сведения о процессе в память. Как это сделать? Один из путей таков: необходимо вычерченную кривую (барограмму) заменить таб-



P_i	P_0	P_1	P_2	P_3	P_4	P_5	P_6
t	0	1	2	3	4	5	6

Рис. 3

лицей результатов. В таблицу занесены показания барометра на конец каждого часа с начала опыта.

Легко заметить, что таблица дает искаженную картину процесса. Очень важные события, наблюдавшиеся в опыте, в таблице не отражены: например, не указано самое большое значение давления, достигнутое между вторым и третьим часом опыта.

Ясно, что таблицу можно улучшить, если занести в нее значения давления, наблюдаемые каждые полчаса. Очевидно, что новая таблица более подробно задаст процесс опыта. Числовые значения давления P_0, P_1, P_2, ... можно закодировать двоичными словами.

Этот пример убедит учащихся в том, что информацию, заданную в непрерывной форме, можно представить алфавитным способом с достаточной для практических целей точностью.

§ 9. ИНФОРМАТИКА — ШКОЛЬНЫЙ АСПЕКТ

Весь цикл занятий, посвящен разъяснению понятия «информация» и сопутствующих вопросов и должен быть связан с раскрытием содержания школьного курса «основы информатики и вычислительной техники».

Термин «информатика» ранее обозначал науку об общих свойствах представления научной информации, об общих закономерностях и особенностях научной коммуникации, о способах организации информационно-поисковых систем.

Научные исследования в давно сложившейся информатике ведутся в таких направлениях:

изучение истории и организации научно-информационной деятельности в различных отраслях знания;

разработка форм научной документации;

изучение свойств документационных потоков;

решение задач формализации извлечения основного смысла из научных документов;

создание систем информационного обуслуживания;

применение техники в библиотечном и архивном деле.

В СССР информатика активно начала развиваться с 50-х гг.,

после создания Всесоюзного института научной и технической информации (1952).

Информатика в школе — это новый *учебный предмет.* В рамках именно этого предмета начинается решение одной из основных задач реформы школы: «...вооружать учащихся знаниями и навыками использования современной вычислительной техники, обеспечить широкое применение компьютеров в учебном процессе».

В методической литературе можно встретить много определений информатики. Мы присоединяемся к определению, данному А. П. Ершовым и соавторами в их методическом пособии «Изучение основ информатики и вычислительной техники»: «Информатика — это отрасль науки, изучающая вопросы, связанные с поиском, сбором, хранением, преобразованием и использованием информации в самых различных сферах человеческой деятельности» (ч. 2, с. 10).

Напомним учителю, что в новом предмете выделяется три важных раздела:

Объект изучения — информация. Информация при этом разъясняется учащимся как фундаментальная компонента окружающего мира. Понятия «вещество», «энергия» и «информация» одинаково необходимы при анализе явлений окружающей действительности.

Алгоритмический тип деятельности. Особенности и значение этого типа деятельности для человека. Возможности и средства реализации такого типа деятельности в системах и машинах. Центральным рабочим понятием в данном разделе является понятие об алгоритме.

Новые орудия труда. ЭВМ как средство для выполнения всей совокупности задач, связанных с поиском, сбором, хранением, преобразованием и использованием информации. Центральной практической задачей в этой части курса является обучение подготовке задач к их решению на ЭВМ, обучение программированию и работе на ЭВМ.

В рамках новой учебной дисциплины формируют два новых педагогических феномена: алгоритмическая культура и компьютерная грамотность. Наряду с такими педагогическими задачами, как формирование культуры речи, культуры поведения в обществе, математической культуры, встает проблема формирования алгоритмической культуры как компоненты современного мировоззрения человека.

Овладеть алгоритмической культурой для школьника — это значит:

хорошо понимать, что значит алгоритмический тип деятельности, уметь объяснять его достоинства и недостатки;

научиться (по возможности) нечто сложное представлять в виде совокупности простого, создавать простейшие алгоритмы в различных предметных областях;

знать основные типы алгоритмов;

овладеть одним (или несколькими) алгоритмическим языком.

Учащиеся должны понимать: как через алгоритмизацию пролегает путь к механизации, а далее — и к полной автоматизации выполнения конкретной работы.

Овладеть компьютерной грамотностью для учащегося — это значит:

понять, как готовятся задачи к их решению на ЭВМ;

разобраться в основных идеях моделирования, в том числе и доступного для него математического моделирования;

овладеть одним (или несколькими) алгоритмическим языком;

научиться программировать;

научиться работать на персональной ЭВМ или на ЭВМ другого класса.

В заключение обратим внимание учителя, что школьный предмет называется «Основы информатики и вычислительной техники». Мы подчеркиваем вторую часть названия предмета для того, чтобы учитель не сводил все обучение к пользовательскому аспекту человека с ЭВМ. Разделы курса, посвященные структуре ЭВМ, функциям ее отдельных узлов, принципу самоуправления ЭВМ, не должны в сознании учащихся быть второстепенными. Нужно помнить о том, что найдется немало школьников, для которых наиболее интересным и увлекающим будет не ответ на вопрос: как заставить ЭВМ решить задачу? а ответ на вопрос: как работает ЭВМ? Как устроена ЭВМ?

Глава II. АЛГОРИТМ

§ 10. АЛГОРИТМ В КУРСЕ ШКОЛЬНОЙ ИНФОРМАТИКИ

«Термин «алгоритм» обязан своим происхождением великому ученому средневекового Востока, чье имя — Мухаммед ибн Муса ал Хорезми. Он жил приблизительно с 783 по 850 гг.»,— из книги, изданной в 1987 г. (Успенский В. А., Семенов А. Л. Теория алгоритмов: основные открытия и приложения).

Теория алгоритмов имеет большое практическое значение. Алгоритмический тип деятельности важен не только как мощный тип деятельности человека, как одна из эффективных форм его труда. Через алгоритмизацию, через расчленение сложных действий на все более простые, на действия, выполнение которых доступно машинам, пролегает путь к автоматизации.

С алгоритмическим способом деятельности человек встречается на каждом шагу, имеют с этим способом дело и учащиеся.

Пример 1. Дан угол. Необходимо провести биссектрису. (Есть способ, в котором указано, как, пользуясь линейкой и циркулем, можно решить эту задачу.)

Пример 2. Даны два целых числа. Необходимо найти их разность. (Имеется правило, в котором ясно изложен весь порядок действий с цифрами данных чисел.)

Пример 3. Рассматривается окончание шахматной партии: на доске остались у белых король и ладья, у черных — только король. Черным следует сдаться, ибо есть рекомендации, пользуясь которыми белые обязательно поставят мат черным. По этим рекомендациям у черных нет никаких шансов на выигрыш.

В трех приведенных примерах речь идет о том, как сложную работу представить в виде последовательности простых действий. Вычитание многоразрядных чисел сводится к действиям с цифрами. При делении угла пополам выполняются совсем несложные построения линейкой и циркулем.

Высказанные соображения следует дополнить: правила вычитания — это правила вычитания любых многоразрядных чисел, а не каких-то конкретных двух. Инструкция проведения биссектрисы такова, что, пользуясь ею, можно разделить пополам любой угол. В шахматном окончании — король и ладья против короля — рекомендации применимы к любым допустимым шахматными правилами исходным позициям.

Опираясь на сказанное, учитель может дать полное разъяснение понятия «алгоритм». Далее под алгоритмом будет пониматься конечная система указаний, адресованных исполнителю, четко и однозначно задающих процесс решения задач какого-либо типа во всех деталях.

Алгоритмический способ деятельности состоит в том, что исполнитель (сначала учитель ведет речь только об исполнителе-человеке) либо сам разрабатывает алгоритм, либо получает его в готовом виде и затем исполняет, строго следуя всем указаниям, образующим алгоритм.

Для учащихся должно быть понятно, что в приведенных ранее трех примерах слова «способ», «правило», «рекомендации» можно заменить одним словом — «алгоритм». Тогда можно говорить об алгоритме деления любого угла пополам, алгоритме вычитания многоразрядных чисел и алгоритме шахматного эндшпиля.

Конечно, в курсе информатики в центре внимания будут алгоритмы преобразования информации, в частности алгоритмы преобразования числовой и символьной информации.

Создание алгоритма для решения задач какого-либо типа, его представление исполнителю в удобной для него форме — это творческий акт. Образно говоря, историю математики можно было бы назвать историей открытия алгоритмов и их внедрения в человеческую практику. Сегодня мы наблюдаем, как растет стремление продвигать алгоритмический способ в различных областях трудовой деятельности людей; мы видим, как ширится класс задач, которые удается алгоритмизировать.

§ 11. АЛГОРИТМЫ — РАЗНООБРАЗИЕ И ЭФФЕКТИВНОСТЬ

Начиная изучение понятия «алгоритм», стоит обратить внимание учащихся на несколько простых, но неожиданных и впечатляющих алгоритмах, на примерах показать, сколь увлекательно следить за мыслью тех, кому оказалось по силам создание этих алгоритмов.

Ниже приводится две контрастирующие друг с другом задачи. В первой, на наш взгляд, алгоритма в принципе быть не может. Дело в том, что в формировании ситуации участвует случай, и кажется, что его влияние непреодолимо. Однако человеческая изобретательность оказалась сильнее. Случай уступил алгоритму.

Во второй задаче требуется найти алгоритм беспроигрышного участия в очень простой игре. Игра настолько проста, что каждому, с ней знакомящемуся, кажется, что алгоритм найти несложно. Но из простоты постановки задачи не следует в данном случае простота разработки алгоритма. До сих пор такой алгоритм не найден.

З а д а ч а 1. Дан бочонок, который может вращаться вокруг вертикальной оси. В днище бочонка имеются четыре совершенно

одинаковых и симметричных относительно центра отверстия, в любое из которых можно опустить руку. В бочонке под каждым отверстием помещен сосуд (находится он либо кверху горлышком, либо — днищем). Что-либо увидеть через отверстие невозможно.

Правила обращения с бочонком следующие. В бочонок можно опускать руки в любые отверстия (в одно отверстие — одну руку). Опущенными руками в бочонке можно при желании перевернуть один (любой) или оба сосуда. Можно и не переворачивать.

После действий с сосудами руки из бочонка вынимаются, и он сам начинает вращаться. Затем бочонок останавливается. Узнать те отверстия, в которые перед вращением опускались руки, невозможно: бочонок после вращения занимает случайное положение, а отверстия по внешнему виду неразличимы.

Затем вновь можно в любые два отверстия опустить руки и манипулировать с сосудами. Делается это для достижения цели: расположить все четыре сосуда либо горлышком кверху, либо книзу.

Алгоритм этой задачи изображен на рисунке 4.

Учителю рекомендуется проанализировать его, показав тем самым, как интересно может устремляться мысль человека при поиске алгоритма.

З а д а ч а 2. Игровое поле — квадратная доска, разделенная на одинаковые клетки. Число клеток может быть четным и нечетным. На каждой клетке установлена шашка.

Играют двое, ходят поочередно. За каждый ход любой из играющих может снять с поля любое количество шашек либо из одного вертикального, либо из одного горизонтального ряда. Брать шашки можно только подряд, не перепрыгивая через пустые клетки. Если, например, первый игрок берет две средние шашки из верхнего ряда, то противник не имеет права брать две оставшиеся шашки за один ход. Проигрывает тот, кто берет последнюю шашку.

Изобретатель этой игры (часто ее называют «такс-тикс») — датский физик П. Хейн. Игра «такс-тикс» возникла сравнительно недавно: ей всего около 30 лет. Как занимательная математическая игра она представляет большой интерес. С одной стороны, игра очень проста, увлекательна, а с другой — отыскание ее беспроигрышной стратегии пока проблема. Вот что пишет по этому поводу М. Гарднер: «Игра «такс-тикс» значительно сложнее, чем это кажется на первый взгляд. До сих пор неизвестно, кто выигрывает даже на доске 4×4, с которой сняты угловые фишки».

Одним из ее вариантов, когда победителем считается тот из игроков, кто сделает последний ход, овладеть нетрудно. Отыскание беспроигрышного алгоритма этого «прямого» варианта игры вполне посильная для школьника задача.

Однако, если играть во второй («обращенный») ее вариант (в этом варианте проигрывает тот, кто сделает последний ход), игра «такс-тикс» неожиданно оказывается сложной.

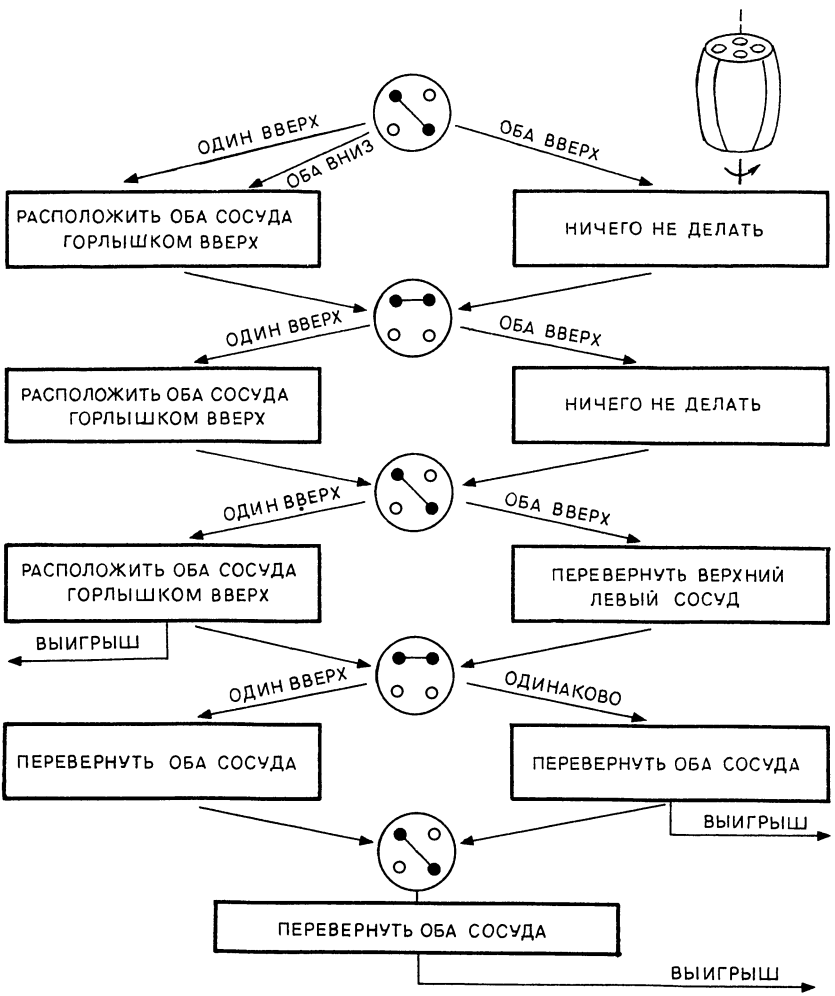


Рис. 4

Приводим комментарии М. Гарднера:

«Обычная («прямая») игра тривиальна из-за слишком простой стратегии, поэтому в «такс-тикс» следует играть «наоборот» и считать проигравшим того, кто возьмет последнюю фишку. Если игра ведется на квадратной доске с нечетным числом клеток, то первый игрок может одержать победу, взяв фишку, стоящую на центральной клетке, и делая затем ходы, симметричные ходам противника. На доске с четным числом клеток второй игрок выигрывает, делая ходы, симметричные ходам своего противника. Для обращенной игры ничего похожего неизвестно...»

Далее он пишет: «Даже элементарный квадрат 4×4 поддается анализу с большим трудом, а при увеличении числа клеток сложность игры быстро возрастает».

Учителю рекомендуется после рассказа об условиях игры и ее истории предложить учащимся попробовать свои силы в отыскании алгоритма. Будущий алгоритм должен содержать ответы на два вопроса: каким по очереди вступать в игру? Как рассчитывать каждый свой ход и каков принцип расчета хода?

Школьники, как правило, принимают вызов и с увлечением работают над поиском алгоритма. Ниже мы приводим текст предполагаемого алгоритма, который разработал девятиклассник из сельской школы.

Пусть число клеток игрового поля равно k^2, и тогда если k — нечетное число, то вступай в игру первым, если же k — число четное, то — вторым.

Делая первый ход, бери все шашки среднего вертикального ряда.

Всякий очередной ход делай симметрично ходу соперника. При этом если k нечетно, то используй симметрию относительно среднего вертикального ряда (на первом ходу из этого ряда были взяты все шашки). Если же k четно, то используй симметрию относительно центра игрового поля.

Перед выполнением очередного (не первого) хода необходимо анализировать размещение оставшихся шашек на игровом поле. Особое внимание следует уделять подсчету одиночных шашек, остающихся еще на поле. Делать очередной ход следует так, чтобы число оставшихся одиночных шашек было четным.

Руководствуясь сформулированными соображениями, вы рано или поздно начнете работать с последним массивом шашек, и тогда будет ясно, что из этого массива следует взять все шашки, кроме одной.

Рекомендуется дать этот предполагаемый алгоритм на проверку учащимся: может быть, кто-то попробует найти путь к доказательству.

Не менее впечатляющим является алгоритм решения задачи о фальшивой монете.

З а д а ч а 3. Пусть имеется 12 неотличимых по внешнему виду монет. Одна из монет фальшивая: она отличается от настоящих по весу, но при этом неизвестно, тяжелее или легче она настоящей монеты.

Можно ли за три взвешивания на весах без гирь обнаружить фальшивую монету и определить, легче она или тяжелее настоящей?

Найти алгоритм непросто. Суть алгоритма в следующем: все монеты перенумеровываются. Взвешивание осуществляется три раза.

При первом взвешивании на левую чашу весов кладутся монеты 1, 2, 3, 4, а на правую — монеты 5, 6, 7, 8.

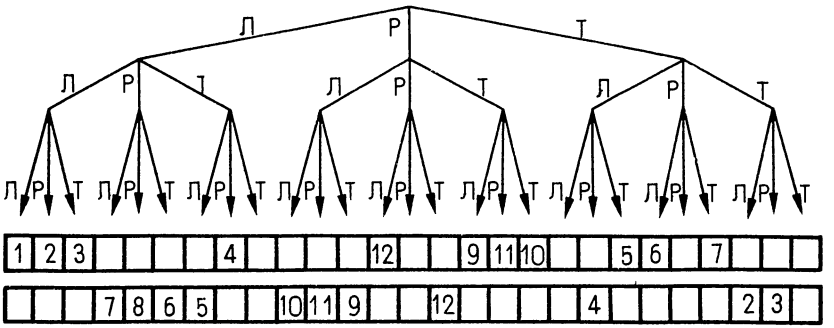


Рис. 5

При втором взвешивании на левую чашу весов кладутся монеты 1, 2, 3, 5, а на правую — 4, 9, 10, 11.

При третьем взвешивании на левую чашу весов кладутся монеты 1, 6, 9, 12, а на правую — 2, 5, 7, 10.

При каждом отдельном взвешивании наблюдение ведется только за левой чашей весов. Может оказаться, что монеты, лежащие на левой чаше, легче, тяжелее или равны по суммарному весу монетам, лежащим на правой чаше.

Каждое взвешивание фиксируется записью: Л (легче), Т (тяжелее) и Р (равно).

Номер взвешивания	Левая чаша	Правая чаша
1-е взвешивание	1, 2, 3, 4	5, 6, 7, 8
2-е взвешивание	1, 2, 3, 5	4, 9, 10, 11
3-е взвешивание	1, 6, 9, 12	2. 5, 7, 10

Все возможные комбинации результатов взвешивания можно изобразить в виде графа-дерева так, как показано на рисунке 5. Это дерево имеет 27 ветвей. Нетрудно заметить, что в 12 случаях фальшивая монета легче, а в 12 случаях — тяжелее. Внимательно проанализировав каждую из 27 ветвей, можно указать номер фальшивой монеты и определить, тяжелее или легче она настоящей.

§ 12. АЛГОРИТМИЧЕСКИЕ ЯЗЫКИ. ЯЗЫК СХЕМ АЛГОРИТМОВ

Выше уже подчеркивалось, что алгоритмы могут быть представлены для исполнения в различных формах.

В вычислительной технике и программировании широкое распространение получила польская запись выражений перед их вы-

числением. Польской записью она называется потому, что впервые была введена польским философом Я. Лукасевичем (1878—1956) в связи с формулами символической логики.

В настоящее время более распространена разновидность польской записи, называемая обратной польской записью.

Пример. Пусть исходная формула имеет вид

$$a \times b + c \times d.$$

Тогда эта же формула в обратной польской записи будет такой:

$$ab \times cd \times +.$$

Обратная польская запись задается схемой:

<операнд> <операнд> <операция>.

Указатель операции помещается после записи операндов. Запись выражения, значение которого вычисляется в форме обратной польской записи, определяет процесс вычисления однозначно: в этой записи отпадает необходимость в скобках и в учете старшинства операций. Эти особенности значительно упрощают обработку данных выражений при их записи на машинном языке.

Примеры:

Обычная запись:	Польская запись:
$(b+c) \times d;$	$bc+d\times;$
$a+(b+c) \times d.$	$abc+d\times+.$

Иногда польскую запись называют бесскобочной записью. Это хороший пример специально созданного формального языка для записи формульных алгоритмов. Это пример алгоритмического языка. Как и всякий язык, он имеет свой адресат: это специалисты, которые работают на высших уровнях программирования.

По нашему мнению, школьникам в качестве первого алгоритмического языка следует рассказать о языке при записи программ для машины Поста. Этот язык крайне прост, рассказ же о его структуре и средствах несет немало важных образовательных идей. Ниже некоторые мысли по этому поводу будут проиллюстрированы примерами.

Без сомнения, следующим языком, объясняемым детально, для учащихся должен стать язык схем алгоритмов. Укажем, что первые понятия о языке схем алгоритмов ввели в 1956 г. советские математики А. А. Ляпунов и Ю. Н. Янов. На Украине этими же вопросами занимался в 1959 г. Л. А. Калужнин.

В каждой схеме алгоритма последовательность указаний, идущих от автора алгоритма к исполнителю, изображается с помощью линий, соединяющих отдельные указания. Существует согласованное число условных обозначений для указаний различных типов. Для придания наглядности и единообразия схем алгоритмов все графические элементы стандартизированы (ГОСТ 19.002—80).

Следует подчеркнуть, что потребность обращения к схеме у

учащихся должна быть воспитана на первых занятиях по изучению понятия «алгоритм». Школьники должны понимать достоинства и недостатки записи алгоритмов в виде схем. Обо всем этом учащиеся должны узнавать из хорошо подобранных примеров.

Таким примером может быть сопоставление двух форм представления одного и того же алгоритма: алгоритма Евклида для отыскания НОД двух чисел, заданного в виде инструкции и в виде схемы.

Инструкция-алгоритм:

«Чтобы найти НОД двух чисел, составьте таблицу из двух столбцов и назовите их *X* и *Y*. Запишите первое из данных чисел в столбец *X*, а второе — в столбец *Y*. Если данные числа не равны, замените большее из них результатом вычитания из большего числа меньшего. Повторяйте такие замены до тех пор, пока числа не окажутся равными, после чего число из столбца *X* считайте искомым результатом». Этот же алгоритм изображен в виде схемы на рисунке 6.

Схема наглядно демонстрирует все связи между элементами. Хорошо различаются элементы, в которых записаны условия ветвления (ромбы), элементы, в которых записаны указания о работе над числами (прямоугольники), а также элементы ввода информации и ее вывода (параллелограммы).

Язык схем настолько четок, что исполнитель, получивший схему алгоритма, ни в каких дополнительных разъяснениях автора алгоритма не нуждается.

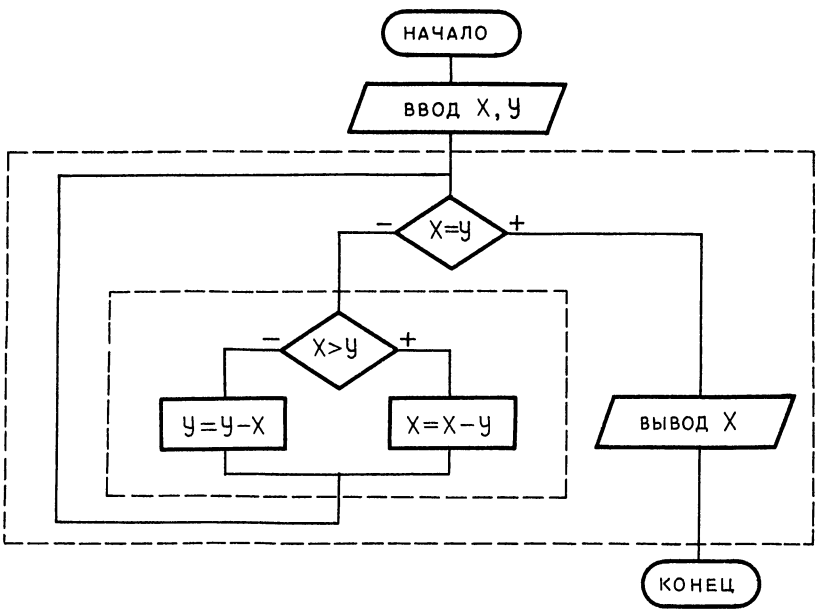


Рис. 6

Язык схем алгоритмов — важный тип языка для представления алгоритмов; схема может быть важным дополнением к алгоритму, записанному в какой-либо форме.

§ 13. СХЕМЫ АЛГОРИТМОВ — МЕТОДИКА ИСПОЛЬЗОВАНИЯ

Очень интересным элементом при ознакомлении школьников с языком схем алгоритмов является демонстрация большой общности схем алгоритмов.

Разъясним эту мысль примером. Учащимся предлагается для анализа схема алгоритма, изображенная на рисунке 7.

Учитель ставит вопросы: какой будет выведен результат, если $a=2$, $b=0$? Каким будет результат при $a=2$, $b=1$? Какой смысл величины k? Какими должны быть значения величин a, b и k, чтобы в процессе вычислений были найдены первые 10 членов последовательности: 5, 9, 13, 17, ...? Какими должны быть значения величин a, b и k, чтобы результатом вычислений оказались первые 7 чисел, которые при делении на 3 дают в остатке 2? Какой будет результат при $a=0$?

Завершая работу с данной схемой, учитель должен подчеркнуть, что алгоритм позволяет вычислять члены любой арифметической прогрессии, где a и b — заданы.

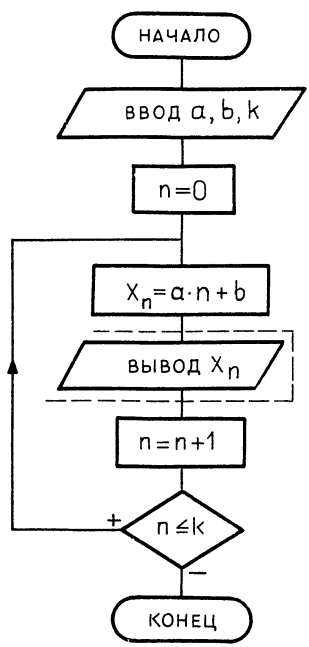


Рис. 7

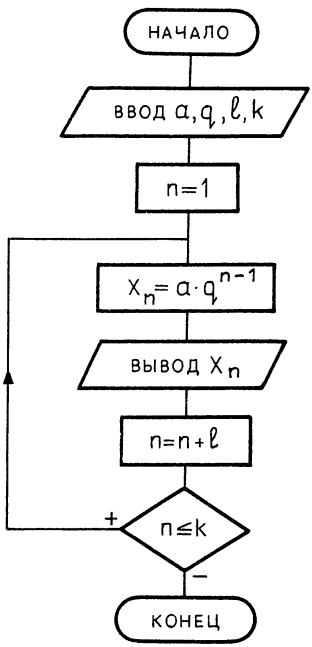


Рис. 8

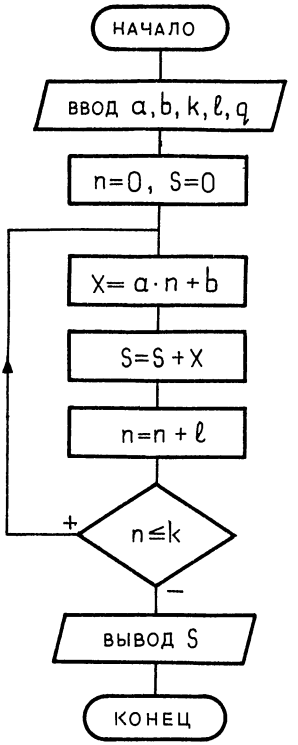


Рис. 9

a — ПЕРВЫЙ ЧЛЕН АРИФМЕТИЧЕСКОЙ ПРОГРЕССИИ;
b — РАЗНОСТЬ ПРОГРЕССИИ;
k — ЧИСЛО ЧЛЕНОВ,
ℓ — ПРИРАЩЕНИЕ НОМЕРА ЧЛЕНА ПРОГРЕССИИ.

$X = a \cdot q^n$

a — ПЕРВЫЙ ЧЛЕН ГЕОМЕТРИЧЕСКОЙ ПРОГРЕССИИ
q — ЗНАМЕНАТЕЛЬ

Следует привлечь внимание учащихся к тому, что совсем небольшое изменение одного указания значительно влияет на весь результат. Рекомендуется рассмотреть алгоритмы, получающиеся, если в них $n=n+1$ заменить на $n=n+2$, $n=n+3$ или $n=c\cdot n+d$.

Интересно проанализировать схему алгоритма, изображенную на рисунке 8.

Учащиеся должны самостоятельно убедиться в том, что схема задает процесс вычисления первых членов геометрической прогрессии (первый член равен a, знаменатель равен q). Учащиеся должны уметь объяснить смысл параметра l.

Еще пример: рассматривается задача суммирования членов различных последовательностей. Задача решается путем дополнения ранее созданной схемы (см. рис. 7).

Рассмотрим фрагмент схемы. Вместо выделенного пунктиром элемента, содержащего указание о выводе x_n, в схему вносится новый элемент с указанием $S=S+x_n$, и непосредственно перед указанием $x_n=a\cdot n+b$ дополнительно вносится блок с указанием $S=0$.

Измененная схема алгоритма суммирования членов последовательности с общим членом ($x_n=a\cdot n+b$) изображена на рисунке 9.

Эту схему рекомендуется использовать для закрепления навыков анализа схем алгоритмов. Рекомендуется поработать с учащимися над вопросами: пусть $a=2$, $b=1$, $k=S$, $l=1$ — каким будет результат? Пусть дана последовательность 3, 7, 11, 15, ...— какие значения следует придать переменным a, b, k и l, чтобы найти сумму $a_1+a_4+a_7+a_{10}$ членов заданной выше последовательности?

Хорошая проработка приведенной схемы должна подготовить

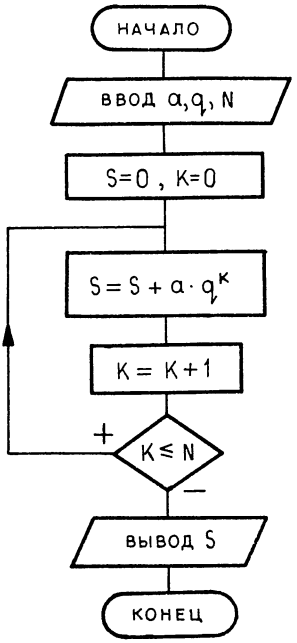


Рис. 10

учащихся к решению задачи суммирования чисел вида $a \cdot q^n$.

Разработку алгоритма суммирования чисел вида $a \cdot q^n$ школьники могут осуществить самостоятельно.

Если задача школьниками решена не будет, то учитель может предложить готовую схему и организовать глубокий ее анализ. Схема может быть такой, какая изображена на рисунке 10.

Подчеркнем, что знакомство со схемами алгоритмов в начале курса — это главным образом анализ алгоритмов. Методы конструирования, синтеза алгоритмов будут рассматриваться позднее. В начале курса информатики главное — полный анализ алгоритма, хорошее понимание всех возможностей конкретной схемы алгоритма, понимание роли каждого из аргументов алгоритма.

Умелое использование аргументов (в приведенных выше алгоритмах в роли аргументов выступали переменные a, b, q, n, k) позволяет создавать удивительные алгоритмы.

Учитель должен ясно себе представлять, как сильно меняются подход и общий стиль решения задач (особенно математических), если при решении использовать схемы алгоритмов. Проиллюстрируем сказанное примером.

Известно, что изобретательность математиков позволила найти изящные формулы для подсчета суммы первых нечетных чисел, суммы их квадратов и кубов:

$$1+3+5+\ldots+(2n-1)=n^2;$$

$$1^2+3^2+5^2+\ldots+(2n-1)^2=\frac{n\,(4n^2-1)}{3};$$

$$1^3+3^3+5^3+\ldots+(2n-1)^3=n^2\,(2n-1).$$

Вывести эти формулы непросто. А как же тогда решать задачу суммирования, если в сумму включаются числа самого общего вида: $(3n-1)^2$, $(5n+3)^3$ — или какие-либо иные.

Решение простое: следует отказаться от поиска общей расчетной формулы, а искомым решением считать схему алгоритма.

Примером такой схемы может быть такая, как приведенная на рисунке 11.

Выделены (выписаны справа) возможные типы суммируемых чисел, их множество.

В заключение заметим, что изучение схем алгоритмов раз-

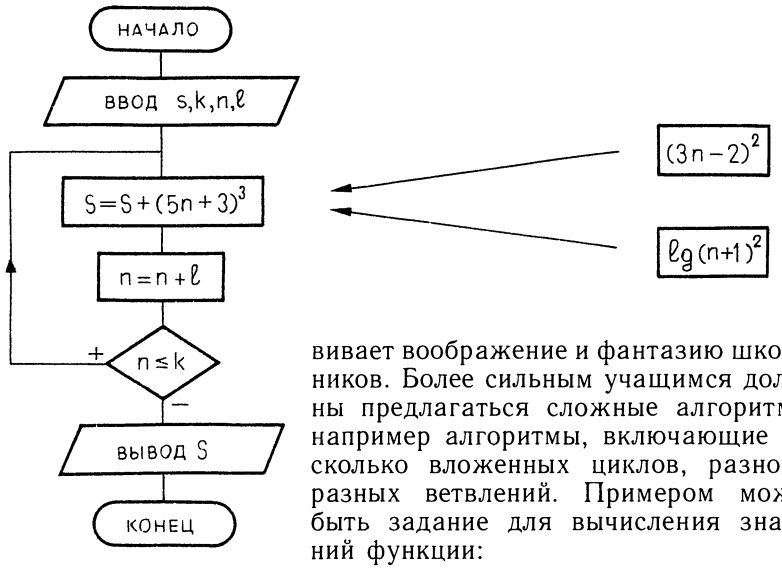


Рис. 11

вивает воображение и фантазию школьников. Более сильным учащимся должны предлагаться сложные алгоритмы, например алгоритмы, включающие несколько вложенных циклов, разнообразных ветвлений. Примером может быть задание для вычисления значений функции:

$y=|(2\cdot|x-1|-3)|$ и $y=|x-|3x-4||$.

Поручая школьникам анализ схем алгоритмов, учитель должен требовать от них четкого понимания того, какая из величин выступает в роли аргумента и какие значения аргумент может принимать.

§ 14. АЛГОРИТМИЧЕСКАЯ СИСТЕМА ПОСТА

Э. Пост — математик из США, специализировавшийся в области математической логики, был одним из первых, кто осознал важность такого понятия, как алгоритмическая полнота системы операций, используемых для преобразования информации.

Свои идеи Э. Пост опубликовал в 1935 г., задолго до возникновения автоматических преобразователей информации. Однако подходы Э. Поста к проблеме преобразования информации настолько фундаментальны, что через 30 лет, в 1967 г., к ним вернулись, чтобы пересказать их с новых позиций, использовать их в создании теории и практики обучения человека сотрудничеству с уже появившимися ЭВМ.

Речь идет о том, что в статье для журнала «Математика в школе» профессор Московского университета В. А. Успенский ввел понятие «машина Поста» и использовал этот им же придуманный термин для того, чтобы образнее и основательнее рассказать учителям об алгоритмической системе Э. Поста.

Профессор В. А. Успенский полагал, что рассказ о воображаемой машине Поста и рассказ о том, как ею управлять, помогут читателям разобраться в алгоритмической системе Э. Поста. Сам же Э. Пост никогда в своих работах термина «машина» не употреб-

лял и даже не подозревал, что его идеи можно интерпретировать так, как это сделал В. А. Успенский.

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

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

Идея В. А. Успенского оказалась очень привлекательной для педагогов. Предложение В. А. Успенского начинать ознакомление учащихся с программированием через изучение машины Поста было проверено на многих экспериментальных занятиях в первой в нашей стране школе юных кибернетиков. Такая школа, в которой занимаются ребята VII—IX классов, работает в Крыму. Опыты оказались успешными: школьники VII и даже VI классов с увлечением занимались программированием для воображаемой машины Поста и своими успехами подтвердили педагогическую гипотезу В. А. Успенского. Однако программирование на бумаге без возможности увидеть работу «живой», действующей машины хотелось бы закрепить программированием «за пультом». Для этого оставалось создать машину Поста в металле.

И такая машина была разработана. Первый экземпляр машины Поста был изготовлен в Симферопольском государственном университете в 1970 г. Позже эти простейшие ЭВМ с программным управлением выпускались на Украине.

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

Э. Пост еще в 1935 г. разработал простейшую из вообще возможных систем обработки информации и показал, что предлагаемая им система обладает весьма важным свойством — *алгоритмической полнотой*.

В чем же суть его системы?

Во-первых, Э. Пост предложил всякую информацию, подлежащую обработке по существу, предварительно преобразовывать по форме. Он требовал каждое слово, данное для обработки, предварительно переписать с привычного алфавита на алфавит двоичный, т. е. на двубуквенный алфавит. С подобной формаль-

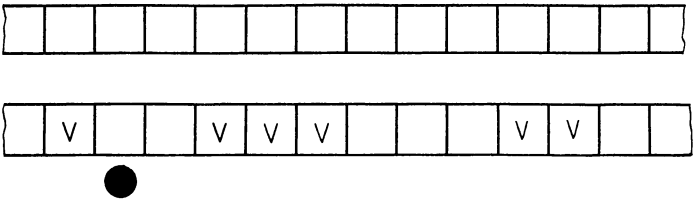


Рис. 12

ной обработкой текста мы встречаемся при обращении к телеграфу. Телеграфист, не меняя смысла, содержания текста, заменяет его форму. При этом телеграфист использует двубуквенный алфавит: его буквы — точка и тире.

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

Второй важной частью его системы является предложение обрабатывать каждое данное слово (уже записанное в двоичном алфавите) побуквенно: букву за буквой. Это приводит к очень простой системе элементарных действий, используя которые можно осуществить любое преобразование двоичного слова, если это вообще возможно.

Как же устроена машина Поста?

Сначала мы рассмотрим, как описал ее В. А. Успенский. После того как определенная информация переписана в двоичном алфавите, она буква за буквой заносится на информационную ленту машины. Информационная лента разбита на одинаковые секции (рис. 12). Число секций на ленте бесконечно. Это значит, что по мере необходимости мы можем подклеивать к информационной ленте справа или слева нужное число секций.

В качестве одной из букв двоичного алфавита, придуманного В. А. Успенским, будем использовать метку ($\vee$), которая может быть помещена в какую-либо секцию ленты; второй буквой будет пустота в секции. Если секция содержит метку, то она называется отмеченной, в противном случае она содержит пустоту и называется неотмеченной. Никаких других знаков, кроме метки и пусто, в секциях информационной ленты не может помещать в процессе работы ни человек, ни машина.

В показанной на рисунке 12 ленте часть секций отмечена. Распределение отмеченных и неотмеченных секций определяет состояние ленты. Вдоль информационной ленты движется каретка (заштрихованный кружок). Она может двигаться только скачками, каждый раз либо на одну секцию вправо, либо — влево. На рисунке каретка расположена под неотмеченной секцией.

Конкретное состояние ленты с указанием, где расположена каретка, определяет состояние машины. С помощью каретки машина может распознавать, является ли конкретная секция ленты, под которой расположена каретка, отмеченной или неотме-

ченной. Каретка может стереть метку, если она имеется в секции, может поместить метку в пустую секцию. С помощью каретки осуществляется побуквенное преобразование конкретного двоичного слова.

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

Система команд машины Поста содержит всего шесть команд:

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

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

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

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

$a!$ — команда «стоп»;

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

Рассмотрим пример использования команд для решения простой задачи. Располагая возможностями машины, нужно составить программу, работая по которой машина сотрет левую метку (под ней находится каретка) и присоединит ее к меткам, расположенным на информационной ленте (рис. 13).

Приводим текст программы:

1. $\updownarrow 2$ — выполнив команду с номером 1, машина сотрет метку и перейдет к выполнению команды с номером 2.
2. $\rightarrow 3$ — сдвиг вправо и переход к команде с номером 3.
3. $? \begin{matrix} 2 \\ 4 \end{matrix}$ — принятие решения: если обозреваемая секция пуста, то переход к команде с номером 2, в противном случае — с номером 4.
4. $\leftarrow 5$ — сдвиг влево.
5. $\vee 6$ — выполняется команда 5: машина ставит метку в пустую секцию.
6. ! — остановка машины.

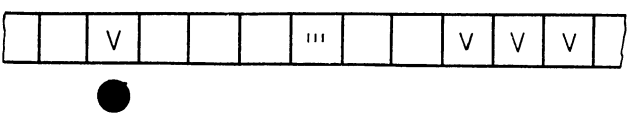


Рис. 13

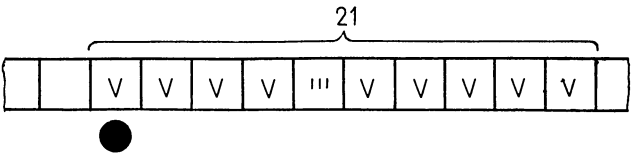


Рис. 14

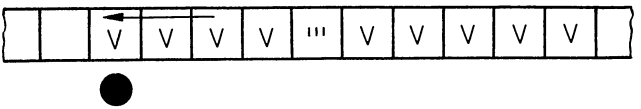


Рис. 15

Для того чтобы привлечь внимание читателя к удивительным возможностям машины Поста, расскажем о том, как она может быть соперником человека в настольной игре. Оказывается, простота системы команд не ограничивает возможностей машины в целом.

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

Вместо 21-го предмета используем информационную ленту машины, на которой отметим массив из 21-й метки (рис. 14). Каретку пометим под самой левой меткой. Потребуем от человека, чтобы при своем ходе он сначала решил, сколько меток он намерен стереть, и лишь после этого стирал метки в секциях так, как это показао на рисунке 15. В игру человек вступает всегда первым.

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

Что касается идеи алгоритма, то она очень проста. После хода соперника следует стирать столько меток, чтобы обоими соперниками вместе было стерто пять меток. Тогда при очередном своем ходе соперник будет вынужден начать стирать метки в следующей пятерке и, следовательно, будет стирать и последнюю, 21-ю метку.

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

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

1.? $\begin{matrix}\nearrow 2\\ \searrow 1\end{matrix}$ — очень интересно начало программы. Машина ждет появления неотмеченной секции над кареткой: это для нее будет обозначать, что человек завершил свой ход и очередь за ней.

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

6. ↕ 7 — стирается метка в секции.

7. ← 8 — каретка движется влево и стирает все не стертые человеком метки до того момента, пока не встретит пустую секцию.

8.? < 9, 6

9. → 10 — каретка сдвигается вправо под отмеченную секцию и «ждет» очередного хода человека.

10.? < 9, 1

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

Программирование по Посту (а лучше сказать, по Успенскому, ибо ему принадлежит основная педагогическая идея такого введения в пограммирование) содержит в себе все основные детали профессионального программирования.

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

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

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

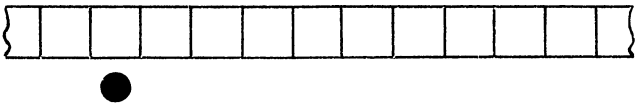


Рис. 16

можные ситуации и, лишь убедившись в безошибочности системы ходов, выписывает текст «окончания».

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

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

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

Пример. Задача В. А. Успенского.

На информационной ленте (рис. 16) либо вправо, либо влево от секции, под которой расположена каретка, находится массив меток. Расстояние до массива выражается конечным числом. Необходимо составить программу, работая по которой машина Поста найдет этот массив и присоединит к нему метку. Программист начинает и выигрывает.

Попробуйте составить текст программы.

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

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

В заключение приведем несколько заданий для школьников и приглашаем учителя попробовать попрограммировать вместе со своими учениками в системе Поста.

1. На ленте машины Поста отмечен массив из N меток. Каретка расположена под левой отмеченной секцией. Какой вид будет иметь лента после окончания работы машины по приводимой ниже программе?

1. → 2
2. ↑ 3
3. ? < 4, 2
4. ← 5
5. ∨ 6
6. !

2. На ленте на расстоянии в N секций расположены две метки. Ниже приводится программа работы машины Поста. Каретка расположена под правой отмеченной секцией. Какой вид будет иметь лента после окончания работы?

1. ← 2
2. ? < 3, 4.
3. ∨ 1
4. !

3. На ленте расположены две отмеченные секции, находящиеся на расстоянии в N секций друг от друга. Каретка расположена под левой из отмеченных секций. Какую работу выполнит машина Поста, действуя по приводимой ниже программе?

1. → 2
2. ? < 1, 3
3. ← 4
4. ? < 3, 1

4. На ленте расположен массив в N отмеченных секций. Необходимо справа от данного массива через одну пустую секцию разместить массив вдвое больший (он должен состоять из 2N меток). Предложите программу для решения данной задачи. Исходный массив может быть стерт.

5. На ленте расположен массив, состоящий из 2N—1 меток. Составить программу отыскания средней метки и стирания ее.

6. На ленте дан массив, содержащий 2N меток. Составить программу, действуя по которой машина раздвинет на расстояние в одну секцию две половины данного массива.

7. На ленте расположен массив из N меток. Составить программу, действуя по которой машина выяснит, делится ли число N на 3.

8. На ленте два массива расположены на расстоянии в одну секцию. Левый массив содержит M меток, правый — N меток. Левый больше правого. Найти разность. По окончании работы машины на ленте должно остаться M — N меток.

9. На ленте два массива в M и N меток. Составить программу выяснения, одинаковы ли массивы по длине.

10. На ленте расположены два массива. Составить программу стирания большего.

§ 15. АЛГОРИТМИЧЕСКАЯ СИСТЕМА ТЬЮРИНГА

Знакомство с системой, предложенной английским математиком А. Тьюрингом, позволит учителю и его сильным ученикам познакомиться с необычной формой представления алгоритма. Алгоритм будет задаваться в виде схем так называемых машин Тьюринга.

Рассказ о системе Тьюринга, машине Тьюринга и методике разработки таких машин (машин-алгоритмов) построим как рассказ о задаче.

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

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

Машина Тьюринга — это еще одна форма существования алгоритма, еще один способ фиксирования и «изображения» агоритма. Термин «машина Тьюринга» — это очень интересная форма существования алгоритма, и он так назван в честь английского математика А. Тьюринга.

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

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

Мы уже говорили о том, что профессор В. А. Успенский, разрабатывая машину Поста, не ставил перед собой инженерной задачи. Его описание машины с инженерной точки зрения было далеко не полным. Он ограничился рассказом только о тех деталях ее конструкции, которые потребуются пользователю машины — программисту. В данном случае ситуация аналогична. Описывая машину Тьюринга, мы ограничимся принципами работы машины и научимся «конструировать ее на бумаге».

Машин Тьюринга можно сконструировать много. Поскольку каждая из них есть машина Тьюринга, то она будет иметь много общего с другими. С другой стороны, каждая машина Тьюринга есть специализированная машина. Это значит, что она чем-то

отличается от других. Заметим, что описание машины Тьюринга напомнит вам описание машины Поста. У них есть общие черты и есть отличия. В дальнейшем мы покажем, что машину Поста можно рассматривать как частный случай машины Тьюринга.

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

Итак, что общего в различных машинах Тьюринга?

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

В отличие от машины Поста каретка машины Тьюринга может находиться в процессе работы машины в различных состояниях, меняя их скачкообразно (смысл термина «состояние» будет подробно раскрыт ниже). Для обозначения конкретного состояния мы будем использовать символы так называемого внутреннего алфавита машины:

$$Q=\{q_0,\ q_1,\ q_2,\ \ldots,\ q_n\}.$$

Внутренний алфавит конкретной машины Тьюринга есть конечный набор символов q_i.

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

Другим важным отличием машин Тьюринга от машин Поста является то, что для каждой конкретной машины Тьюринга проектируют ее собственный внешний алфавит. Символы внешнего алфавита могут распознаваться и выписываться кареткой. Пост использует единый стандартный для любой задачи двоичный алфавит. Тьюринг для каждой машины определяет алфавит заново. Одна машина Тьюринга может отличаться от другой внешним алфавитом.

Алфавит может быть таким:

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

или таким:

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

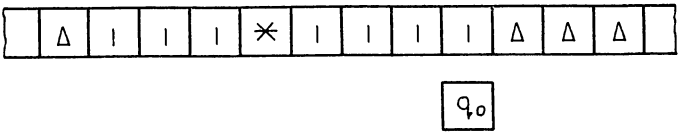


Рис. 17

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

Приведем несложный пример конструирования машины Тьюринга.

На информационной ленте машины Тьюринга (рис. 17) имеется два массива палочек (|), разделенных звездочкой (*). Необходимо разработать функциональную схему машины Тьюринга, которая крайнюю справа палочку правого массива сотрет и запишет ее вместо звездочки. Каретка в состоянии q_0 обозревает крайнюю справа палочку. Символом Δ обозначены пустые секции ленты.

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

Помним о том, что наши возможности ограничены: мы можем использовать умение каретки двигаться скачками, умение распознавать символы внешнего алфавита в обозреваемой секции информационной ленты, умение заменять один из символов внешнего алфавита на любой другой из этого же алфавита. Наконец, мы можем использовать возможность смены состояния каретки. Конечно, мы можем и остановить работу в необходимый момент.

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

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

Пример 1. α П q_7 — выполняя эту команду, машина должна в обозреваемую секцию поместить символ α, затем сдвинуться на один шаг вправо и сменить свое предшествующее состояние на состояние q_7.

Пример 2. 5 Л q_0 — в обозреваемую секцию вписывается «5», делается шаг влево, и состояние меняется на q_0.

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

Таблица 1

Символ внешнего алфавита	Символы внутреннего алфавита	
	q_0	q_1
|	$\Delta Л q_1$	Л
*		| !
Δ		

В самом левом вертикальном столбце выписаны символы внешнего алфавита: $A=\{*, |, \Delta\}$.

В верхней горизонтальной строке — символы внутреннего алфавита: $Q=\{q_0, q_1\}$.

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

Как же работает машина, имеющая такую функциональную схему?

В «поле зрения» машины находится в начальный момент секция с помещенной в ней палочкой (см. рис. 17). Иначе говоря, машина в состоянии q_0 обозревает |. В таблице 1 на пересечении строки с символом | и столбца с символом q_0 мы обнаруживаем команду $\Delta Л q_1$. Выполняя эту команду, машина сотрет палочку (заменит ее символом Δ — пусто), каретка сдвинется влево и сменит состояние q_0 на q_1.

На втором такте работы (рис. 18) каретка в состоянии q_1 обозревает палочку |. Команду, которую машине придется исполнять, находим на пересечении первой строки и второго столбца схемы. Эта команда представляет собой указание Л — «сдвинься влево». Указания о смене обозреваемого символа нет, нет и указания о смене состояния, а это значит, что ничего менять не следует.

Сдвинувшись влево в том же состоянии q_1, каретка вновь будет обозревать секцию, содержащую символ-палочку |, и вновь ей придется исполнять ту же самую команду — «сдвинься влево». Так будет продолжаться до тех пор, пока, сдвигаясь влево, каретка в состоянии q_1 не окажется под секцией со звездочкой (*).

На следующем такте каретка выполнит команду, помещенную во второй строке схемы на пересечении со вторым столбцом:

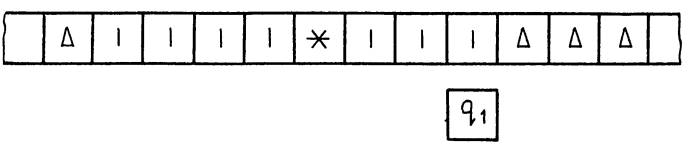


Рис. 18

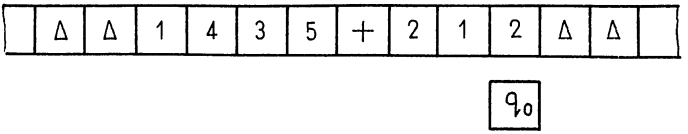


Рис. 19

| !. Эта команда расшифровывается так: «замени * на | и остановись».

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

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

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

Проектирование машины начнем с определения нужного нам внешнего алфавита. Так как данное число и сумма есть числа восьмеричной системы счисления, то нам потребуются цифры 0, 1, 2, 3, 4, 5, 6, 7.

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

Алфавит, следовательно, такой:

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

Изобразим условно на ленте данные задачи (рис. 19). Справа — число троичной системы счисления, слева — число восьмеричной. Каретка в состоянии q_0 обозревает цифру младшего разряда правого числа.

Идея решения задачи может быть такой: от числа, расположенного справа, отнимаем единицу (действуем, конечно, по правилам троичной арифметики), затем продвигаемся влево и к восьмеричному числу прибавляем единицу (действуем уже по правилам восьмеричной арифметики). После этого возвращаемся к правому числу и повторяем первый цикл. Работа будет продолжаться до тех пор, пока число, расположенное справа, не будет исчерпано. После того как к левому числу прибавляем единицу, в последний раз мы сдвигаемся вправо, стираем знак «+» и останавливаемся.

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

Таблица 2

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

Не правда ли, оригинальное решение? Как естественно выглядит процедура отыскания суммы чисел.

Приведенный пример можно интерпретировать и таким образом: пусть необходимо сконструировать машину Тьюринга, которая сможет выступить в роли троично-восьмеричного дешифратора. Сконструированная выше машина сможет это сделать — достаточно левое слагаемое считать равным нулю. Тогда, уменьшая правое троичное число единица за единицей, мы построим слева от знака «+» новое число, равное уничтоженному, но записанное уже в восьмеричной системе.

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

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

З а д а ч а.

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

Глава III. ПРЕОБРАЗОВАНИЕ ИНФОРМАЦИИ НА ЭВМ

§ 16. ПРЕОБРАЗОВАНИЕ ИНФОРМАЦИИ — ПОСТАНОВКА ЗАДАЧИ

Прежде чем приступить к преобразованию информации, ее следует получить. Если говорить о человеке, то он получает информацию с помощью своих органов чувств. В этом случае речь идет о том, что человек слышит, видит, осязает, обоняет, ощущает температурный уровень среды.

Известны диапазоны, в которых действуют органы чувств; за пределами чувствительности органов чувств человек применяет специальные датчики. В результате возникает возможность видеть в темноте, слышать в диапазоне ультразвуковых колебаний, ощущать радиационную опасность и многое другое.

Учитель, ведущий занятия по теме «Информация и ее преобразование», должен быть подготовлен к проведению беседы на тему «Современные методы сбора информации из внешней среды». Преподаватель должен расширить и уточнить представление учащихся о методах получения информации в микромире — и они услышат о возможностях современной микроскопии. Учитель должен рассказать своим ученикам о методах сбора информации, поступающей из космоса,— и они узнают об удивительных гигантских сооружениях для связи с искусственными космическими объектами и с далекими галактиками.

Учитель должен рассказать своим ученикам о таких приемах сбора информации, как компьютерная томография, о методах, основанных на использовании магнитного резонанса и ультразвука. Наиболее яркие примеры таких подходов дает медицина.

При реализации последних из упомянутых методов большое место занимают многократные преобразования информации из одной формы в другую. Компьютерная томография, например, воспроизводит изображения, необходимые при диагностике, используя компьютеры для перевода двумерных рентгеновских снимков в цифровой код. Закодированная информация обрабатывается на ЭВМ, и в результате получаются трехмерные цветные изображения. Врач изучает внутренний орган в небывалых до этого условиях — в условиях удивительной информированности.

Этот пример подчеркивает важнейшую особенность преобразования информации, предъявляемой с какой-то целью ЭВМ. ЭВМ работает с представленной в двоичном коде информацией.

Учитель должен понимать, что какая бы природа физического сигнала ни рассматривалась, в конечном счете этот сигнал (звуковой, оптический, тепловой или сигнал, «несущий запах») должен быть квантован (раздроблен на разумные доли) и каждый квант должен быть закодирован.

Огромное число инженерных проблем, возникающих при этом, не принято обсуждать в рамках информатики. Речь идет о том, как непрерывный, например звуковой или электрический, поток следует квантовать.

В центре внимания информатики и той ее части, которая составляет фундамент школьного предмета, находятся проблемы работы с уже подготовленной к обработке на ЭВМ информацией.

Учитель должен ясно понимать, что есть некая граница в классификации задач преобразования информации. Есть такие задачи, которые можно условно считать задачами чисто внешнего преобразования информации. Примером может быть усиление речи, сжатие изображения во времени — это делается, чтобы развернуть быстро протекающий процесс в удобную для демонстрации и изучения форму. Пуля, мгновенно пробивающая стекло, на экране может двигаться медленнее черепахи. Напротив, медленное развитие какого-нибудь процесса можно позднее наблюдать в сжатые сроки как процесс, протекающий значительно быстрее.

Если же речь идет об операции вычислительной, например об операции извлечения квадратного корня, то преобразование исходной информации (число, из которого извлекается корень) в новую информацию (величина извлеченного корня) затрагивает смысл исходной информации. И такое преобразование мы относим к более значительному типу преобразования информации.

Вообще, преобразование числовой информации, решение вычислительных задач, несомненно, самый важный тип преобразования.

§ 17. ПРЕОБРАЗОВАНИЕ ЧИСЛОВОЙ ИНФОРМАЦИИ

Математическое моделирование — один из важнейших видов моделирования. Разнообразен мир математических моделей. Уже в средней школе изучаются модели в виде уравнений и неравенств первой, второй, а иногда и третьей степени.

Школьники учатся исследовать и решать системы линейных уравнений и неравенств. В форме простейшего дифференциального уравнения предстает модель радиоактивного распада или гармонического колебания.

Анализ математических моделей требует подчас огромного числа вычислений — без использования ЭВМ обойтись невозможно.

Учитель, ведущий преподавание информатики, имеющий выход на ЭВМ, может обогатить курс математики сопутствующими сведениями о значении математического моделирования, демонстри-

руя ЭВМ в работе над математическими моделями конкретного содержания, подчеркивая этим роль моделирования в системе познания мира — познания, которому учит детей школа.

Подлинная культура вычислений на ЭВМ требует от учителя ясного представления основ вычислительной математики, реализации вычислений на ЭВМ. И приобретать такую культуру следует с расширения представлений о способах записи числовой информации, с уточнения понятия о системах счисления.

Здесь следует упомянуть о том, что у некоторых педагогов складывается несколько ограниченная позиция по проблеме изучения систем счисления. Предполагается, что достаточно познакомиться с сильно взаимосвязанными системами счисления: с основаниями $q=2$, $q=8$ и $q=16$.

Столь узкий взгляд, несомненно, должен быть осужден. Дело в том, что в древнейшей области математики — в области арифметических вычислений над целыми числами — и сегодня появляются нетривиальные, воздействующие на практику, на прикладную математику идеи и методы. Еще больше идей несут в себе современные подходы в арифметике действительных чисел.

Учитель информатики должен знать об особенностях таких систем счисления, как системы с симметричным основанием. Достаточно напомнить, что в 1969 г. в Советском Союзе была выпущена партия уникальных ЭВМ серии «Сетунь», арифметика которой базировалась на троичной уравновешенной системе счисления. Теоретический спор о достоинствах этой системы в сравнении с двоичной не закончен. В чем суть этого спора, учителю знать неплохо.

Не может учитель не знать и о классе машин, реализующих систему счисления остаточных классов. Математические основы этой системы связаны с задачей о восстановлении натурального числа по известным остаткам от деления его на несколько других натуральных чисел и восходят к далекому прошлому китайской математики.

Нельзя согласиться с тем, что в курс информатики не включаются основы арифметики нормализованных чисел. О какой подлинной культуре точных вычислений можно вести речь, если школьник не разбирается хорошо в понятиях «мантисса» и «порядок» нормализованного числа!

Уже этих весьма кратко сформулированных соображений достаточно, чтобы привлечь внимание учителя к вопросам основ машинной арифметики.

§ 18. СОВРЕМЕННЫЕ ПРЕДСТАВЛЕНИЯ О СИСТЕМАХ СЧИСЛЕНИЯ

Развитие вычислительной техники, появление быстродействующих программно-управляемых электронных вычислительных машин, возникновение искусства программирования — все это по-

требовало глубокого и целенаправленного изучения десятичной и других систем счисления.

Перед математиками и конструкторами вычислительных машин в 50-х гг. встала проблема отыскания таких систем счисления, которые отвечали бы запросам прикладного программирования и требованиям, идущим от разработчиков новых вычислительных устройств. В результате активных поисков в очень короткое время наши представления о системах счисления и методах вычислений оказались значительно измененными. Выяснилось, что одно из древнейших интеллектуальных умений — арифметический счет — даже в наше время может совершенствоваться, подчас весьма неожиданно и на удивление эффективно.

Знакомство с историей поисков и достигнутыми результатами в этой области математики учителю необходимо прежде всего потому, что арифметический счет, современно организуемый, есть базис всей вычислительной математики.

В число тем, комментируемых ниже, включены:

1. Позиционный принцип в системе счисления.
2. Унарная система счисления.
3. Двоичная система счисления.
4. Двоичная арифметика.
5. Связь между системами счисления. Связь между двоичной системой счисления и другими системами счислений.
6. Уравновешенные системы счисления.
7. Арифметика троичной уравновешенной системы счисления.
8. Понятие о смешанных системах счисления.

1. Позиционный принцип в системе счисления. Под системой счисления принято понимать совокупность приемов обозначения (записи) чисел. Конечно, изучение записи и изучение чтения чисел взаимосвязаны.

Наиболее совершенными являются позиционные системы счисления, т. е. системы, в которых значение каждой цифры в изображении числа зависит от ее положения (позиции) в ряду других чисел, изображающих число.

Учитель должен привлечь внимание учащихся к понятию «базис системы счисления».

Базис системы счисления — это последовательность так называемых ключевых чисел, каждое из которых задает значение цифры «по месту». Наряду с широкоизвестными системами счисления, базис которых образуют члены геометрических прогрессий, существуют так называемые смешанные системы счисления.

Примеры:

Базис двоичной системы счисления: ..., 2^n, ..., 16, 8, 4, 2, 1.

Базис восьмеричной системы счисления: ..., 8^n, ..., 64, 8, 1.

Или в более общем виде: ..., $q_n = q^n$, ..., $q_3 = q^3$, $q_2 = q^2$, $q_1 = q$, $q_0 = 1$,

где $q \in N$ и $q \neq 1$.

Число q называют основанием системы.

Каждое число в любой из таких систем может быть записано в цифровой и многочленной форме:

$$A_q = \overline{a_n a_{n-1} \ldots a_2 a_1 a_{0q}};$$

$$A_q = a_n \cdot q^n + a_{n-1} \cdot q^{n-1} + \ldots + a_2 q^2 + a_1 q + a_0.$$

Примеры:

Факториальная система: ..., $n!$, $(n-1)!$, ..., 2!, 1!

Фибоначчиева система: ..., 21, 13, 8, 5, 3, 2, 1.

Более детальные сведения о смешанных системах счисления будут приведены ниже. В данном месте только привлекается внимание к тому, что числа, образующие базис, необязательно образуют геометрическую прогрессию.

Подчеркнем, что имеется ряд смешанных систем счисления, имеющих практическое значение.

2. Унарная система счисления. На первый взгляд кажется странным обращение к унарной системе счисления. Так ли уж часто мы прибегаем к прибавлению единицы?

Так ли уж важно вспоминать столь далекие времена человеческой культуры?

На наш взгляд, это важно.

Под унарной системой понимают систему счисления, в которой для записи чисел применяется только один вид знаков — палочка. Каждое число в такой системе обозначается с помощью строки, составленной из отдельных палочек. Количество палочек равно изображаемому числу. Из сказанного ясно, что унарная система счисления есть предельный случай позиционной системы счисления с основанием $q=2$.

От палочки или мешочка с камешками до компьютера — вот какой огромный путь прошел человек, совершенствуясь в искусстве счета.

Ведя обсуждение важных для себя дел, человек постепенно начинал использовать числа, связывая числительное с объектом: «пять овец», «пять дней», «пять братьев». Постепенно возникло абстрактное понятие числительного — «пять», просто 5.

На этом этапе, по существу, имело место зарождение первых абстракций, появление первых математических моделей. Мешочек с камешками — модель стада овец, палочка с нанесенными на ней засечками — модель календаря.

Осознав эти свои достижения, человек получил возможность для более свободного манипулирования данными, облегчил себе учет, научился делать первые шаги в планировании.

Разве об этом в нескольких словах учитель не должен рассказывать детям?

Используется ли унарная система счисления сейчас?

Да, используется! Попробуйте задать вопрос школьникам: как к семи прибавить пять?

Мысль школьника уведет его в I класс к счетным палочкам, к унарной системе счисления. Обратите внимание: в са-

мом фундаменте арифметики эта система является мощным педагогическим средством, вводит школьников в мир счета.

3. Двоичная система счисления. Двоичная система счисления, т. е. система, основание которой $q=2$, образно говоря, является минимальной системой, в которой реализуется принцип позиционности в цифровой форме записи числа. В двоичной системе счисления значение каждой цифры по месту при переходе от любого данного разряда к следующему старшему увеличивается вдвое.

К двоичной форме записи числа приводит пересчитывание предметов, предварительно группируемых в пары.

История развития двоичной системы счисления — одна из ярких страниц в истории математики.

Официальное «рождение» двоичной арифметики связывают с именем Г. В. Лейбница: он в 1703 г. опубликовал статью Memoires de L'Academie Royale des Siences), в которой были рассмотрены правила выполнения всех арифметических операций над двоичными числами.

Следует иметь в виду, что Г. Лейбниц не рекомендовал двоичную систему для практических вычислений: он считал ее полезной лишь при рассмотрении теоретических вопросов. В частности, он полагал, что закономерности в последовательностях из многих цифр можно обнаружить, если эти последовательности записать в виде двоичных чисел, т. е. в виде множества единиц и нулей.

Так, по его предложению Я. Бернулли пытался отыскать закономерности в распределении цифр числа π, используя 35-значное двоичное приближение числа π. Из других современных применений двоичной системы счисления следует указать на использование ее в дискретной математике и при анализе игр и головоломок.

Авторский приоритет в публикации основных понятий этой системы, по свидетельству Д. Кнута, «принадлежит испанцу Х. Лобковицу, который довольно подробно рассмотрел представление чисел в системах счисления по основаниям 2, 3, 4, 5, 6, 7, 8, 9, 10, 12 и 60, но не привел никаких примеров арифметических операций, в недесятичных системах счисления» (Кнут Д. Искусство программирования для ЭВМ.— М.: Мир, 1977.— Т. 2.— С. 208).

До начала 30-х гг. XX в. двоичная система счисления оставалась вне поля зрения прикладной математики. Потребность в создании надежных и простых по конструкции счетных механических устройств и удивительная простота двоичной арифметики привели к более глубокому изучению двоичной системы как системы, пригодной для аппаратурной реализации.

Первые двоичные вычислительные механические машины были построены во Франции и Германии. Пионером в проектировании вычислительных устройств двоичного действия на электронно-ламповой основе является инженер Дж. Атанасов, болгарин по происхождению, проживающий в США. Одновременно с ним (1937)

двоичную машину, но на релейной (электромагнитной) основе спроектировал Дж. Штибиц. В 1941 г. немецкий инженер К. Цузе построил сначала механическую, а затем и релейную двоичную вычислительную машину.

Утверждение двоичной арифметики в качестве общепринятой основы при конструировании ЭВМ с программным управлением состоялось под несомненным влиянием работы А. Беркса, Х. Гольдстайна и Дж. фон Неймана о проекте первой ЭВМ с хранимой в памяти программой. Работа была написана в 1946 г.

В этой работе наиболее аргументированно обоснованы причины отказа от десятичной арифметики и переход к двоичной системе счисления как основы машинной арифметики.

4. Двоичная арифметика. Связь между двоичной системой счисления и другими системами счисления. Арифметика двоичной системы счисления, как и всякой другой позиционной системы, основывается на использовании таблиц сложения и умножения цифр. Эти таблицы чрезвычайно просты.

+	0	1
0	0	1
1	1	10

×	0	1
0	0	0
1	0	1

Главное достоинство двоичной системы в том, что сложение и умножение цифр легко моделировать. Существует немало удивительных по простоте счетных устройств для выполнения арифметических действий над двоичными числами. Для младших школьников непременно следует показать в работе двоичные счеты, на каждой проволочке которых всего две косточки (рис. 20).

Действия вычитания и деления чисел в двоичной арифметике можно выполнять по общепринятым для позиционных систем правилам. Однако, и это очень важно для практики, особенности двоичной системы счисления позволяют создать специфические алгоритмы вычитания и деления двоичных чисел, наиболее подходящие для аппаратурной реализации. Учителю рекомендуется познакомиться с такими интересными алгоритмами, как вычитание с использованием двоичного дополнения вычитаемого и т. д.

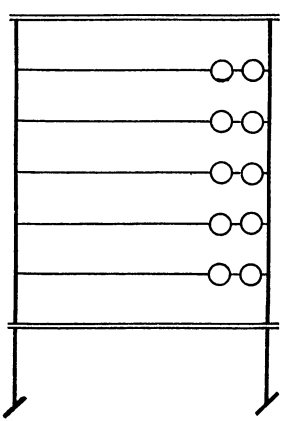


Рис. 20

Учитель должен понимать, что естественным обобщением двоичной системы являются системы с основаниями $q=8$ и $q=16$. Сегодня эти системы счисления

широко распространены. Краткие сведения об этих системах должны школьникам сообщаться.

История восьмеричной системы счисления не может сравниться по богатству событий с историей двоичной системы. В качестве любопытного курьеза обычно упоминают, что шведский король Карл XII в 1717 г. увлекался восьмеричной системой, считал ее более удобной, чем десятичная, и намеревался королевским указом ввести ее как общегосударственную. Неожиданная смерть помешала осуществлению столь необычного намерения.

Любопытно, что в 1862 г. американский инженер, швед по происхождению, Дж. Нистром выпустил в Филадельфии книгу «Проект новой арифметической и денежной системы, а также системы мер и весов, которую предполагается назвать тональной системой с основанием, равным шестнадцати». Он писал: «Я не боюсь и ничуть не колеблюсь выступить в защиту двоичной системы в арифметике и метрологии. Я знаю, на моей стороне природа; если мне не удастся убедить вас в ее полезности и чрезвычайной важности для человечества, это не сделает чести нашему поколению, нашим ученым и философам».

Сегодня эти мысли кажутся пророческими. Приведем цитату из работы еще одного замечательного французского математика и естествоиспытателя Б. Паскаля: «Десятичная система построена довольно неразумно, конечно — в соответствии с людскими обычаями, а вовсе не требованиями естественной необходимости, как склонно думать большинство людей».

Эти ссылки подтверждают педагогический тезис о необходимости расширения у школьников представлений о системах счисления.

Взаимосвязь между системами счисления с основаниями $q=2$, $q=8$ и $q=16$ полностью определяется двумя теоремами.

Теорема 1. Для записи целого двоичного числа в системе счисления с основанием $q=2^n$ достаточно данное двоичное число разбить на грани справа налево (т. е. от младших разрядов к старшим) по n цифр в каждой грани. Затем каждую такую грань следует рассмотреть как n-разрядное двоичное число и записать его как цифру в системе с основанием $q=2^n$.

Пример 1. Число 101100001000110010_2 заменить равным ему числом восьмеричной системы счисления, т. е. системы с основанием $q=2^3$.

В соответствии с теоремой 1 разбиваем число на грани по три цифры в каждой:

$$\begin{array}{cccccc} 101 & 100 & 001 & 000 & 110 & 010 \\ 5 & 4 & 1 & 0 & 6 & 2 \end{array}$$

Внизу под каждой из граней выписаны цифры, соответствующие трехразрядным двоичным числам:

$101_2=5_8$, $100_2=4_8$, $001=1_8$, $000_2=0_8$, $110_2=6_8$, $010_2=2_8$.

Теорема 2. Для замены целого числа, записанного в системе с основанием $p=2^n$, равным ему числом в двоичной системе счисления достаточно каждую цифру данного числа заменить n-разрядным двоичным числом.

Пример 2. Число 3567_8 заменить равным ему двоичным числом.

В соответствии с теоремой 2 запишем

3	5	6	7
011	101	110	111

В результате получим равенство

$3567_8=011101110111_2$.

Из сказанного следует, что замена двоичного числа на равное ему восьмеричное и наоборот может осуществляться механически, без всяких вычислений.

Нетрудно представить себе пишущую машинку, у которой на клавишах восьмеричные цифры: 0, 1, 2, 3, 4, 5, 6, 7 — и на молоточках, которые бьют по бумаге, соответствующие им трехразрядные двоичные числа: 000, 001, 010, 011, 100, 101, 110, 111. Такая машинка позволит всякое восьмеричное число, отстукиваемое на клавиатуре, отпечатать на бумаге в виде равного ему двоичного числа.

Замените клавиши на молоточки и наоборот — и будет изготовлена двоично-восьмеричная кодирующая машинка.

Доказательство двух теорем несложно, учитель может ограничиться доказательством для следующего случая: q и $p=q^3$.

Доказательство теоремы 1. Пусть дано

$$(abcdef)_q=a\cdot q^5+b\cdot q^4+c\cdot q^3+d\cdot q^2+eq+f=$$
$$=(aq^2+bq+c)\,q^3+(dq^2+eq+f)=$$
$$=A\cdot q^3+B=Ap+B=AB_p;$$
$$A_p=aq^2+bq+c=\overline{abc}_q;$$
$$B_p=dq^2+eq+f=\overline{def}_q.$$

Трехразрядные числа системы с основанием q записаны как цифры системы с основанием p.

Доказательство теоремы 2. Пусть дано

$$\overline{ABC}_p,$$

где $p=q^3$.

Имеем

$$ABC_p=Ap^2+Bp+c=(aq^2+bq+c)\,(q^3)^2+$$
$$+(dq^2+eq+f)\,q^3+(kq^2+lq+m)=aq^8+bq^7+cq^6+$$
$$+dq^5+eq^4+fq^3+kq^2+lq+m=\overline{abcdefklm}_q.$$

Заметим, что доказанные теоремы — это теоремы о взаимосвязи позиционных систем с основаниями q и $p=q^n$, где необязательно $q=2$.

Подчеркнем, что рассмотренные алгоритмы применимы и для замены правильных q-ичных систематических дробей на равные им правильные q^n-ичные систематические дроби и наоборот.

Описанные алгоритмы имеют большое практическое значение. Обмен информацией между узлами и устройствами большинства современных ЭВМ осуществляется путем передачи командных слов, которые, как правило, являются двоичными словами. Использование двубуквенного алфавита (0 и 1), как это уже подчеркивалось, диктуется инженерными требованиями. Однако пользоваться словами, записанными в двоичном алфавите, из-за большой длины отдельных слов (до 64 букв в слове) и своеобразной «зрительной однородности» текста человеку неудобно. Поэтому программисты и инженеры, обслуживающие ЭВМ, заменяют все «машинные слова» на эквивалентные им восьмеричные или шестнадцатеричные слова и числа.

В первом случае длина исходного слова сокращается в три раза, во втором — в четыре. Такие слова становятся более удобными для рассмотрения и запоминания.

5. Связь между системами счисления. Подчеркивая значительную роль систем с основаниями $q=2$, $q=8$ и $q=16$, учитель должен продвигать в сознание своих учеников мысль о том, что и другие из позиционных систем интересны и могут оказаться практически ценными.

Из многих вопросов, возникающих при изучении позиционных систем счисления, рассмотрим проблему замены целых чисел и правильных дробей какой-либо системы с основанием q равными числами системы с основанием p.

Алгоритм I. Для того чтобы исходное целое число A_q заменить равным ему целым числом B_p, необходимо число A_q разделить по правилам q-арифметики на новое основание p. Цифрами искомого числа B_p являются остатки от деления, выписанные так, чтобы последний остаток являлся бы цифрой старшего разряда числа B_p. (Ясно, что число p перед делением должно быть записано в системе с основанием q.)

Рассмотренный алгоритм пригоден для использования при любых p и q, но рекомендуется при $q>p$. Так целые десятичные числа, пользуясь именно этим алгоритмом, заменяют на равные им числа двоичные и восьмеричные.

Алгоритм II. Для того чтобы данное целое число A_q заменить равным ему числом B_p, достаточно цифру старшего разряда числа A_q умножить по правилу p-арифметики на основание q. К полученному произведению прибавить цифру следующего разряда числа A_q. Полученную сумму вновь умножить на q по правилам p-арифметики, вновь к полученному произведению прибавить цифру следующего (более младшего) разряда. Так поступают до тех пор, пока не будет прибавлена младшая цифра числа A_q.

Алгоритм II пригоден для любых p и q, однако рекомендуется при $p>q$.

Алгоритм III. Для того чтобы исходную правильную дробь $0, A_q$ заменить равной ей правильной дробью $0, B_p$, нужно $0, A_q$ умножить на «новое» основание p по правилам q-арифметики, целую часть полученного произведения считать цифрой старшего разряда искомой дроби. Дробную часть полученного произведения вновь умножить на p, целую часть полученного результата считать следующей цифрой искомой дроби.

Эти операции продолжать до тех пор, пока дробная часть не окажется равной нулю либо не будет достигнута требуемая точность.

Пример. Дробь $0{,}375_{10}$ заменить равной ей двоичной дробью.

Решение.

$0{,}375 \cdot 2 = 0{,}750$

$0{,}75 \cdot 2 = 1{,}50$

$0{,}5 \cdot 2 = 1{,}0$ (дробная часть равна 0)

$0{,}375_{10} = 0{,}011_2$

Этот алгоритм рекомендуется при $q > p$.

Алгоритм IV. Для того чтобы исходную правильную дробь $0, A_q$ заменить равной ей правильной дробью $0, B_p$, необходимо цифру младшего разряда дроби $0, A_q$ разделить на основание p по правилам p-арифметики, к полученному частному прибавить цифру следующего (более старшего) разряда и далее поступать так же, как и с первой взятой цифрой.

Эти операции продолжать до тех пор, пока не будет прибавлена цифра старшего разряда искомой дроби. После этого полученную сумму разделить еще раз на p и к результату приписать запятую и нуль целых.

Пример. Дробь $0{,}1101_2$ заменить равной ей десятичной правильной дробью.

Решение. Составляем вспомогательную таблицу.

0	1	1	0	1
$0{,}8125_{10}$	$1{,}625_{10}$	$1{,}25_{10}$	$0{,}5_{10}$	1

$1:2+0=0{,}5_{10}$

$0{,}5:2+1=1{,}25_{10}$

$1{,}25:2+1=1{,}625_{10}$

$1{,}625:2+0=0{,}8125_{10}$

Имеем $0{,}1101_2 = 0{,}8125_{10}$.

Алгоритм рекомендуется при $p > q$.

6. Уравновешенные системы счисления. Обобщение понятия «позиционная система счисления» может осуществляться не только за счет того, что под основанием системы понимаются иные, не равные десяти натуральные числа.

Учитель, среди учеников которого есть способные школьники, может с этими школьниками рассмотреть так называемые уравновешенные системы счисления. Целесообразность такой работы будет показана ниже.

Занятия со школьниками можно начать с рассмотрения известной задачи «о взвешивании». Эта задача была известна еще Л. Пизанскому (Фибоначчи) в 1215 г. Иногда эту задачу называют задачей Баше и задачей Менделеева.

З а д а ч а. Найти набор из четырех гирь такой, чтобы с их помощью на чашечках равноплечных весов можно было бы взвесить груз массой от 1 до 40 кг включительно. Гири можно располагать на любой из чашек весов.

Р е ш е н и е. Искомый набор равен 1, 3, 9 и 27 кг.

Составляем таблицу результатов взвешиваний.

27	9	3	1	
0	0	0	0	0 кг
0	0	0	1	1 кг
0	0	1	$\bar{1}$	2 кг
0	0	1	0	3 кг
0	0	1	1	4 кг
0	1	$\bar{1}$	$\bar{1}$	5 кг
0	1	$\bar{1}$	0	6 кг
0	1	$\bar{1}$	1	7 кг
	...			...
1	1	$\bar{1}$	1	34 кг
1	1	0	$\bar{1}$	35 кг
	...			...
1	1	1	1	40 кг

Для записи результатов использовались значки: 0,1 и $\bar{1}$, каждая из этих цифр принимает значение «по месту». При переходе от разряда к разряду вес цифры изменяется в три раза.

Из сказанного заключаем, что запись результатов взвешивания ведется в позиционной системе счисления с основанием $q=3$; среди цифр системы цифра $\bar{1}$ означает минус единицу: ведь наличие цифры $\bar{1}$ (единица с чертой) в каком-либо разряде означает, что гиря этого веса вычитается из общей суммы.

П р и м е р ы:

$1\bar{1}0\bar{1}_3 = 1\cdot 3^3 - 1\cdot 3^2 + 0\cdot 3 - 1 = 17_{10}$;

$10\bar{1}\bar{1}_3 = 1\cdot 3^3 + 0\cdot 3^2 - 1\cdot 3 - 1 = 23_{10}$.

О п р е д е л е н и е. *Уравновешенной троичной системой счисления или троичной системой с симметричным основанием называется система с основанием* $q=3$ *и цифрами, принадлежащими множеству* $\{1, 0, \bar{1}\}$, *где цифра* $\bar{1}$ *означает* -1 *(минус единицу).*

Можно записать и числа, начинающиеся с цифры $\bar{1}$:

$\bar{1}111_3 = -1 \cdot 3^3 + 1 \cdot 3^2 + 1 \cdot 3 + 1 = -14_{10}$;

$\bar{1}11\bar{1}_3 = -1 \cdot 3^3 + 1 \cdot 3^2 - 1 \cdot 3 - 1 = -22_{10}$.

Числа, начинающиеся с цифры 1, положительны, а числа, начинающиеся с цифры $\bar{1}$, отрицательны.

Учитель должен обратить внимание учащихся на то, что в этой системе счисления нет знаков перед числами. Далее будет показано, что при выполнении арифметических действий никаких правил знаков нет, которые столь важны и, к сожалению, трудны в нашей арифметике.

7. Арифметика троичной уравновешенной системы счисления. Приводим таблицы сложения и умножения цифр в троичной уравновешенной системе.

+	1	0	$\bar{1}$
1	$1\bar{1}$	1	0
0	1	0	$\bar{1}$
$\bar{1}$	0	$\bar{1}$	$\bar{1}1$

×	1	0	$\bar{1}$
1	1	0	$\bar{1}$
0	0	0	0
$\bar{1}$	$\bar{1}$	0	1

Примеры:

$$\begin{array}{r} 1\bar{1}01_3 \\ +\ \bar{1}\bar{1}1_3 \\ \hline 10\bar{1}_3 \end{array} \qquad \begin{array}{r} \bar{1}1_3 \\ \times\ 1\bar{1}_3 \\ \hline \bar{1}1 \\ +\ \bar{1}1 \\ \hline \bar{1}\bar{1}_3 \end{array}$$

Во втором примере умножалось число положительное ($1\bar{1}$) на отрицательное ($\bar{1}1$), результат получился ($\bar{1}\bar{1}$) отрицательным. Никаких правил знаков не применялось.

Важным, практически весьма и весьма значимым, достоинством рассматриваемой системы является простота алгоритма для получения числа противоположного данному: достаточно все цифры 1 исходного числа заменить на $\bar{1}$, а все цифры $\bar{1}$ — на цифры 1.

Пример:

Исходное число: $1\bar{1}01\bar{1} = 56_{10}$;

Число противоположное: $\bar{1}10\bar{1}1 = -56_{10}$.

Из сказанного следует, что вычитание в этой арифметике становится чрезвычайно простой операцией. Вычитание есть сложение с числом, противоположным вычитаемому, а противоположные числа получать легко.

Конечно, в троичной уравновешенной системе рассматриваются и систематические дроби. Связь троичной уравновешенной системы

с другими системами осуществляется с помощью вышерассмотренных алгоритмов II и III.

Как и при изучении других систем счисления, важной является проблема технической реализации конкретной арифметики. Речь идет о возможности создания технических средств, в которых реализуются все подмеченные достоинства этой системы.

Еще раз обратим внимание на то, что при записи отрицательных чисел не выписывается знак «—», при выполнении операций не используются правила знаков и легко получать противоположные числа.

Эти особенности системы привлекают внимание конструкторов вычислительных машин. В Советском Союзе в 1958 г. была построена экспериментальная модель машины, арифметика которой базировалась на троичной уравновешенной системе счисления. Инициатором разработки этой уникальной машины была группа математиков МГУ во главе с академиком С. Л. Соболевым при участии Н. П. Брусенцова и С. П. Маслова. В 1962—1965 гг. было выпущено более 50 экземпляров промышленных образцов ЭВМ «Сетунь» — так по имени подмосковной речки была названа эта машина.

Особенности этой машины до сих пор привлекают внимание, инженерные поиски в создании троичных машин продолжаются. Главный конструктор ЭВМ «Сетунь» Н. П. Брусенцов указывает, что в этой машине реализованы далеко не все полезные свойства троичного кода и трехзначной логики.

Заметим, что можно рассматривать уравновешенные системы счисления с основаниями $q=5$, $q=7$ или $q=9$ и т. д. Вероятно, первой работой по этой теме явилась статья, опубликованная Дж. Лесли еще в 1917 г. Независимо от него в 1840 г. аналогичную работу издал О. Коши.

В этих работах рассматривались системы счисления с симметричным основанием, и не только троичные, а пятеричные и семиричные.

Одной из первых современных работ в этой области была работа К. Шеннона, опубликованная в 1950 г. Наиболее глубокие исследования проведены в 60-х гг. в МГУ.

8. Понятие о смешанных системах счисления. В рассмотренных выше системах счисления «вес» единицы любого разряда, кроме первого, всегда равнялся «весу» единицы предшествующего разряда, умноженному на основание системы. Можно, однако, представить вариант системы счисления, в которой понятие «основание системы счисления» сильно отличается от традиционного.

Существо нового подхода легко представить, если рассмотреть счеты необычной конструкции (рис. 21).

На нижней проволочке расположены две косточки, «вес» каждой равен единице. На второй проволочке косточек уже три, на третьей проволочке косточек четыре и т. д. На n-й проволочке $(n+1)$ косточек.

Так как каждая из косточек второй проволочки заменяет две косточки, расположенные на первой проволочке, то ее «вес» равен 2; каждая же косточка третьей проволочки заменяет три косточки весом в 2 и поэтому сама «весит» 6 единиц. Продолжая такие рассуждения, мы получим, что косточка, расположенная на n-й проволочке, имеет «вес», равный $n!$. «Вес» единиц от разряда к разряду растет, но растет неравномерно. Это приводит к неожиданному представлению чисел в многочленной форме:

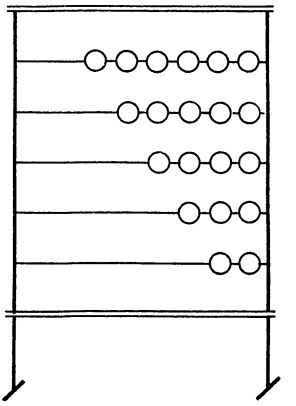


Рис. 21

$$\overline{(a_n a_{n-1} \ldots a_2 a_1)_\text{ф}} =$$
$$= a_n \cdot n! + a_{n-1}(n-1)! + \ldots + a_2 \cdot 2! + a_n \cdot 1!,$$

N — разрядное число, «списанное со счет»,— оказывается представленным не в виде суммы степеней основания q, а является суммой факториалов n первых натуральных чисел.

Пусть на счетах отложены числа $3221_\text{ф}$ и $40301_\text{ф}$. Их представления в форме многочлена таковы:

$3221_\text{ф} = 3 \cdot 4! + 2 \cdot 3! + 2 \cdot 2! + 1! = 89_{10}$;

$40301_\text{ф} = 4 \cdot 5! + 3 \cdot 3! + 1! = 499_{10}$.

Рассмотренная система есть пример смешанной системы счисления. Система называется факториальной.

Аналогично другим позиционным системам в факториальной рассматриваются систематические дроби и смешанные числа.

Пример:

$$301{,}102_\text{ф} = 3 \cdot 3! + 1 \cdot 1! + 1 \cdot \frac{1}{1!} + 2 \cdot \frac{1}{3!}.$$

Задать смешанную систему счисления это значит указать ее базис. Под базисом смешанной системы счисления понимается последовательность ключевых чисел, каждое из которых задает «вес» одного из разрядов.

Можно говорить и о базисе традиционных систем. Базис десятичной системы счисления:

..., 10^n, ..., 10^3, 10^2, 10, 1, 10^{-1}, 10^{-2}, ..., 10^{-n}, ...

Базис двоичной системы счисления:

..., 2^n, ..., 2^3, 2^2, 2, 1, 2^{-1}, 2^{-2}, 2^{-3}, ..., 2^{-n}, ...

Базис факториальной системы счисления:

..., $n!$, ..., 120, 24, 6, 2, 1, $\frac{1}{2}$, $\frac{1}{6}$, $\frac{1}{24}$, ..., $\frac{1}{n!}$, ...

Для примера обратим внимание на так называемую фибоначчиеву систему счисления. Ее базис составляют числа Фибоначчи:

..., 34, 21, 13, 8, 5, 3, 2, 1.

Пример:

$37_{10} = 34 + 3 = 10000100$.

Естественно, что возникает вопрос: имеют ли практическое значение какие-либо из смешанных систем? Ответ. Имеют.

Ниже будет приведен пример алгоритма решения линейных диофантовых уравнений и неравенств, при создании которого существенную роль сыграло представление линейного диофантового уравнения как числа, записанного в смешанной системе. И это лишь один из многих примеров.

Учитель должен знать, что поиск в мире систем счисления продолжается и в наше время. Ниже будет рассказано о практически очень важной системе — системе остаточных классов. Рекомендуем по литературе ознакомиться с нега-позиционными системами счисления. В этих позиционных системах основание целое отрицательное число. Обсуждаются системы с основаниями $q=-2$, $q=-4$ и даже $q=-10$. В поле зрения математиков — и системы с основанием, содержащим мнимую единицу. Неожиданные применения нашли системы с основаниями $q=2i$ и $q=1-i$.

Учитель должен понимать, что поиск в этой, казалось бы, очень консервативной области математики не завершен. Системой занимательных задач и упражнений учитель должен привлекать школьников к изучению особенностей систем счисления.

§ 19. ПРЕОБРАЗОВАНИЕ СИМВОЛЬНОЙ ИНФОРМАЦИИ

Научить ЭВМ работать со словом в обычном грамматическом смысле — важная задача. Огромна потребность в том, чтобы ЭВМ помогала в библиографическом деле, в ведении учета выпускаемой продукции, работала в информационно-справочных системах, где информация, заданная словами, совмещается с числовой информацией. Примером могут быть системы информационного обеспечения Олимпийских игр и иных состязаний, справочные транспортные системы.

В начале 60-х гг. появились первые языки программирования, которые обеспечивали человеку хорошие условия для решения на ЭВМ задач символьной обработки. Из отечественных языков назовем прежде всего язык Аналитик, разработанный Украинским институтом кибернетики под руководством академика В. М. Глушкова. Несколько позднее харьковские специалисты предложили язык Сириус. За рубежом в это время широкое распространение имел язык символьной обработки FORMAC.

В настоящее время средства символьной обработки не кажутся экзотическими и включаются в развитые версии языков, ориентированных на массового пользователя и даже на школьника.

Учитель должен знать и рассказывать своим ученикам, что с появлением такого языка, как Аналитик, появилась возможность сокращать алгебраические дроби, в числителе и знаменателе которых многочлены, приводить подобные члены, решать некоторые классы уравнений в общем виде, осуществлять дифференцирова-

ние. На математиков 60-х гг. такие способности ЭВМ производили большое впечатление.

Трудно сказать, сколько времени будет иметь учитель, чтобы научить школьников технике применения средств символьной обработки. Сам же он непременно должен овладеть методикой работы с такими средствами.

Ниже сведения об основных средствах и приемах символьной обработки иллюстрируются конкретными средствами языка Бейсик в версии MSX.

Еще раз следует подчеркнуть, что ЭВМ работает только с кодами — с кодами чисел и с кодами символов. Кодирование символов из любой предметной области осуществляется в единой системе кодирования, общепринятой является система ASCII.

Приступая к изучению средств символьной обработки и методик решения задач в этой области, следует понимать, что речь будет идти о своеобразной алгебре, в которой объектами являются символы и слова. В этой алгебре используются переменные, значениями которых являются уже не числа, а символы и строки символов. Над символьными данными определяются основные операции и вводится ряд функций.

Такой общематематический взгляд на все проблемы средств символьной обработки помогает в усвоении деталей.

1. Представление символьной информации. Для обеспечения взаимодействия между человеком и ЭВМ требуется более сотни символов. В набор требуемых символов входят буквы русского и латинского алфавита, арабские цифры, знаки арифметических действий, разделительные знаки (точка, запятая, точка с запятой, двоеточие), скобки, дефис, кавычки и многие другие. В отдельных странах требуется введение в этот набор иероглифов или иных специфических знаков.

Наиболее распространенной международной согласованной системой кодирования всех символов является система ASCII. Набор ASCII состоит из 128 символов, каждый символ кодируется семибуквенным двоичным словом ($128=2^7$).

Все кодируемые символы делятся на две группы: символы для записи информации и символы для взаимодействия с ЭВМ. Первые символы принято называть *печатаемыми*, вторые — *управляющими*. Печатаемые символы имеют коды от 0100000 до 1111110 (от 32 до 126).

Пользователь ЭВМ может по известному коду вызвать печатаемый символ на экран или на бумагу. Для этого используется функция SHR$ (I): если I=65, то CHR$ (65)=A; если I=90, то CHR$ (90)=Z.

Для решения обратной задачи — задачи определения кода для задаваемого печатаемого символа — используется функция ASC ("X"):

ASC ("C")=67; ASC ("K")=75.

Значением функции CHR$ (7) является не печатаемый символ, а выдача машиной звукового сигнала.

Как и при обработке числовой информации, при работе с символами используются постоянные и переменные величины.

Задать конкретную символьную величину можно в виде строки из символов, взятой в кавычки:

"А", "Алфавит", "11АВ", "MCMLXXXVIII"

Символьная константа (строка, взятая в кавычки) может в Бейсике содержать не более 255 символов. Рассматривается и пустая константа: " ". Символьные константы можно перечислять наряду с числовыми в операторе DATA.

Пример: 5DATA 5, —123, "год", "месяц"

Имена символьных переменных образуются из символов путем присоединения в конце символа $.

Символьная переменная может быть как простой переменной, так и переменной с индексом: A$, B$ (5).

Значения символьным переменным присваиваются с помощью оператора присваивания.

Пример:
```
10 LET A$="буква"
15 LET B$="1011"
20 LET C$="0"
```

Для обмена значениями символьных переменных можно использовать оператор SWAP. Из DATA значения символьных переменных берутся с помощью оператора READ. Значения символьным переменным задаются с клавиатуры с помощью оператора INPUT.

Сказанное разъясняет мысль о том, что использование символьных величин во многом схоже с применением числовых величин.

Заметим, что для букв латинского алфавита в Бейсике принят лексикографический порядок:

$$\text{ASC}(\text{"A"}) < \text{ASC}(\text{"B"}) < \text{ASC}(\text{"C"}) < \ldots < \text{ASC}(\text{"Z"})$$

Сравнивать можно не только отдельные символы, а и строки. Строка A длины L предшествует строке B длины L, если при посимвольном сравнении строк слева направо условие $a_i < b_i$ встретится раньше, чем условие $a_i > b_i$.

2. Основные функции от аргумента-символа. Наиболее распространен набор следующих функций:

```
LEN (X)
INSTR (X)
LEFT $ (X)
RIGHT $ (X)
MID $ (X)
```

Функция LEN (X) в качестве значения принимает число, равное количеству букв (символов) в строке X.

Пример:

```
5 LETN=LEN ("озеро")
10 PRINT "N="; N:END
```

На экран будет выведено N=5.

С помощью функции INSTR организуется проверка вхождения строки Y в строку X. Значением функции INSTR является номер символа проверяемой строки X, начиная с которого в нее входит строка Y.

Если Y не входит в X или N>LEN (X), то INSTR=0, где N — номер символа в строке X, начиная с которого осуществляется поиск вхождения строки Y в строку X. Если N опущено, т. е. функция записана в формате INSTR (X, Y), то N по умолчанию считается равным 1.

Пример:

```
5 LET A$="BASIC"
10 LET K=INSTR (1, A$, "SIC")
15 PRINT"K="; K
20 END
```

На экран будет выведено K=3.

Выделение в заданной строке отдельных частей (подстрок) выполняется в Бейсике с помощью следующих символьных функций:

LEFT$	(left — левая часть)
RIGHT$	(right — правая часть)
MID$	(middle — средняя часть)

Значением функции LEFT$ является строка (подслово исходного слова), состоящая из N первых слева символов данной строки.

Формат функции

```
LEFT$ (Y, N), Z
```

где Y — данная строка; N — число (или арифметическое выражение), равное количеству символов, выделяемых из данной строки.

Пример:

```
10 LET A$="BASIC"
15 PRINT LEFT$ (A$, 3)
20 END
```

На экран будет выведено слово BAS.

Значением функции RIGHT$ является строка из N символов исходного слова, взятых справа:

```
RIGHT$ (X, N)
```

где X — строка символов; N — арифметическое выражение или число, указывающее количество символов, отсчитываемых от конца данного слова (X).

Пример:

```
5  LET A$="BASIC"
10 PRINT RIGHT$ (A$, 3)
15 END
```

На экран будет выведено слово SIC.

Для выделения части слова, расположенного внутри данного слова X, используется функция MID$. Формат функции:

MID$ (X, I, N)

где X — данное слово (строка символов); N — число выделяемых символов в строке; I — номер символа, с которого, считая слева направо, начинается выделение подстроки.

Пример:
```
5  LET A$="BASIC"
10 PRINT MID$ (A$, 2,2)
15 END
```

На экран будет выведено слово AS.

Если N опущено, то результатом применения MID$ (A$, I) будет вся строка символов — от I до последнего символа A$.

Широко применяется в Бейсике функция VAL, имеющая формат VAL (X).

Значение функции, примененной к строковой переменной X, будет равно нулю, если начальный (слева направо) отрезок строки X не является изображением числа. Если же начальный отрезок строки является изображением числа, то это число становится значением функции.

Пример:
```
5  LET A$="251 (A+B)"
10 LET K=VAL (A$)
15 PRINT K:END
```

На экран будет выведено число 251.

Часто алгоритм требует замены конкретного числа на изображение его двоичного кода. Например, вместо числа 15 хотелось бы иметь строку="1111".

Это можно осуществить с помощью функции BIN$ (X), где X — целое выражение.

Пример:
```
5  LET A$=BIN$ (15)
10 PRINT A$; :PRINT BIN$(15)
15 END
```

На экран будут выведены две строки:
1111 1111

Аналогично используются и функции OCT$ и HEX$, с помощью которых можно получить в виде строк восьмеричные и шестнадцатеричные коды целых чисел.

Пример:
```
5  LET A$=BIN$ (15)
10 LET B$=OCT$ (15)
15 LET C$=HEX$ (15)
20 PRINT A$; B$; C$
25 END
```

На экран будут выведена строка:

1111 17 F

Учитель должен знать, что среди функций языка Бейсик есть функция STR$, с помощью которой арифметический двоичный код числа преобразуется в строку.

Формат функции:

STR$ (арифметическое выражение).

Пример: 5 LETR=VAL (RIGHT$(STR$(N)), 3)

Разъяснение примера: пусть N=12369, тогда выполнение операции STR$(12369) даст строку "12369". К этой строке можно применить функцию RIGHT$ и строку, полученную после этого: "369" с помощью функции VAL(369) преобразуем в число 369, а это есть остаток от деления числа 12369 на 100.

3. Операции над строками символов. Выражение A$+B$ в Бейсике есть указание выполнить операцию соединения двух строк в одну. Операндами в операции соединения (конкатенации) могут быть символьные величины любого вида: символьные константы, символьные переменные, символьные функции и выражения, содержащие операцию конкатенации.

Пример:
```
5  LET A$="MSX"
10 LET B$="BASIC"
15 LET C$=B$+A$
20 PRINT C$:END
```

На экран будет выведена фраза BASIC MSX.

Другим не менее сильным средством является операция замены в данной строке одной подстроки на другую равной длины.

Для этого используется функция MID$, но способ ее применения отличен от рассмотренного выше.

Формат этой операции следующий:

MID$ (Y, I, K)

где Y — строка символов, в которой часть строки будет заменена на строку символов X; I, K — числа или арифметические выражения; при этом I — номер символа, с которого в данной строке (Y) будет вестись замена; K — число символов в заменяющей строке (X): если K опущено, т. е. применяется формат MID$ (Y, I), то по умолчанию K=LEN (X).

Пример:
```
5  LET A$="ЭЛЕКТРОСТАНЦИЯ"
10 MID$ (A$, 8,7) =''ФИКАЦИЯ''
15 PRINT A$:END
```

На экран будет выведено слово ЭЛЕКТРОФИКАЦИЯ.

4. Простейшие задачи символьной обработки. Приводимые ниже задачи позволят учителю образно и основательно познакомить учащихся с задачами, требующими применения рассмотренных средств символьной обработки.

1. Напечатать все подслова данного слова A$, начинающиеся с первой буквы, по возрастанию их длин.

2. Напечатать все подслова данного слова A$, заканчивающиеся последней буквой, по убыванию их длин.

3. Напечатать все подслова данного слова A$, начинающиеся с первой буквы, по убыванию их длин.

4. Дан массив слов A$(N). Найти в нем все слова, начинающиеся с заданной приставки X$.

5. Дан массив слов A$ (N). Найти в нем все слова, имеющие данное окончание X$.

6. Дан массив слов A$ (N). Упорядочить массив строк (слов) по возрастанию их длин.

7. Дан массив слов A$ (N). Составить программу нахождения минимального и максимального слов в массиве.

8. Дан массив слов A$ (N). Написать все слова заданной длины.

9. Найти количество слов заданных длин L1 и L2 в массиве A$ (N).

10. Составить программу, которая данное слово запишет в обратном порядке.

11. Составить программу, проверяющую, является ли заданное слово палиндромом (т. е. словом, одинаково читающимся слева направо и справа налево).

12. Составить программу, упорядочивающую символы строки по алфавиту. Например, данная строка "BASIC" — результат строки "ABCIS".

13. Составить программу проверки, является ли данная строка повторением другой строки n раз.

14. Составить программу подсчета числа вхождений каждой буквы данного слова.

15. Составить программу преобразования десятичных натуральных чисел в римскую систему нумерации.

16. Составить программу преобразования чисел римской нумерации в десятичные числа.

17. Пусть дан многочлен с одной переменной x, расположенной по убывающим степеням x. Коэффициенты многочлена — целые числа. Многочлен введен в память ЭВМ в виде значения символьной переменной.

Пример: 5*X^5−3.2*X^4+71*X^2−12.7*X+4.9

Осуществить простейший анализ многочлена; установить степень многочлена, чему равен каждый коэффициент, и образовать массив коэффициентов, внеся в него и нулевые коэффициенты.

18. Написать данный многочлен, расположенный по убывающим степеням переменной x, в виде многочлена, раположенного по возрастающим степеням.

19. Дан массив O$ (N), образованный из N одночленов. Найти в нем подобные члены и привести их.

20. Дано несколько одночленов вида AX^n, где A — действительное число. Составить из данных одночленов многочлен, расположенный по убывающим степеням x.

Глава IV. МАШИННАЯ АРИФМЕТИКА

§ 20. ОБЩИЕ МЕТОДИЧЕСКИЕ ПОДХОДЫ

Вопросы реализации арифметики в ЭВМ требуют хорошего и тщательного разъяснения. Общая схема работы ЭВМ в сознании учащихся должна быть четкой. Здесь еще раз уместно подчеркнуть, что часть школьников может увлечься программированием, и этим учащимся особенности машинной арифметики помогут разобраться в тайнах точных вычислений на ЭВМ. Часть учащихся может заинтересоваться самими арифметическими устройствами: их больше привлекает конструкция сумматора, сдвигового регистра, преобразователя кода и других устройств.

До сих пор, с сожалению, существует тенденция эту часть знаний сообщать упрощенно. Нередко можно услышать, что ЭВМ работает в двоичной системе счисления и это все, что запоминает школьник.

Рассмотрим вариант разработки темы. Желательно еще раз уточнить представления о нормализации чисел (в том числе и нормализации двоичных чисел), об арифметике нормализованных чисел, о двоичных кодах и арифметике двоичных кодов, отдельно рассмотреть вопросы арифметики многоразрядных чисел, как целых, так и рациональных. В качестве дополнительной информации предлагается рассказ о системе счисления остаточных классов. Система рассматривается как пример и как результат целеустремленных поисков в области математики и вычислительной техники.

Что из предложенного ниже учитель выберет для сообщения на уроке или на занятиях кружка, решать самому учителю.

§ 21. НОРМАЛИЗАЦИЯ ЧИСЕЛ. АРИФМЕТИКА НОРМАЛИЗОВАННЫХ ЧИСЕЛ

Чтобы познакомить школьников с простейшими вычислительными средствами, учителю нужно ввести понятие «нормализованное число». Запись чисел в нормализованной форме будет использоваться и позднее, во второй части курса при изучении ЭВМ. Сведения, изложенные ниже, о нормализованных числах адресованы прежде всего учителю.

Рассмотрение понятия «нормализованное число» начнем с введения понятия «десятичное нормализованное число».

Определение. *Число A_{10} называется нормализованным, если оно представлено в виде*

$$A_{10} = \pm M_{10} \cdot 10^{\pm p_{10}} \quad (27{,}32_{10} = 0{,}2732 \cdot 10^2),$$

где M_{10} — мантисса, десятичная правильная дробь, равная или больше 0,1 (т. е. $0{,}1 \leqslant M_{10} < 1$); p_{10} — порядок, целое десятичное число.

Примеры:

$297_{10} = 0{,}297 \cdot 10^3$ ($M_{10} = 0{,}297$; $p_{10} = 3$);
$0{,}031_{10} = 0{,}31 \cdot 10^{-1}$ ($M_{10} = 0{,}31$; $p_{10} = -1$);
$-34{,}176_{10} = -0{,}34176 \cdot 10^2$ ($M_{10} = -0{,}34176$; $p_{10} = 2$).

При нормализации числа происходит своеобразное расчленение его на составляющие:

знак числа — знак порядка;

модуль мантиссы — модуль порядка.

Зная каждое из этих составляющих, можно восстановить исходное число, например:

знак числа — "—"; знак порядка — "+"
модуль мантиссы — $M_{10} = 0{,}6983$; модуль порядка — $p_{10} = 2$.

Искомое число: $-0{,}6983 \cdot 10^2 = -69{,}83_{10}$.

В нормализованной форме могут быть представлены натуральные, целые, рациональные и иррациональные числа. Нормализованные числа образуют множество чисел, в которых действует своя арифметика.

Арифметика нормализованных чисел тесно связана с ее реализацией в вычислительных устройствах. Связь состоит в том, что в любом вычислительном устройстве, использующем арифметику нормализованных чисел, в памяти для хранения результатов и в сумматоре отводится раз навсегда в установленное для этого класса устройств число разрядов как для порядка (p_{10}), так и для мантиссы (M_{10}); определен способ хранения знаков мантиссы и порядка.

В качестве примера опишем, как организуется размещение нормализованного числа в ячейке микрокалькулятора МКШ-2 (рис. 22). На рисунке показано, что под мантиссу отводится пять, а под порядок два десятичных разряда. Иногда (это, кстати, реа-

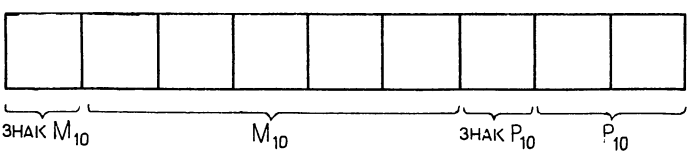


Рис. 22

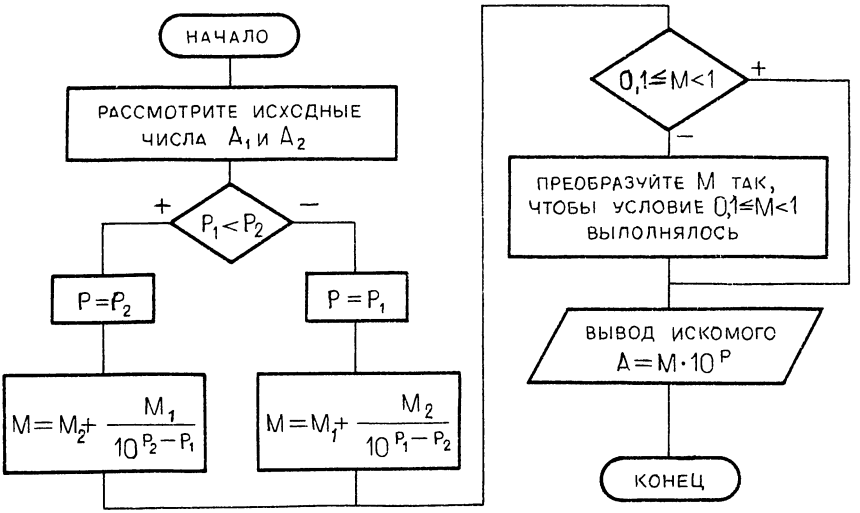


Рис. 23

лизовано в МКШ-2) на мантиссу накладываются иные ограничения: $1{,}0 \leqslant M_{10} < 10$ и в этом случае называют не нормализованными, а записанными в нормальной форме. Примеры чисел, записанных в нормальной форме: $-2{,}316 \cdot 10^{-2}$; $-8{,}37 \cdot 10^{3}$.

Заметим, что сложение и вычитание нормализованных чисел — это арифметика, в которой операции выполняются над новыми объектами; они обладают специфическими свойствами и поэтому будут обозначаться по-новому: $\oplus$ и $\ominus$.

Сложение и вычитание двух нормализованных чисел: $A_1 = M_1 \cdot 10^{p_1}$ и $A_2 = M_2 \cdot 10^{p_2}$ — выполняется в соответствии с алгоритмом, изображенным на рисунке 23. Подчеркнем, что при выполнении сложения и вычитания могут возникать искажения исходных чисел. Одним из источников может быть сдвиг вправо цифр мантиссы при выравнивании порядка (это необходимо делать до начала операции). Часть цифр мантиссы может оказаться при этом за пределами отведенной разрядной сетки и потеряться. Может переполниться и та часть разрядной сетки, которая отведена для цифр порядка.

Рассмотрим примеры. В каждом примере полагаем, что для цифр мантиссы отведено пять разрядов.

1. Даны слагаемые: $A_1 = 0{,}87854 \cdot 10^{1}$ и $A_2 = 0{,}36101 \cdot 10^{4}$. Найти $A_1 \oplus A_2$.

Найти сумму $A_1 \oplus A_2$ — это значит найти порядок искомого числа и его мантиссу. Алгоритм предусматривает отдельную работу с мантиссами и порядком исходных слагаемых:

$$M_1 = 0{,}87854, \qquad p_1 = 1;$$
$$M_2 = 0{,}36101, \qquad p_2 = 4.$$

Так как $p_1 < p_2$ $(1 < 4)$, то в соответствии с алгоритмом имеем

$$M = M_2 + \frac{M_1}{10^{p_2 - p_1}} = 0{,}36101 + \frac{0{,}87854}{10^3} = 0{,}36101 + 0{,}00087 = 0{,}36188;$$

$$p = p_2 = 4.$$

Следовательно $A_1 \oplus A_2 = M \cdot 10^p = 0{,}36188 \cdot 10^4$. Мы видим, что произошло переполнение разрядной сетки, часть цифр мантиссы M_1 пропала — мантисса M_1 еще до сложения с мантиссой оказалась измененной: она уменьшилась.

2. Даны слагаемые: $A_1 = 0{,}96098 \cdot 10^1$ и $A_2 = 0{,}98009 \cdot 10^2$. Найти $A_1 \oplus A_2$.

Выравниваем порядки, увеличив порядок p_1 на единицу:

$$\begin{array}{r} A_1 = 0{,}09609 \cdot 10^2 \\ + \quad A_2 = 0{,}98009 \cdot 10^2 \\ \hline 1{,}07618 \cdot 10^2 \end{array}$$

Сумма требует нормализации:

$A_1 \oplus A_2 = 1{,}07618 \cdot 10^2 = 0{,}10761 \cdot 10^3$.

В этом примере дважды терялись цифры мантиссы.

Из уже приведенных примеров можно сделать вывод, что ограничения, вносимые размерами разрядной сетки, могут приводить к заметным погрешностям. По этой же причине арифметика нормализованных чисел обладает «неожиданностями». Примером одной из таких «неожиданностей» является то, что суммирование некоторых нормализованных чисел не обладает сочетательным свойством, т. е. $(A_1 \oplus A_2) \oplus A_3 \neq A_1 \oplus (A_2 \oplus A_3)$, для некоторых чисел, например:

$A_1 = 0{,}93659 \cdot 10^0$; $A_2 = 0{,}96877 \cdot 10^4$; $A_3 = 0{,}56301 \cdot 10^3$.

Найдем сумму $(A_1 \oplus A_2) \oplus A_3$:

$$\begin{array}{r} A_1 \oplus A_2 \\ 0{,}00009 \cdot 10^4 \\ + \quad 0{,}96877 \cdot 10^4 \\ \hline 0{,}96886 \cdot 10^4 \end{array} \qquad \begin{array}{r} (A_1 \oplus A_2) \oplus A_3 \\ 0{,}96886 \cdot 10^4 \\ + \quad 0{,}05630 \cdot 10^4 \\ \hline 1{,}02516 \cdot 10^4 = 0{,}10251 \cdot 10^5 \end{array}$$

Теперь найдем сумму $A_1 \oplus (A_2 \oplus A_3)$:

$$\begin{array}{r} A_2 \oplus A_3 \\ 0{,}96877 \cdot 10^4 \\ + \quad 0{,}05630 \cdot 10^4 \\ \hline 1{,}02507 \cdot 10^4 = 0{,}10250 \cdot 10^5 \end{array} \qquad \begin{array}{r} A_1 \oplus (A_2 \oplus A_3) \\ 0{,}10250 \cdot 10^5 \\ + \quad 0{,}00000 \cdot 10^5 \\ \hline 0{,}10250 \cdot 10^5 \end{array}$$

Ответы не совпали — для подобранных чисел ассоциативный закон сложения оказался неверным.

Умножение и деление нормализованных чисел осуществляется по более простым правилам. Найти произведение нормализованных чисел $A_1 \otimes A_2$ — это значит найти мантиссу и порядок произведения.

Если $A_1 = M_1 \cdot 10^{p_1}$ и $A_2 = M_2 \cdot 10^{p_2}$, то искомые порядок и мантисса находятся так: $p = p_1 + p_2$; $M = M_1 \cdot M_2$.

Полученное число M может потребовать нормализации. Деление числа $A_2 = M_2 \cdot 10^{p_2}$ на число $A_1 = M_1 \cdot 10^{p_1}$ осуществляется в соответствии с правилами: $p = p_2 - p_1$; $M = M_2 / M_1$.

Заметим, что и при выполнении умножения и деления также могут иметь место «неожиданности»:

умножение некоторых нормализованных чисел не обладает сочетательным свойством;

умножение некоторых нормализованных чисел не обладает распределительным свойством относительно суммы некоторых нормализованных чисел;

существуют такие $A_1 \neq A_2$, что $A_1 \oplus A_1 = A_2 \oplus A_2$.

Может возникнуть вопрос: зачем же использовать арифметику, в которой столько отклонений от традиционных законов?

Ответ один: нормализация чисел позволяет создать единую арифметику; для вычислительных устройств нет проблемы распознавания типа числа, так как арифметическое устройство работает только с мантиссами и порядками.

§ 22. СПЕЦИАЛЬНЫЕ ДВОИЧНЫЕ КОДЫ И ИХ АРИФМЕТИКА

В вычислительных машинах арифметические устройства работают с нормализованными числами, но не с десятичными, а двоичными.

Примеры: $110{,}1101_2 = 0{,}1101101_2 \cdot 10_2^{11_2}$ ($M_2 = 0{,}1101101_2$; $p_2 = 11_2$);

$-0{,}0101_2 = -0{,}101_2 \cdot 10_2^{-12}$ ($M_2 = -0{,}101_2$; $p_2 = -1_2$).

Всякое десятичное число, прежде чем оно попадает в память ЭВМ, преобразуется по схеме: $A_{10} \rightarrow A_8 \rightarrow A_2 \rightarrow A_2 = M_2 \cdot 10^{p_2}$. Подчеркнем важное обстоятельство — на этом преобразование исходного числа не завершается: в полученном нормализованном двоичном числе мантисса с ее знаком заменяется кодом мантиссы, а порядок с его знаком заменяется кодом порядка. В дальнейшем ЭВМ имеет дело только с кодами порядка и мантиссы.

Используются следующие коды двоичных чисел:

прямой код, обратный код, дополнительный код.

Во всех этих кодах вместо знаков «+» или «—» используются символы 0 и 1, записываемые в так называемом знаковом разряде. Знаковый разряд расположен слева перед старшим разрядом числа.

Прямой код двоичного числа (а это либо мантисса, либо порядок) образуется по такому алгоритму:

1. Рассмотреть данное двоичные число — оно либо целое (порядок), либо правильная систематическая двоичная дробь (мантисса).

2. Если рассматриваемое число является дробью, то отбросить

нуль целых и зачеркнуть запятую; полученную запись рассмотреть как целое двоичное число и перейти к пункту 3.

3. Если рассматриваемое число целое, то обратить внимание на его знак, и если число положительное, то в знаковый разряд образуемого прямого кода вписать 0, в противном случае 1.

4. Все символы полученной записи считать цифрами искомого прямого кода.

Пример 1. Даны два числа: $A=-0{,}101101_2$ и $B=0{,}1101101_2$. Прямыми кодами соответственно будут:

$$A_{\text{пр}}=\boxed{1}\ 101101 \text{ и } B_{\text{пр}}=\boxed{0}\ 1101101.$$

Обычно цифры знаковых разрядов в рамки не берут. Обратный и дополнительный коды отличаются от прямого способом представления отрицательных чисел и совпадают с прямым, если данные числа положительные.

Обратный код отрицательного числа образуется из прямого по следующему алгоритму:

1. Во всех разрядах, кроме знакового, нули заменить на единицы, а единицы — на нули.

2. Полученная запись является обратным кодом данного отрицательного числа.

Пример 2. Даны два числа: $A=-0{,}10101_2$ и $B=0{,}1101_2$. Для каждого из них образуем сначала прямой код, а затем обратный: $A_{\text{пр}}=110101$, $A_{\text{обр}}=101010$; $B_{\text{пр}}=01101$, $B_{\text{обр}}=01101$.

Дополнительный код отрицательного двоичного числа образуется путем увеличения на единицу (по правилам сложения двоичных чисел) обратного кода этого числа.

Пример 3. Даны два числа: $A=-0{,}10010_2$ и $B=0{,}1011_2$. Для каждого из них образуем сначала прямой код, затем обратный и дополнительный: $A_{\text{пр}}=110010$, $A_{\text{обр}}=101101$, $A_{\text{доп}}=101110$; $B_{\text{пр}}=B_{\text{обр}}=B_{\text{доп}}=01011$.

Обратный и дополнительный коды двоичных чисел позволяют заменить операцию вычитания чисел сложением их кодов.

Преобразование чисел в требуемый код автоматически осуществляется при вводе чисел в ЭВМ. Полученные коды порядка и мантиссы для каждого числа перемещаются в ячейки памяти ЭВМ. Для каждой цифры кода в ячейке памяти отводится один двоичный разряд, или бит. В процессе выполнения вычислений коды мантисс и порядка конкретных чисел извлекаются из ячеек памяти и направляются в арифметическое устройство ЭВМ — АУ. В АУ над кодами по специальным правилам арифметики кодов выполняются действия. Подчеркнем, что в АУ действия выполняются не над двоичными числами по правилам двоичной арифметики, а над кодами двоичных чисел по правилам арифметики кодов. Это очень важное различие.

В нашу задачу не входит изучение основ арифметики двоичных кодов. Важно помнить, что в АУ над кодами порядка и ман-

тиссы выполняются действия, в итоге которых получают коды порядка и мантиссы результата. По этим данным восстанавливается результат в виде двоичного числа со знаком, а затем оно преобразуется в восьмеричное и в конечном счете в искомое десятичное.

§ 23. АРИФМЕТИКА МНОГОРАЗРЯДНЫХ ЦЕЛЫХ ЧИСЕЛ

Как организуется на ЭВМ работа с числами, содержащими в своей записи много цифр? Как получить результат, если заранее известно, что количество цифр велико? Иначе говоря, можно ли вычислить на ЭВМ типа «Корвет» знаменитое шахматное число $2^{64}-1$, если известно, что это число 18446744073709551615 не поместится в одну ячейку памяти? Каким образом можно вычислить число π с более чем ста цифрами после запятой?

Желательно, чтобы учитель нашел время разъяснить учащимся приемы, применяемые программистами при работе с многоразрядными целыми числами.

Остановимся на одном из возможных подходов в организации хранения многоразрядных чисел в памяти ЭВМ. Пусть необходимо запомнить целое число, содержащее в записи 65 цифр. Это можно сделать, отведя в памяти ЭВМ под каждую из 65 цифр одну ячейку. При этом цифры будут образовывать массив из 65 элементов. Таким образом, можно запомнить 255-разрядное число — не более. Если исходное число содержит более 255 цифр, то придется формировать уже не один, а несколько массивов. Очевидно, что это не лучший подход: ведь элементом числового массива может быть не цифра, а многоразрядное число.

Можно поступить так: данное 65-разрядное число заменить равным ему числом в системе счисления с большим основанием, например с основанием $q=1000\,000$. Это сделать очень просто: достаточно в данном числе отделить шестерки цифр, продвигаясь от младших разрядов к старшим. Например, исходное число $12356780165943156447765430 2_{10}$ содержит 27 цифр. Запишем его в виде

$$123\cdot(10^6)^4+567801\cdot(10^6)^3+659431\cdot(10^6)^2+564477\cdot(10^6)^1+ \\ +654302\cdot(10^6)^0.$$

Это число записано в системе с основанием $q=10^6$, его цифры: 123, 567801, 659431, 564477, 654302. Каждую из этих цифр можно считать элементом массива, в этом случае потребуется всего пять ячеек. Аналогично можно разместить в памяти ЭВМ какое-нибудь другое многоразрядное целое число (например, B_{10}).

Из сказанного вытекает, что арифметика многоразрядных чисел сводится к действиям над «необычными» цифрами — в роли цифр выступают многоразрядные десятичные числа. Заметим, что подбор нового основания системы (в приведенном выше примере

это было основание $q=10^6$) должен учитывать, какие действия над числами будут выполняться. Если предстоит суммирование чисел, то новое основание может равняться и ста миллионам; если же предстоит умножение двух многоразрядных чисел, то новое основание системы должно быть таким, чтобы произведение двух цифр могло поместиться в одной ячейке: ведь произведение двух цифр (речь идет о цифрах системы с новым основанием) есть либо цифра, либо двухразрядное число в новой системе счисления.

Учитель должен проработать со школьниками алгоритмы суммирования и умножения цифр в системе с любым основанием, когда каждая цифра есть элемент массива.

Полезно рассмотреть суммирование и умножение чисел, когда основание системы необязательно является степенью десяти.

В качестве задач рекомендуются такие:

1. Составить программу вычисления числа 100!.

2. Составить программу вычисления $99\cdot97\cdot95\cdot93\cdot\ldots\cdot5\cdot3\cdot1$.

3. Составить программу вычисления следующих чисел Мерсенна: $2^{19}-1$, $2^{31}-1$, $2^{61}-1$.

4. Составить программу вычисления 10-ти чисел Фибоначчи, начиная с сотого числа.

5. Составить программу вычисления чисел Каталана, используя формулу $K_n=K_{n-1}(4n-6)$ при $n=100$, $K_1=K_2=1$.

6. Ниже приводится текст программы вычисления числа $N!$ при достаточно большом N, например $N=57$. Найти 57!.

```
10 '------------ ВЫЧИСЛЕНИЕ ФАКТОРИАЛА ЧИСЛА N ----------
20 INPUT N
30 DIM A(N)
40 A(1)=1 : L=1
50 FOR K=1 TO N
60      I=1 : R1=0 : R2=0
70      IF R2=0 THEN IF I>L THEN 120
80      R=A(I)*K+R2
90      R2=INT(R/(10^6))
100     R1=R-R2*10^6
110     A(I)=R1 : I=I+1 : GOTO 70
120     L=I-1
130 NEXT K
140 PRINT "РЕЗУЛЬТАТ" N; : LOCATE CSRLIN, POS(0) - 1:
    PRINT "! = " ;
150 FOR J=1 TO I-1
160     SM$="000000" : KK$=STR$(A(I-J))
170     KK$=RIGHT$(KK$,LEN(KK$)-1)
180     MID$(SM$,7-LEN(KK$))=KK$
190     PRINT SM$; " ";
200 NEXT J
210 END
```

В операторах 70—110 осуществляется формирование цифр искомого результата A ($I-J$), каждая из этих цифр есть остаток отделения промежуточного произведения вида $(i)!\cdot(i+1)$ на 10^6. В связи с этим нужно иметь в виду, что данный алгоритм ограничен величиной N. Если N очень велико, например более 10^4, то промежуточное произведение R (оператор 80) может по числу цифр выйти за рамки разрядной сетки конкретной ЭВМ.

Если рассмотреть результат вычислений, то обнаружится, что результат выписан цифра за цифрой в системе с основанием $q=10^6$. Для каждой цифры при выводе ее на печать отведено по шесть позиций. Напомним, что цифры в системе с основанием $q=10^6$ удовлетворяют условию: $000000 \leqslant a_i \leqslant 999999$. Ни одна из позиций не может быть утеряна.

```
РЕЗУЛЬТАТ 100! =000093 326232 150424 994420 050856 392080 321696 982432
206828 083266 872052 452016 688036 607252 092048 608680
891288 078416 213228 141462 862428 225612 688064 000000 000000 000000
000000
```

Обеспечение грамотного вывода результата на печать в этой задаче и аналогичных задачах при работе с многоразрядными числами является самостоятельной программистской задачей.

В данной программе использованы средства символической обработки величин. Так, введена символьная переменная SM$, которая выполняет роль ячейки, в которую могут быть записаны все шесть (или меньше) символов любой из цифр a_i.

Запись A_i в ячейку SM$ осуществляется в несколько шагов. На первом шаге (строка программы 160) число A (I—J) преобразуется в строку символов: KK$.

На втором шаге (строка программы 170) строку символов делают короче на один символ (убирается знаковый разряд), и, наконец, на третьем шаге с помощью функции MID$ в ячейку SM$ помещается символьная переменная a_i (она займет в ячейке позиции с (7—LEN (KK$)). На печать будет выведено не число, а каждая цифра a_i в виде символьной переменной SM$.

Учитель, познакомивший учащихся с проблемами вычисления больших чисел, может предложить задачу, решение которой непосредственно опирается на приведенную программу. Речь идет о проблеме вычисления степени N натурального числа S: например, вычислить 2^{64}.

Предлагаются следующие соображения: вычисление числа $N!$ требует выполнения $(N-1)$ умножения, при этом каждый новый сомножитель на единицу больше предшествующего ($N!=1\cdot 2\times$ $\times 3\cdot\ldots\cdot N$). Если осуществить $(N-1)$ умножение, но при этом все сомножители взять одинаковыми, например S, то результат окажется равным S^N.

Ясно, что в программу вычисления факториала, приведенную выше, изменений будет внесено немного. В строке программы 80

следует вместо переменной K вписать постоянную S. Строка будет иметь вид R=A (I)·S+R2.

Естественно, что более просто будет организован и вывод результата S^N.

Текст программы приводится ниже.

```
10 '------------ ВЫЧИСЛЕНИЕ N-Й СТЕПЕНИ ЧИСЛА S --------
20 INPUT S, N
30 DIM A(N)
40 A(1)=1 : L=1
50 FOR K=1 TO N
60      I=1 : R1=0 : R2=0
70      IF R2=0 THEN IF I>L THEN 120
80      R=A (I)*S+R2
90      R2=INT(R/(10^6))
100     R1=R-R2*10^6
110     A(I)=R1 : I=I+1 : GOTO 70
120     L=I-1
130 NEXT K
140 PRINT "РЕЗУЛЬТАТ"S"^"N"= ";
150 FOR J=1 TO I-1
160     SM$="000000" : KK$=STR$(A(I-J))
170     KK$=RIGHT$(KK$,LEN(KK$)-1)
180     MID$(SM$,7-LEN(KK$))=KK$
190     PRINT SM$; " ";
200 NEXT J
210 END
```

Завершая обсуждение проблем арифметики многоразрядных целых чисел, обратимся к задаче вычисления биномиальных коэффициентов.

Обычно в литературе приводится расчетная формула в виде

$$C_n^k = \frac{n!}{(n-k)\cdot k!}. \qquad (*)$$

Создавая программу для вычислений C_n^k, школьники часто действуют прямолинейно, все вычисления планируют в строгом соответствии с формулой (*). Они сначала отдельно вычисляют значения $n!$, $(n-k)!$, $k!$, а затем работают с дробью $n!\,/((n-k)!\,k!)$, и здесь обнаруживаются недостатки такого подхода. Функция факториал растет очень быстро (уже при $n=50$ и $k=7$ формула дает отказ из-за переполнения разрядной сетки). Небольшие по величине коэффициенты вычислить не удается потому, что не продуман алгоритм таких вычислений.

Учитель может обратиться здесь к понятию «рекуррентная формула» и показать, что имеет место формула

$$C_n^k = \frac{n-k+1}{k}\cdot C_n^{k-1}, \qquad (**)$$

где $C_n^1 = n$.

Располагая такой формулой, можно организовать вычисление одного за другим чисел C_n^k, начиная с $C_n^1=n$ или $C_n^0=1$.

Так, для $n=5$ будем иметь

$$C_n^0=1;\ C_n^1=C_5^1=5;$$

$$C_n^2=C_5^2=\frac{5-2+1}{2}\cdot C_5^1=10;\ C_n^3=C_5^3=\frac{5-3+1}{3}\cdot C_5^2=10;$$

$$C_n^4=C_5^4=\frac{5-4+1}{4}\cdot C_5^3=5;\ C_n^5=C_5^5=\frac{5-5+1}{5}\cdot C_5^4=1.$$

Так как величины биномиальных коэффициентов изменяются медленно, то программа, составленная для вычислений по рекуррентной формуле (**), выдаст значительно большее количество искомых биномиальных коэффициентов.

Текст программы может быть таким:

```
10 , ВЫЧИСЛЕНИЕ C(N,K)
20 INPUT "H"=; N
30 INPUT "K"=; K
40 IF K>N-K THEN K=N-K
50 C=1
60 FOR I=1 TO K
70 C=((N-I+1)/I)*C
80 NEXT I
90 PRINT"C(N,K)="; C
100 END
```

Работая по программе, ЭВМ вычислила коэффициенты:

```
RUN
H=? 40
K=? 6
C (N, K) =3838380
OK
RUN
H=? 50
K=? 5
C (N, K) =2118760
OK
RUN
H=? 60
K=4
C (N, K) =487635
OK
RUN
H=? 60
K=5
C (N, K) =5461512
OK
```

В качестве дополнительных задач по развитию идей, заложенных в рассмотренных алгоритмах, рекомендуем следующие:

1. Внести изменения в программу вычисления факториала числа N, с тем чтобы ЭВМ могла вычислить произведение $1\cdot3\cdot5\cdot7\times\times\ldots\cdot(2N-1)$.

2. Внести изменения в программу вычисления факториала числа N, с тем чтобы ЭВМ могла вычислить произведение $1\cdot2\cdot4\cdot6\cdot8\cdot\ldots\cdot(N-2)$.

3. Внести изменения в программу вычисления N-й степени числа S, с тем чтобы ЭВМ могла возвести в степень рациональные числа.

4. Внести изменения в программу вычисления N-й степени числа S, с тем чтобы ЭВМ могла вычислить степени отрицательных целых чисел.

§ 24. АРИФМЕТИКА РАЦИОНАЛЬНЫХ ЧИСЕЛ НА ЭВМ

Часто у школьников возникает представление, что ЭВМ не может работать с рациональными числами, записанными в виде обыкновенных дробей (p/q). Это представление учитель должен изменить: есть немало задач, когда и компоненты вычислений, и результаты представляются в виде дробей, числитель и знаменатель которых есть целые числа.

Заметим, что арифметика обыкновенных дробей требует умения решать такие простые задачи, как:

вычисление НОД двух чисел (это необходимо для выполнения операции сокращения дробей);

вычисление НОК двух чисел (это необходимо при приведении дробей к общему знаменателю).

В целом арифметика обыкновенных дробей может рассматриваться как источник задач для самостоятельного решения учащимися. Можно рекомендовать такие задачи:

1. Составить программу сокращения дроби M/N.
2. Сравнить две обыкновенные дроби: p/q и m/n — по величине.
3. Дано несколько обыкновенных дробей. Найти среди них наибольшую и наименьшую.
4. Составить программу суммирования двух обыкновенных дробей.
5. Составить программу вычитания одной обыкновенной дроби из другой.
6. Составить программу умножения двух обыкновенных дробей.
7. Составить программу деления одной обыкновенной дроби на другую.
8. Составить программу возведения в степень обыкновенной дроби.
9. Составить программу разложения данного натурального числа на простые множители.
10. Составить программу отыскания всех делителей натурального числа — и простых, и составных.

Заметим, что к двум последним задачам близки по своему характеру задачи на отыскание совершенных чисел, простых чисел-близнецов, чисел дружественных.

Совершенным называется натуральное число, равное сумме всех его делителей. Такими числами являются:

$$6=1+2+3 \text{ и } 28=1+2+4+7+14.$$

В связи с этим учитель может предложить своим ученикам такие задачи:

1. Проверить, являются ли числа 496 и 8128 совершенными.
2. Составить программу формирования совершенных чисел.
3. Проверить на примерах теорему Эвклида: «В тех случаях, когда число $p=1+2+4+8+\ldots+2^n=2^{n+1}-1$ простое, то число $2^n \cdot p$ является совершенным».

Дружественными числами A и B называются тогда, когда сумма всех делителей числа A равна числу B и наоборот. Отсюда следуют задания:

4. Проверить являются ли числа 220 и 284 дружественными.
5. Проверить на примерах теорему Сабита: «Если все три числа: $p=3\cdot 2^{n-1}-1$, $q=3\cdot 2^n-1$ и $r=9\cdot 2^{2n-1}-1$ — числа простые, то числа $A=2^n\cdot p\cdot q$ и $B=2^n\cdot r$ есть числа дружественные». (Проверку выполнить для $n=2,4$ и 7.)

В заключение рассмотрим решение задачи о представлении обыкновенной дроби суммой египетских дробей. Идею алгоритма раскроем на примере, а затем сформулируем ее в общем виде.

Пусть имеется дробь 15/16. Попробуем представить ее в виде суммы первой из египетских дробей 1/2 и пока неизвестной дроби p/q. Имеем

$$\frac{15}{16}=\frac{1}{2}+\frac{p}{q}. \text{ Отсюда } \frac{p}{q}=\frac{15}{16}-\frac{1}{2}=\frac{15\cdot 2-16\cdot 1}{16\cdot 2}=\frac{7}{16}.$$

Далее дробь 7/16, которая не является египетской, представляем в виде суммы египетской дроби 1/3 и вновь пока неизвестной дроби p/q. Имеем

$$\frac{7}{16}=\frac{1}{3}+\frac{p}{q}, \text{ откуда } \frac{p}{q}=\frac{7}{16}-\frac{1}{3}=\frac{7\cdot 3-16\cdot 1}{16\cdot 3}=\frac{5}{48}.$$

К дроби 5/48 применим тот же прием и получаем

$$\frac{5}{48}=\frac{1}{10}+\frac{p}{q} \text{ и } \frac{p}{q}=\frac{5}{48}-\frac{1}{10}=\frac{5\cdot 10-48}{48\cdot 10}=\frac{1}{240}.$$

Последняя дробь оказалась египетской — это значит, что задача решена:

$$\frac{15}{16}=\frac{1}{2}+\frac{1}{3}+\frac{1}{10}+\frac{1}{240}.$$

Если представить в виде суммы египетских дробей дробь 3/7, то будем иметь

$$\frac{3}{7}=\frac{1}{3}+\frac{1}{11}+\frac{1}{231}.$$

Алгоритм в общем виде задается схемой, изображенной на рисунке 24.

Программа, составленная в соответствии с приведенной схемой, может быть такой, как приведенная ниже.

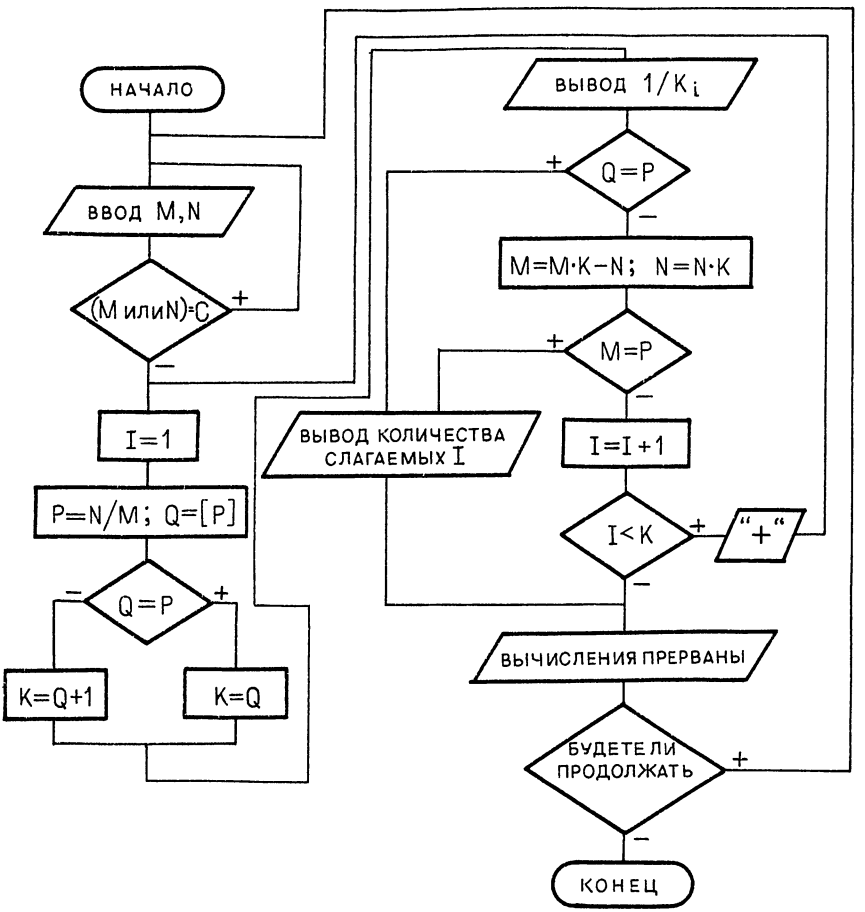


Рис. 24

```
10 PRINT "РАЗЛОЖЕНИЕ ДРОБИ В СУММУ ЕГИПЕТСКИХ ДРОБЕЙ"
20 PRINT
30 INPUT "ВВЕДИТЕ ЧИСЛИТЕЛЬ ДАННОЙ ДРОБИ M"; M
40 INPUT "ВВЕДИТЕ ЗНАМЕНАТЕЛЬ ДАННОЙ ДРОБИ N"; N
50 IF M=0 THEN PRINT "M=0 - НЕДОПУСТИМО": GOTO 30
60 IF N=0 THEN PRINT "N=0 - НЕДОПУСТИМО": GOTO 30
70 PRINT "ИСХОДНАЯ ДРОБЬ ===> "; M: "/"; N
80 PRINT "ИСКОМОЕ РАЗЛОЖЕНИЕ ===> ";
90 LET I=1
100 LET P=N/M
110 LET Q=INT(P)
120 IF Q=P THEN LET K=Q ELSE  LET K=Q+1
130 PRINT "1/"; K;
```

```
140 IF Q=P THEN 210
150 LET M=M*K-N
160 LET N=N*K
170 IF M=O THEN 210
180 LET I=I+1
190 IF I<15 THEN PRINT "+"; : GOTO 100
200 PRINT "ВЫЧИСЛЕНИЯ ПРЕРВАНЫ": GOTO 230
210 PRINT "В ИСКОМОЙ СУММЕ"; I; "СЛАГАЕМЫХ"
220 PRINT
230 PRINT "ПРОДОЛЖИТЬ РАБОТУ? ДА ИЛИ НЕТ?"
240 INPUT A$
250 IF A$="ДА" THEN 30
260 END
```

Ниже приводится протокол работы ЭВМ по решению конкретной задачи: дробь 63/64 представить в виде суммы египетских дробей.

```
 RUN
РАЗЛОЖ. ДРОБИ В СУММУ ЕГИПЕТСКИХ ДРОБЕЙ
ВВЕДИТЕ ЧИСЛИТЕЛЬ ДАННОЙ ДРОБИ? 63
ВВЕДИТЕ ЗНАМЕНАТЕЛЬ ДАННОЙ ДРОБИ? 64
ИСХОДНАЯ ДРОБЬ 63/64
ИСКОМОЕ РАЗЛОЖЕНИЕ 1/2+1/3+1/7+1/123+1/18368 В ИСКОМОЙ СУММЕ СЛАГАЕМЫХ
ПРОДОЛЖИТЬ РАБОТУ? ДА ИЛИ НЕТ?
?
OK
```

Конечно, у предложенного алгоритма есть ограничения. Если знаменатели получаемых дробей велики, то ЭВМ выводит их в нормализованной форме, и тем самым точное решение задачи заменяется приближенным.

Такое затруднение преодолимо. Учитель может воспользоваться алгоритмами вычислений с многоразрядными числами и получить точные решения обсуждаемых задач. Однако следует иметь в виду, что такое решение задачи рассчитано на сильных учащихся и может быть рассмотрено либо на занятиях кружка, либо выдано для самостоятельной творческой работы сильному школьнику.

Глава V. ЭЛЕМЕНТЫ МАТЕМАТИЧЕСКОЙ ЛОГИКИ. ПРЕОБРАЗОВАТЕЛИ ИНФОРМАЦИИ

§ 25. О РОЛИ МАТЕМАТИЧЕСКОЙ ЛОГИКИ В ИНФОРМАТИКЕ

Среди задач, для решения которых привлекаются ЭВМ, немало таких, которые по традиции принято называть логическими. Кто не знает шуточной задачи о перевозке волка, козы и капусты с одного берега на другой! В этой задаче властвует не арифметика, а умение рассуждать.

К помощи логики прибегает человек, составляя различные 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 противоречивыми, что такие твердые логические орешки не под силу раскусить и вдумчивому математику.

Из сказанного следует, что ЭВМ необходима при решении логических задач. Необходимо подчеркнуть, что умение использовать логические операции повышает эффективность программирования. Часто, формируя условия в операторе условной передачи управления, программист использует логические операции.

Следует помнить, что и в самой математической логике немало задач, решение которых может быть поручено ЭВМ. Примером таких задач может быть поиск тавтологий, т. е. тождественно истинных высказываний в алгебре высказываний.

Какие из приведенных ниже высказываний истинны:

$$A+BC=(A+B)\cdot(A+C);$$
$$A\to BC=(A\to B)\cdot(A\to C);$$
$$A\sim(BC)\equiv(A\sim B)\cdot(A\sim C)?$$

Такая область математической логики, как алгебра высказываний, хорошо освоена в информатике. В настоящее время нет ни одного языка программирования, который не включал бы основных операций алгебры высказываний.

Приобщение ЭВМ к математической логике продолжается. Созданы и работают языки программирования, которые способны оперировать понятиями алгебры предикатов. ЭВМ может вести «рассуждения» в заданной ей системе суждений. Примером может быть язык *PROLOG*.

§ 26. АЛГЕБРА ВЫСКАЗЫВАНИЙ. ОСНОВНЫЕ ЛОГИЧЕСКИЕ ОПЕРАЦИИ

Познакомить школьников с алгеброй высказываний можно по следующему плану:

1. Уточнить понятие об алгебре, приведя при этом примеры алгебр (алгебра натуральных чисел, алгебра рациональных чисел, алгебра многочленов, алгебра векторов и т. д.).

2. Ввести определение понятия «высказывание», закрепить примерами.

3. Определить основные операции: логическое сложение, логическое умножение, логическое отрицание.

4. Изучить свойства отдельных логических операций и их совместные свойства (распределительные свойства).

5. Познакомить с методами простейших тождественных преобразований (операции «поглощения» и «склеивания»).

6. Ввести понятия о тождественно истинных, тождественно ложных и эквивалентных высказываниях.

7. Познакомить с таблицами истинности сложных высказываний.

8. Привести примеры прикладных задач, для решения которых можно с успехом применить аппарат алгебры высказываний.

На первый взгляд предлагаемый план кажется очень емким. Найдется ли время для его реализации? Опыт показывает, что все указанные темы хорошо прорабатываются за 6—8 уроков. Значение полученных знаний столь важно, что этого времени жалеть не следует.

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

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

Высказывания 1 и 3 истинные, высказывание 2 ложное.

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, то $A = 0$.

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 называются логическим умножением или конъюнкцией, а результат операции — 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 — «Минск является столицей Белоруссии»; B — «В Минске проживает 1543 тыс. человек».

Тогда логическим произведением, или конъюнкцией, этих высказываний будет составное высказывание: «Минск является столицей Белоруссии, и в Минске проживает 1543 тыс. 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 восемь равно 24»; B — «Кит — морское животное».

Составное высказывание, полученное после выполнения операции импликации, будет таким: «Если трижды 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 многоточия. Указание совершить операцию эквивалентности над высказываниями A и B записывается так: $A \sim B$, иногда $A \Leftrightarrow B$ (читается: «A эквивалентно B»).

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 двуместными, 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: $\bar{A}$ — «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 выс-

казываний. В таблице истинности приведены все значения сложных высказываний при всех комбинациях значений используемых высказываний в каждой из логических операций. Знак 1 обозначает слово истинно, а 0 — ложно.

A	1	1	0	0
B	1	0	1	0
A+B	1	1	1	0
AB	1	0	0	0
$A \to B$	1	0	1	1
$A \sim B$	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:

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

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

Составные высказывания можно обозначать одной буквой, например: $X=A+B$; $Y=\overline{A \to \bar{B}}$; $K=\overline{C\bar{D} \to B}$.

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

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

A	B	$\bar{A}$	$\bar{B}$	$\overline{AB}$	$\overline{AB}+\bar{A}$	$\overline{\overline{AB}+\bar{A}}$
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. Ниже в таблице приведены три пары таких высказываний. Эквивалентные высказывания можно соединять знаком $\equiv$.

При необходимости в формулах вместо одного высказывания можно вписывать ему эквивалентное.

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

$A+B \equiv B+A$ $\quad$ $AB \equiv BA$ $\quad$ $A \Leftrightarrow B \equiv B \Leftrightarrow A$

Используя таблицу, 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, какое значение истинности принимают высказывания, входящие в них; например:

$A+\bar{A}$; $\quad$ $AB \Leftrightarrow BA$;

$A \to A$; $\quad$ $(A+B) \Leftrightarrow (B+A)$.

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: $A\bar{A}$ или $A\bar{B} \cdot (A \to B)$.

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 о свойствах этих операций будет сказано дальше.

Отметим также два важных свойства операций импликации и эквивалентности: $A \to B \equiv \bar{A}+B$; $\quad A \Leftrightarrow B \equiv AB+\bar{A}\bar{B}$.

В том, что последние два высказывания верны, легко убедиться, если составить таблицы истинности для каждого из них. Эти два свойства позволяют преобразовывать формулы, в которых встречаются знаки импликации и эквивалентности, в тождественные им формулы, но такие, в которых будут использоваться только операции логического умножения, сложения и отрицания.

§ 27. МОДЕЛИРОВАНИЕ ЛОГИЧЕСКИХ ОПЕРАЦИЙ НА ЭВМ

В предлагаемых для изучения школьнику языках программирования, таких, как Бейсик и Паскаль, логические операции имеются. Встречаются, однако, случаи, когда на конкретной ЭВМ в числе машинных операций логических нет.

В этом случае пользователь может научить ЭВМ выполнять эти операции, моделируя их средствами арифметики. Моделирование логических операций учитель может рассматривать как интересную и поучительную задачу на программирование.

Ниже приводится вариант создания арифметических моделей для логического умножения, логического сложения, логического отрицания, импликации и эквивалентности. Приводятся таблицы: в левой задается логическая операция, в правой — ее арифметическая модель.

Логическое отрицание:

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

A	$1-A$
1	0
0	1

Формула $1-A$ есть арифметическая модель логического отрицания; пользуясь ею, получают те же результаты, что и при выполнении операции $\bar{A}$. В этом следует убедиться подстановкой.

Логическое сложение:

A	B	$A+B$
1	1	1
1	0	1
0	1	1
0	0	0

A	B	$A+B-AB$
1	1	1
1	0	1
0	1	1
0	0	0

Импликация:

A	B	$A \to B$
1	1	1
1	0	0
0	1	1
0	0	1

A	B	$1-A+AB$
1	1	1
1	0	0
0	1	1
0	0	1

Эквивалентность:

A	B	$A \sim B$
1	1	1
1	0	0
0	1	0
0	0	1

A	B	$1-(A-B)^2$
1	1	1
1	0	0
0	1	0
0	0	1

Еще раз подчеркнем, что построены арифметические модели логических операций. Каждое переменное в арифметической моде-

ли принимает числовые значения 0 и 1 и «выдает» значение либо 0, либо 1, которое мы уже рассматриваем как логическое.

Построены модели отдельно взятых логических операций, часто же требуется модель сложного логического высказывания.

Рассмотрим пример получения модели сложного логического высказывания. Пусть необходимо составить таблицу истинности для высказывания, заданного формулой

$$X=(A \to B)\ (\bar{A}+\bar{B}).$$

Арифметическую модель получают заменой:

$$A \to B \text{ на } 1-A+AB;$$

$$\bar{A}+\bar{B}=1-AB.$$

После этого имеем

$$Y=(1-A+AB)\cdot(1-AB).$$

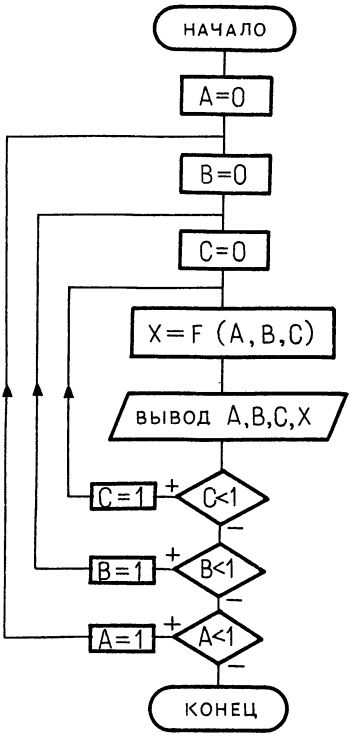


Рис. 25

Полученная формула есть уже формула арифметическая; каждая буква в ней принимает значение не «истина» и «ложь», а числовые: 0 и 1. Вычисленные по этой формуле значения могут быть либо только единицей, либо нулем, но истолковывать их по окончании вычислений следует 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.

Алгоритм для такой работы, если арифметическое выражение содержит три переменные A, B и C, может быть таким, какой изображен на схеме (рис. 25).

Приведенная схема алгоритма содержит три вложенных цикла. Внутренний имеет параметр C, средний — параметр B и внешний — параметр A. Каждый из этих параметров принимает всего два числовых значения: 0 и 1.

Заполнение таблицы истинности при работе по такому алгоритму ЭВМ начнет с вычисления арифметического выражения при $A=B=C=0$. Затем параметр C примет значение $C=1$ и вычислится второе значение арифметического выражения, обозначенного на схеме как $F(A, B, C)$. Таблица истинности будет заполняться сверху вниз.

A	B	C	$F(A, B, C)$
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Для учащихся не составит труда внести изменения в алгоритм, с тем чтобы можно было составить таблицу истинности для выражений, содержащих число переменных, отличное от трех.

Умение составлять таблицу истинности с привлечением ЭВМ лежит в основе алгоритма для решения большого класса логических задач, который разбирается в § 28.

§ 28. АЛГОРИТМЫ РЕШЕНИЯ ЛОГИЧЕСКИХ ЗАДАЧ

Познакомимся с основными алгоритмами решения логических задач. Прежде всего отметим, что обширный класс логических задач решается по схеме:

изучи условие задачи;

обозначь используемые и разыскиваемые высказывания символами (буквами или буквами с индексами):

используя логические связи между данными высказываниями, составь логическое выражение для всех требований задачи;

вычисли все значения этого логического выражения;

проверь полученное решение по условию задачи.

Пусть дана 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 в.».

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

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

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 (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 в.» — П.

«Сосуд изготовлен в III в.» — Т.

«Сосуд изготовлен в IV в.» — Ч.

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

Алеша сказал: «Это сосуд греческий и изготовлен в V в.» — такое сложное высказывание можно записать так: ГП.

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 $Г\bar{П}+\bar{Г}П=1$.

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

Рассуждая аналогично, получим еще два сложных истинных высказывания: $Ф\bar{Т}+\bar{Ф}Т=1$; $\bar{Г}\bar{Ч}+ГЧ=1$.

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 $ФГ=0$, $ПТ=0$, $ПЧ=0$ и $ТЧ=0$, или, что то же самое: $\bar{Ф}+\bar{Г}=1$; $\bar{П}+\bar{Т}=1$; $\bar{П}+\bar{Ч}=1$; $\bar{Т}+\bar{Ч}=1$.

Содержание каждого высказывания 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

$$Г\bar{П}+\bar{Г}П=1;\quad Ф\bar{Т}+\bar{Ф}Т=1;\ \bar{Г}\bar{Ч}+ГЧ=1;\ \bar{Ф}+\bar{Г}=1;\ \bar{П}+\bar{Т}=1;\ \bar{П}+\bar{Ч}=1;\ \bar{Т}+\bar{Ч}=1.$$

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

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

Обозначим эту формулу так: $X=F$ (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 истинности и найти ту единственную строку, в которой $X=1$.

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	
.	.	.	.	.	
.	.	.	.	.	
.	.	.	.	.	

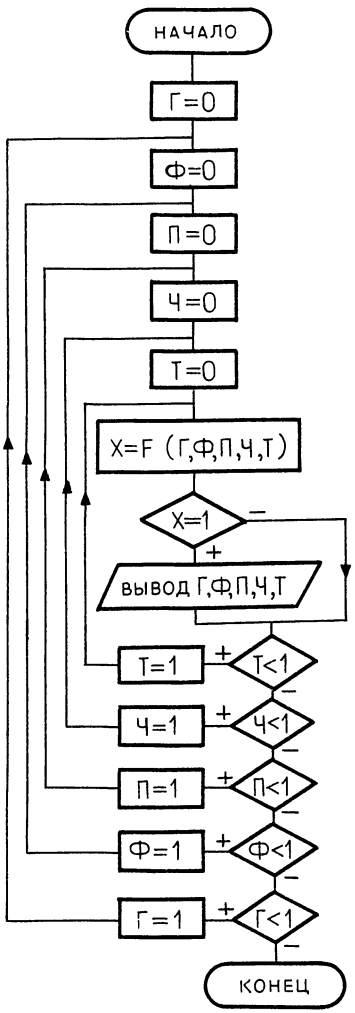


Рис. 26

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

Рассмотрим схему алгоритма (рис. 26). В нее введен один распознаватель $X=1$ и один оператор вывода значений Г, Ф, П, Ч, Т. 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, Ч, Т) 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 в.».

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 четырех 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 проходили 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?»

Условие задачи можно выразить следующей системой логических уравнений:

$$F_1=\bar{М}_3\bar{М}_6\bar{М}_9\cdot(М_1М_4+М_1М_7+М_4М_7)=1;$$

$$F_2=\bar{Ф}_3\bar{Ф}_6\bar{Ф}_9\,(Ф_1+Ф_4+Ф_7)\,\bar{Ф}_4\bar{Ф}_8\bar{Ф}_9=1;$$

$$F_3=(И_1+И_2+И_5)\cdot\overline{(И_1С_2+И_2С_3+И_4С_5+И_5С_6+И_7С_8+И_8С_9)}=1;$$

$$F_4=\bar{Х}_7\bar{Х}_8\bar{Х}_9\,(Ф_1Х_2+Ф_1Х_3+Ф_2Х_1+Ф_2Х_3+\ldots+Ф_9Х_7+Ф_9Х_8)=1;$$

$$F_5=(С_3+С_6+С_9+С_2О_3+С_5О_6+С_8О_9)\,\bar{С}_7\bar{С}_8\bar{С}_9)=1;$$

$$F_6=\overline{(М_1М_2+М_1М_3+М_2М_3+М_4М_5+\ldots+М_8М_9)}\times$$

$$\times(Ф_1Ф_2+Ф_1Ф_3+\ldots+Ф_8Ф_9)=1;\quad F_7=О_1+О_9=1.$$

Эту систему можно заменить одним сложным логическим уравнением $F=F_1\cdot F_2\cdot F_3\cdot F_4\cdot F_5\cdot F_6\cdot F_7=1$.

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.

Большое количество задач такого типа учитель найдет в книге «Некоторые способы решения логических задач» (автор В. Е. Шевченко).

Рекомендуется прорешать задачи, в которых условия выражаются с использованием импликации, например задача 5 на с. 32.

Большой класс логических заданий возникает при изучении математических игр и при попытке их моделировать на ЭВМ. Алгоритмы многих классов задач удобно оформлять в виде так называемых граф-схем.

Граф-схемы — это чрезвычайно эффективный и широко используемый в информатике способ задания логических алгоритмов. Для иллюстрации сказанного рассмотрим алгоритмы игры «Побеждает чет». Сначала остановимся на частном случае игры.

Условия игры. Играют двое, ходят поочередно. К началу игры имеется группа из 25 предметов. Выполнить ход — это значит взять из общей группы от 1-го до 4-х предметов, накапливая эти предметы у себя. Победителем считается тот, кто сумеет к концу игры накопить четное число предметов.

Беспроигрышная стратегия в игре должна давать ответы на такие вопросы:

1. Каким по очереди вступать в игру: первым или вторым? Иначе говоря, что лучше — начинать игру или ждать хода соперника?

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

Такие вопросы возникают при создании модели любой игры. В данном случае, когда игра конкретна, стратегия очень проста:

1. В игру следует вступать вторым! (Почему это так, разъясним позднее.)

2. Каждый ход следует делать, руководствуясь граф-схемой алгоритма, изображенной на рисунке 27.

Граф-схема — это своеобразная шпаргалка, которую программист помещает в память ЭВМ и обучает ЭВМ пользоваться этой необычной шпаргалкой.

Разъясним условные обозначения. Кружочками с буквами a_1, a_2 и a_3 внутри обозначены три состояния в игре. Весь процесс игры — это переход из одного состояния в другое. Переходы из состояния a_i в состояние a_k обозначены стрелками. Рядом со стрелкой выписаны числа, например 4 (2), что означает: партнер взял четыре предмета — бери при своем ходе два.

На рисунке 28 показана часть граф-схемы, которую следует понимать так: игра находится в состоянии a_2 и соперник — ЭВМ. Человек отделил себе два предмета. Указание для ЭВМ выписано в скобках и означает: бери себе три предмета.

Рекомендуется сыграть несколько партий, чтобы хорошо овладеть чтением граф-схемы.

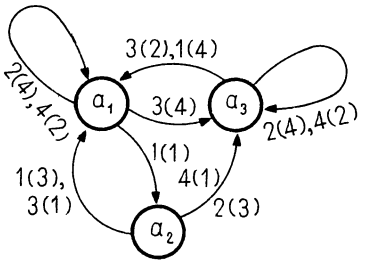


Рис. 27

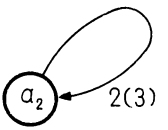


Рис. 28

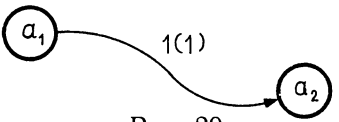


Рис. 29

Каким же образом передать шпаргалку, т. е. граф-схему, для исполнения ЭВМ? Один из путей основан на применении операций алгебры высказываний.

Опишем поведение ЭВМ в состоянии a_1 при всевозможных вариантах действий ее соперника — человека. Из граф-схемы видно, что если в состоянии a_1 человек отделит себе один предмет, то ЭВМ в ответ должна отделить себе тоже один предмет и перевести игру в состояние a_2. На рисунке 29 показан этот фрагмент процесса игры. Все сказанное можно записать так:

$$(A=1)\cdot(G=1)\rightarrow(M=1)\cdot(A=2)$$

где $A=1$ означает, что игра находится в состоянии a_1; $G=1$ — человек взял один предмет; $M=1$ — ЭВМ взяла один предмет; $A=2$ — игра переведена в состояние a_2.

Это же самое на языке Бейсик запишется так:

```
IF (A=1 AND G=1) THEN (M=1 AND A=2)
```

Аналогично рассуждая, получим

```
IF (A=1 AND G=3) THEN (M=4 AND A=2)
IF (A=1 AND G=2) THEN (M=4 AND A=1)
IF (A=1 AND G=4) THEN (M=2 AND A=1)
```

При составлении программы придется выписать 12 таких условий, по четыре на каждое состояние. Ниже приводится текст программы, моделирующей весь процесс игры; при этом ЭВМ не проигрывает.

```
10 'ИГРА ПОБЕЖДАЕТ ЧЕТ
20 N=25: A=1
30 N=N-G-M
40 IF N<=0 THEN PRINT "ВЫ ПРОИГРАЛИ": "ВАША СУММА= "; SG: END
50 PRINT "Я ДЕЛАЮ ХОД" - "; M
60 PRINT "ОСТАЛОВЬ"; N; "ПРЕДМЕТОВ"
70 IF N>=4 THEN I=4 ELSE I=N
```

```
80 PRINT "ВАШ ХОД 0<G<="; I
90 INPUT G: IF G>I OR G<=0 THEN 90
100 SG=SG+G : SM=SM+M
110 IF (A=1 AND G=1) THEN M=1 : A=2 : GOTO 30
110 IF (A=1 AND G=2) THEN M=4 : A=1 : GOTO 30
110 IF (A=1 AND G=4) THEN M=2 : A=1 : GOTO 30
110 IF (A=1 AND G=3) THEN M=4 : A=3 : GOTO 30
110 IF (A=2 AND G=2) THEN M=3 : A=3 : GOTO 30
110 IF (A=2 AND G=3) THEN M=1 : A=1 : GOTO 30
110 IF (A=2 AND G=1) THEN M=3 : A=1 : GOTO 30
110 IF (A=2 AND G=4) THEN M=1 : A=1 : GOTO 30
110 IF (A=3 AND G=2) THEN M=4 : A=3 : GOTO 30
110 IF (A=3 AND G=4) THEN M=2 : A=3 : GOTO 30
110 IF (A=3 AND G=3) THEN M=2 : A=1 : GOTO 30
110 IF (A=3 AND G=1) THEN M=4 : A=1 : GOTO 30
```

На этом примере проиллюстрированы возможности алгебры высказываний в описании логически запутанных игровых ситуаций (после № 110 номера строк идут в порядке возрастания). Полученные формулы алгебры высказываний легко записать на конкретном языке программирования.

Для примера приведем известную задачу о доме с привидениями, которую У. Р. Эшби предложил читателям книги «Введение в кибернетику» (М., 1959).

Дорогой друг!

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

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

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

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

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

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

Поручим ЭВМ вести себя так, как ведет себя дом с привидениями из Замогилья. Представим дом с привидениями как некий «черный ящик», у которого имеется два выхода: по одному из них транслируется Смех, по другому — Пение. Есть у «черного ящика» и два входа: по одному из них ведется воздействие «Игра на Органе», по другому — «Сжигание Ладана».

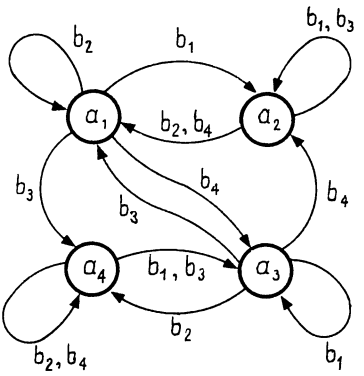


Рис. 30

Вводятся понятия о состоянии «черного ящика»:

a_1 — нет Смеха и нет Пения;

a_2 — есть Смех и нет Пения;

a_3 — нет Смеха и есть Пение;

a_4 — есть Смех и есть Пение.

Воздействия подаются парами, и поэтому можно выделить такие черты воздействия:

b_1 — нет игры на Органе и нет сжигания Ладана;

b_2 — нет игры на Органе и есть сжигание Ладана;

b_3 — есть игра на Органе и нет сжигания Ладана;

b_4 — есть игра на Органе и есть сжигание Ладана.

Соответствующая граф-схема имеет вид, изображенный на рисунке 30.

Анализируя граф-схему, легко увидеть путь, который ведет из состояния a_4 в состояние a_1:

1-я минута — воздействие b_1 (прекратить игру на Органе и зажечь Ладан);

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

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

Если теперь в доме все время будет гореть Ладан (Орган будет молчать), то установится требуемая тишина.

Ниже приводится программа на языке Бейсик, в которой десять первых строк полностью описывают граф-схему. Работая по приведенной программе, ЭВМ ведет себя как описанный дом. Правильно вводя свои воздействия, человек может «успокоить» ЭВМ.

```
10 'ДОМ С ПРИВИДЕНИЯМИ
20 LET A=2 : LET B=1
30 IF A=1 AND B=1 THEN LET C=2
```

```
40 IF A=1 AND B=3 THEN LET C=4
50 IF A=1 AND B=2 THEN LET C=1
60 IF A=1 AND B=4 THEN LET C=3
70 IF (A=2 AND(B=1 OR B=3)) THEN LET C=2
80 IF (A=2 AND(B=2 OR B=4)) THEN LET C=1
90 IF A=3 AND B=1 THEN LET C=3
100 IF A=3 AND B=2 THEN LET C=4
110 IF A=3 AND B=3 THEN LET C=1
120 IF A=3 AND B=4 THEN LET C=2
130 IF (A=4 AND(B=2 OR B=4)) THEN LET C=4
140 IF (A=4 AND(B=1 OR B=3)) THEN LET C=3
150 LET A=C : PRINT A
160 ON A GOTO 170, 190, 240, 260
170 INPUT "ВАШЕ ВОЗДЕЙСТВИЕ В"; B
180 GOTO 20
190 FOR I=1 TO 5
200 SOUND 40, 1
210 SOUND 1000, 1
220 NEXT I
230 GOTO 170
240 SOUND 1000, 10
250 GOTO 170
260 SOUND 40, 10
270 GOTO 170
280 FOR I=1 TO 5
290 SOUND 40, 2
300 SOUND 600, 1
310 SOUND
320 NEXT
```

§ 29. ЭЛЕМЕНТЫ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ В ШКОЛЬНОМ КУРСЕ ИНФОРМАТИКИ

Полное наименование школьного предмета — «Основы информатики и вычислительной техники». Отсюда следует, что учащимся нужно знать, как используются ЭВМ, как они устроены и каковы принципы их конструирования. Речь идет о том, что учащиеся должны быть ориентированы и как будущие пользователи ЭВМ, и как люди, которые в будущем могут избрать профессию, связанную с созданием преобразователей информации. Потребность в такой ориентации школьников имеется. Сводить изучение основ информатики только к обучению их искусству программирования — это значит сознательно сужать образовательное значение предмета.

Может быть, ознакомление школьников с принципами конструирования преобразователей информации потребует много времени? Может быть, потребуется введение многих новых понятий? Может быть, от учителя потребуются инженерные знания, которые он не получил в свое время? На все эти вопросы можно дать один ответ: нет, не потребуется.

Прежде всего в основе теории создания дискретных преобразователей информации лежат аппарат алгебры логики (аппарат алгебры высказываний), сведения о двоичной арифметике и теории кодирования (в самой элементарной ее части). Все эти сведения школьник получает как сведения, необходимые для применения в программировании. Часы, отводимые на знакомство с элементарными преобразователями информации, с алгоритмом синтеза логических устройств,— это небольшая часть курса. Значение же этих сведений в образовании учащихся очень велико. Именно в этой части курса школьник получает доказательство существования дискретных преобразователей информации, в частности доказательство существования самой ЭВМ.

Имеющийся опыт показывает, что на весь раздел, который условно можно назвать «Элементы конструирования преобразователей информации», достаточно 8—10 уроков. Если при этом школьники будут иметь возможность поработать на «полигоне логических устройств», то тема хорошо понимается и усваивается. Известно немало примеров конструирования оригинальных логических устройств силами учащихся.

Обсуждаемый раздел может быть построен по плану:

1. Элементарные преобразователи информации.
2. Структурные формулы и функциональные схемы логических устройств.
3. Двоичный одноразрядный сумматор.
4. Моделирование памяти. Триггер.

§ 30. ЭЛЕМЕНТАРНЫЕ ПРЕОБРАЗОВАТЕЛИ ИНФОРМАЦИИ

Простейшими преобразователями информации являются логические преобразователи дискретного действия. Условно такие преобразователи изображаются так, как показано на рисунке 31. Буквами A_1, A_2, ..., A_n обозначены входы, а буквами X_1, X_2, ..., X_k — выходы преобразователя. По входам преобразователь получает информацию, на выходах демонстрирует результат преобразования. На каждый вход подаются сигналы только двух типов, обозначаемые 0 и 1. Такие и никакие другие сигналы появляются на каждом выходе. В любой момент времени на любом входе и выходе удерживается какой-то один из сигналов: либо 0, либо 1. В процессе работы преобразователя сигналы сменяют друг друга скачком (мгновенно). В этом смысл понятия «дискретность действий».

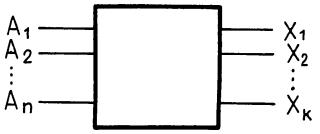


Рис. 31

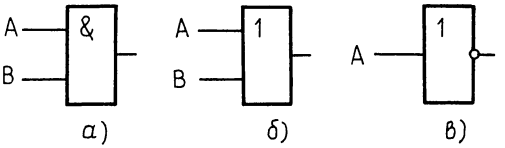


Рис. 32

Определение. *Преобразователи, которые могут, получая сигналы об истинности отдельных простых высказываний, обработать их и в результате выдать значение логического произведения, или логической суммы, или отрицания, называются логическими элементами.*

Логический элемент И предназначен для «вычисления» значения логического произведения. Условное обозначение приведено на рисунке 32,*а*. Элемент имеет два входа: A и B — и выход X, на котором демонстрируется сигнал о значении истинности логического произведения AB.

Логический элемент ИЛИ обеспечивает определение истинности логической суммы (рис. 32,*б*). Логический элемент НЕ служит для вычисления значения истинности высказывания $\bar{A}$ по известной истинности A (рис. 32,*в*).

Сигналы и значения истинности можно реализовать многими способами. Входы и выходы логического элемента могут быть, например, трубками, по которым в элемент подается воздух. Если давление в трубке A равно 1 атм, то можно считать, что на вход A подана 1; если же давление в трубке A равно 0,5 атм, то это означает, что на вход A подан 0. Таким образом, устроенный элемент работает на пневматической основе.

Ниже приводятся электрические схемы логических элементов, состоящие из электромагнитных реле (рис. 33).

Работа элемента И (рис. 33,*а*) протекает так: условимся нажатием кнопки A «сообщать» элементу о том, что высказывание

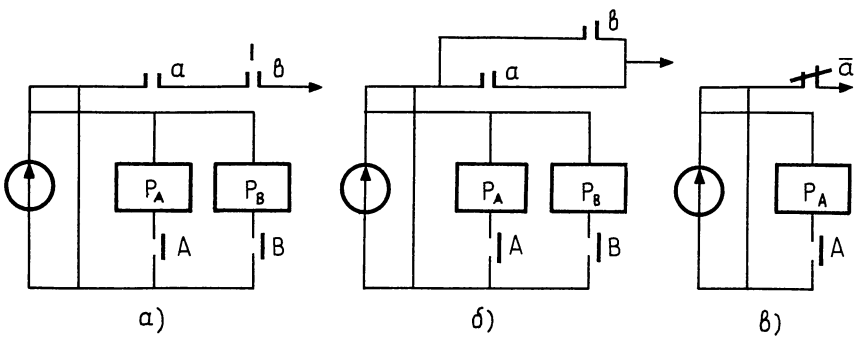


Рис. 33

$A=1$, а нажатием кнопки B есть сигнал о том, что $A=0$. Если нажать кнопку A, то по катушке реле P_A потечет ток и контакты A замкнутся. То же случится с контактами B, если нажать на кнопку B. Сигнал 1 на выходе зажжет лампочку X — это будет означать, что $AB=1$. Из схемы элемента И видно, что лампочка зажжется только при одновременном нажатии кнопок A и B, т. е. при $A=B=1$.

Из схемы элемента ИЛИ видно, что он выдает сигнал 1, если нажата какая-то одна из кнопок: A или B — или обе одновременно. Элемент НЕ при нажатии кнопки A ($A=0$) выдает сигнал $X=1$, а при нажатии на A он выдаст сигнал $X=0$.

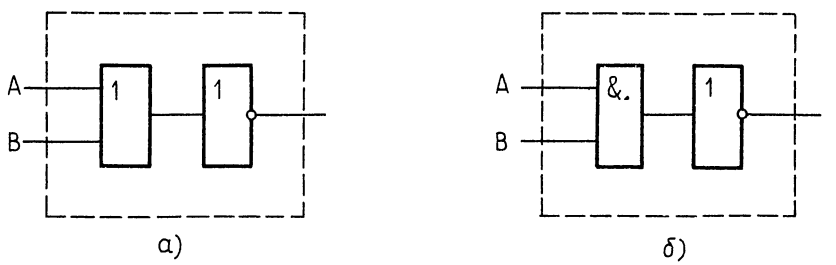
Замечание. Сигнал, получаемый на выходе любого логического элемента, можно использовать не только для зажигания лампочки, но и для подачи на вход другого элемента В этом случае сигнал, передаваемый в другой элемент, выполнит там ту же работу, что и нажатие кнопки.

§ 31. СТРУКТУРНЫЕ ФОРМУЛЫ И ФУНКЦИОНАЛЬНЫЕ СХЕМЫ ЛОГИЧЕСКИХ УСТРОЙСТВ

Так как сигнал, выработанный одним логическим элементом, можно подавать на вход другого элемента, то это дает возможность образовывать цепочки из отдельных логических элементов. На рисунке 34 показаны примеры таких цепочек. На рисунке 34,*а* элемент ИЛИ соединен с элементом НЕ, на рисунке 34,*б* элемент И соединен с элементом НЕ. Каждую такую цепочку будем называть логическим устройством: ведь она состоит из нескольких элементов. Соответствующие схемы называются функциональными схемами.

На рисунке 35 приведены две более сложные функциональные схемы. Анализируя функциональную схему, можно понять, как работает логическое устройство.

Не менее важной формой описания логических устройств является структурная формула. Покажем на примере, как выписывают формулу по заданной функциональной схеме. Обратимся к рисунку 35,*а*. Ясно, что в элементе И осуществляется логическое умножение значений $\bar{A}$ и B, т. е. вычисляется значение



а) б)

Рис. 34

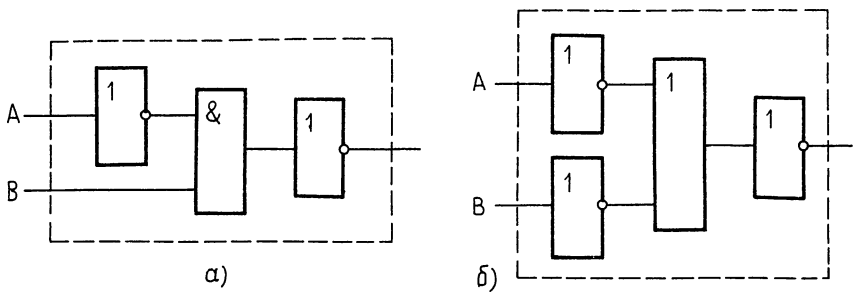


Рис. 35

произведения $\bar{A}B$. Над результатом в элементе НЕ осуществляется операция отрицания, т. е. вычисляется значение выражения $\overline{\bar{A}B}$. Формула $X=\overline{\bar{A}B}$ и есть структурная формула логического устройства.

Структурная формула устройства, функциональная схема которого изображена на рисунке 35,*б* имеет вид $X=\overline{\bar{A}+\bar{B}}$. Устройство вычисляет отрицание значения логической суммы значений $\bar{A}$ и $\bar{B}$.

Важно уметь решать и обратную задачу: по структурной формуле вычерчивать соответствующую функциональную схему.

Пример. Пусть дана структурная формула $X=\overline{\bar{A}+B\cdot A}$.

Вычертим соответствующую функциональную схему. Какие же для этого потребуются логические элементы и как их следует друг с другом соединить?

Ясно, что над значением A будет выполняться операция отрицания — для этого необходим элемент НЕ. Необходим и элемент ИЛИ: в нем будут складываться значения $\bar{A}$ и B. Схема будет такой, какой показано на рисунке 36.

Структурные формулы исчерпывающим образом описывают логический преобразователь информации. Из формулы ясно, истин-

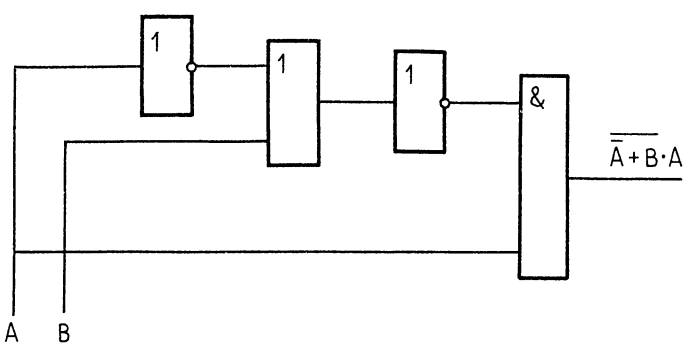


Рис. 36

ность каких сложных высказываний может определить данное логическое устройство. Тождественно преобразуя структурную формулу, можно получить более простую, которая задаст и более простой логический преобразователь информации.

§ 32. ДВОИЧНЫЙ ОДНОРАЗРЯДНЫЙ СУММАТОР

Из отдельных логических элементов можно составить устройство арифметического назначения. Пусть необходимо изготовить устройство для сложения двух n-разрядных двоичных кодов. Ясно, что при сложении цифр i-го разряда (рис. 37) складывать приходится цифры кодов (a_i и b_i) и к сумме прибавлять еще и цифру p_{i-1} — «перенос» из предшествующего младшего разряда. В результате сложения получаем цифру кода суммы i-го разряда — C_i и цифру p_{i+1} — «перенос» в следующий разряд, старший. Работа в любом одном разряде при сложении кодов сводится к сложению трех одноразрядных двоичных чисел. Устройство, которое выполняет такую работу, называется сумматором.

На рисунке 38 изображено условное обозначение сумматора. Сумматор имеет три входа и два выхода. На рисунке 39 показано, как из четырех сумматоров можно составить устройство для сложения двух четырехразрядных двоичных кодов. На рисунке 40 изображена функциональная схема сумматора, для его конструирования потребовалось 5—И, 3— ИЛИ, 1— НЕ.

Убедитесь в том, что сумматор складывает три одноразрядных двоичных числа. Заполните таблицу, описывающую его работу, следя за работой по схеме и наблюдая за каждым логическим элементом.

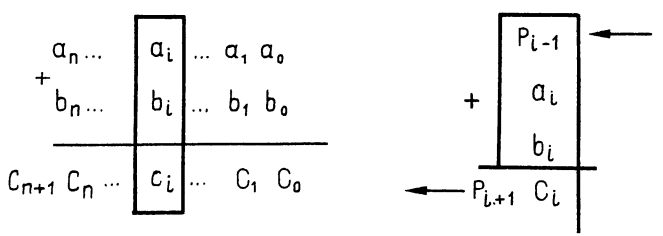


Рис. 37 Рис. 38

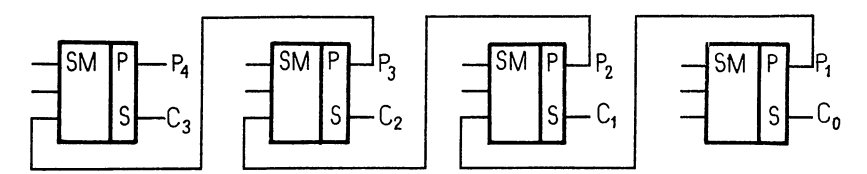


Рис. 39

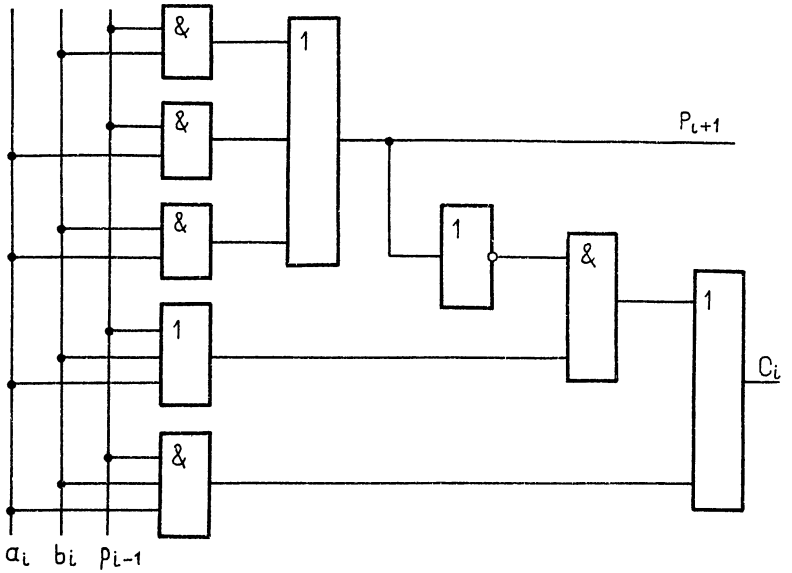


Рис. 40

§ 33. МОДЕЛИРОВАНИЕ ПАМЯТИ. ТРИГГЕР

Все рассмотренные выше преобразователи информации могут преобразовывать только слова, записанные в двоичном алфавите, с помощью букв 0 и 1. Преобразователи информации, изготовленные на базе логических элементов И, ИЛИ, НЕ, хорошо различают двоичные слова.

Возникает вопрос: нельзя ли научить преобразователи запоминать информацию, запоминать и вспоминать двоичные слова? Ясно, что для этого необходимо сначала научиться запоминать отдельные буквы двоичного алфавита.

Определение. *Устройство, которое может запоминать буквы 0 и 1, демонстрировать их, а в случае необходимости и забывать, называется триггером.*

Оказывается, что для изготовления триггера достаточно иметь логические элементы И, ИЛИ и НЕ. Изучение внутренней конструкции триггера в программу средней школы не входит. Рассмотрим принцип действия триггера и способ обращения с ним. Из нескольких типов триггеров выбран так называемый триггер со счетным входом. Условное изображение его дано на рисунке 41. Такой триггер имеет один вход (A) и два выхода (X_1 и X_2). В любой момент времени на его выходах демонстрируются буквы 0 и 1, и причем если $X_1 = 1$, то $X_2 = 0$, и наоборот.

Как работает триггер? Как им управляют? Пусть в некоторый момент времени $X_1 = 1$, а $X_2 = 0$. Если теперь на вход A подать на

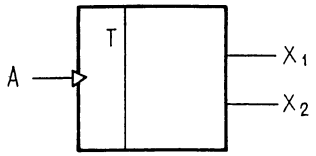


Рис. 41

некоторое время, на доли секунды, сигнал 1, то на выходах X_1 и X_2 буквы сменятся на противоположные: теперь $X_1=0$, а $X_2=1$. Эти буквы будут демонстрироваться на этих выходах сколь угодно долго, до тех пор, пока на вход A вновь не будет подан сигнал 1. Каждая кратковременная подача сигнала 1 на вход A «переключает» триггер, вызывает смену букв на его двух выходах. При необходимости сигналы (буквы), снимаемые с его выходов, можно передавать по проводам в другие устройства.

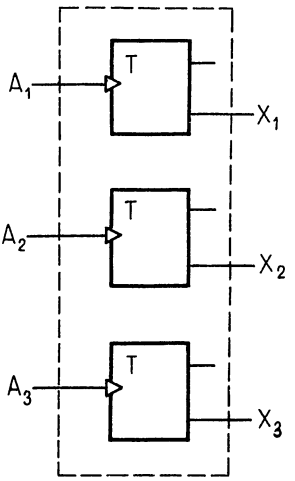


Рис. 42

Несколько триггеров можно объединить в группы — так называемые регистры. На рисунке 42 изображен регистр из трех триггеров. Такой регистр можно использовать для запоминания и демонстрации трехбуквенных двоичных слов и можно рассматривать как простейшее запоминающее устройство (ЗУ). Рассмотрим управление таким ЗУ. Пусть, например, в какой-то момент времени $X_1=X_2=X_3=0$. Это означает, что ЗУ помнит и демонстрирует на своих выходах слово 000. Подадим на вход A_1 сигнал 1— это вызовет смену буквы на выходе X_1 и на внешних выходах появится новое слово: 100. Если теперь подать сигнал 1 на выход A_3, то это приведет к стиранию слова 100 и запоминанию слова 101. Так, воздействуя на входы отдельных триггеров, образующих регистр, можно одно двоичное слово заменять для хранения на другое.

Заметим, что если в регистр входит n триггеров, то при таком ЗУ можно организовать запоминание и демонстрацию n-разрядных двоичных слов.

Оперативная память ЭВМ часто конструируется в виде набора регистров. Как правило, один регистр образует одну ячейку памяти, каждая ячейка в оперативном ЗУ имеет номер.

Из сказанного можно сделать вывод, что ЭВМ состоит из огромного числа отдельных логических элементов, образующих все ее узлы и память.

Глава VI. ПРОГРАММНОЕ УПРАВЛЕНИЕ ЭВМ

§ 34. ПОСТАНОВКА МЕТОДИЧЕСКИХ ЗАДАЧ

В данной главе рассматриваются методические проблемы, возникающие при решении основных учебных задач:

обучение программированию для ЭВМ;

обучение применению ЭВМ.

Прежде всего отметим, что необходимым условием работы учащегося является прочное усвоение понятия «алгоритм» и «алгоритмический способ деятельности».

Уже на первых уроках по введению в программирование учитель еще раз обращает внимание школьников на важнейшую особенность исполнения алгоритмов: исполнение хорошо проработанных алгоритмов сводится к формальному, чисто механическому выполнению отдельных указаний, к исполнению одного за другим чрезвычайно простых действий.

Именно в этом основа возможности передачи работы по исполнению алгоритма от человека к машине. Через алгоритмизацию проходит путь к автоматизации, к самоуправлению в работе ЭВМ.

Учитель должен постараться как можно ярче и яснее раскрыть связь между теорией алгоритмов и программированием. В чем суть шага от алгоритма к программе? Искусство разработки алгоритма переходит в искусство составления программы.

Программирование для ЭВМ в широком смысле слова является прикладным разделом теории алгоритмов. Проблема формулируется так: как различные виды умственной деятельности человека могут быть реализованы средствами ЭВМ? Предполагается, что передача нагрузки на ЭВМ осуществляется после завершения формализации моделируемого процесса и создания алгоритма.

Такое развернутое определение задачи программирования адресуется учителю — школьник же должен знать, что программирование есть процесс записи готового, ранее разработанного, обдуманного и проверенного алгоритма в виде текста программы. При этом школьной программой предусматривается изучение только одного способа формирования программ, а именно записи программ на языке программирования.

Такой вид программирования, как ручное программирование,

в деталях не рассматривается. Учитель только дает представление о возможностях программирования вне развитого языка, говорит об особенностях и недостатках такого программирования.

Подчеркнем, что существуют различные способы взаимодействия человека и ЭВМ в процессе решения задачи. Эти способы меняются, следуя за совершенствованием вычислительных машин. Доля машинного участия при этом растет.

На самых первых машинах с программным управлением алгоритм решения задачи записывался самим пользователем на внутреннем языке машины, на языке кодов. Это значит: человек, готовивший ЭВМ к решению задачи, выписывал одно за другим все необходимые указания для ЭВМ, обдумывал каждое условие ветвления процесса решения задачи, предусматривал одно за другим все необходимые указания для ЭВМ, обдумывал каждое условие ветвления процесса решения задачи, предусматривал конкретные ячейки памяти для хранения исходных данных, промежуточных результатов и принимал решение о форме выдачи результата. Все детали процесса решения он записывал в виде слов двоичного алфавита, в виде двоичных кодов.

Такой способ программирования принято называть ручным программированием. Это важный способ составления программ — его достоинства и недостатки следует знать. К недостаткам относят трудоемкость работы: при поставлении программы решения сложной задачи приходится выписывать очень много команд и условий (иногда десятки тысяч), очень трудно следить за взаимосвязью между отдельными блоками программы. Все это отягощается тем, что все тексты выписаны в виде двоичных слов (позднее восьмеричных), разобраться в тексте кем-то записанной программы очень сложно: комментарии и разъяснения помогают очень мало. Анализ текста программы бывает иногда столь затруднителен, что легче составить программу заново, нежели разобраться в ее тексте, составленном другим программистом. Все это приводило к тому, что подготовка программы (по уже выверенному алгоритму) и ее отладка длились очень долго, порой несколько месяцев.

В 50-е гг., когда программирование вручную было господствующим, для повышения эффективности использования ЭВМ, для обеспечения их более полной загруженности прибегали к увеличению числа штатных программистов. Запись программ стали вести не в виде слов двоичного алфавита, а в виде восьмеричных слов — это делало каждую команду в среднем сжатой в три раза ($110111011_2 = 673_8$). Двоично-восьмеричные и обратные замены кодов ЭВМ делала сама. И все-таки громоздкость программ мешала обмену опытом, их специфичность отпугивала непрофессионалов. Все это снижало эффективность применения вычислительной техники в народном хозяйстве.

Недостатки ручного программирования были столь серьезными, а потребность в широком использовании ЭВМ столь значитель-

ной, что к решению проблемы повышения труда программистов и операторов были привлечены большие силы: работали математики, лингвисты, инженеры-специалисты по вычислительной технике. В начале 60-х гг. появились первые практические подходы к замене ручного программирования автоматическим.

Не следует, однако, думать, что ручное программирование как метод изживает себя. Нет, дело в том, что существуют программы, которые целесообразно писать именно вручную, самым эффективным образом используя аппаратурные особенности всех элементов и узлов ЭВМ. Речь идет о программах-трансляторах и интерпретаторах — так называют программы, с помощью которых реализуются идеи автоматического программирования.

§ 35. ОТ ЯЗЫКА АЛГОРИТМИЧЕСКОГО К ЯЗЫКУ ПРОГРАММИРОВАНИЯ

Языки программирования — это специально разрабатываемые искусственные языки, предназначенные исключительно для записи алгоритмов, исполнение которых поручается ЭВМ. Подчеркнем, что алгоритм, записанный на алгоритмическом языке (например, на языке схем), ориентирован на исполнителя-человека; алгоритм, записанный на языке программирования, ориентирован на исполнителя-ЭВМ.

Каждый язык программирования имеет свой алфавит, в котором под буквами понимаются все символы, использующиеся в этом языке для записи выражений. В каждом из языков программирования действует своя грамматика — четкие правила образования слов и фраз на этом языке. Языки могут заметно отличаться друг от друга словарным запасом — набором слов, которые можно использовать для записи. Если язык предназначен, например, экономистам, то в его словаре много терминов, которыми пользуются бухгалтеры-плановики, но нет слов, в которых нуждаются биологи или архитекторы.

Подчеркнем, что на любом из языков программирования нельзя выразить или записать в виде текста многое из того, что мы говорим друг другу, выражая различные чувства или описывая природу. Словарь любого из языков программирования очень мал по объему.

С момента появления первых языков программирования в 1952—1954 гг. и до настоящего времени во всем мире было зарегистрировано создание более 2000 языков. Благодаря языкам программирования подготовка задач к их решению стала более простой и удобной. Использование языков позволяет значительно расширить круг людей, которые могут пользоваться ЭВМ. Изучив рекомендуемый язык, к помощи ЭВМ могут обратиться биолог и химик, астроном и экономист, технолог и конструктор; причем этим специалистам не потребуется помощь профессионального программиста.

Отметим, что среди языков принято выделять близкие друг другу по их назначению. Имеется, например, группа языков, на которых удобно записывать алгоритмы решения вычислительных задач. К ним относятся Алгол, Фортран, Алмир и т. д. В этих языках предусмотрены такие возможности, которые могут облегчить запись алгоритмов решения очень сложных вычислительных задач. Язык программирования Кобол — представитель группы языков, на которых записываются алгоритмы решения экономических и бухгалтерских задач (например, алгоритм начисления заработной платы рабочим завода). Существуют и другие, иным образом предметно-ориентированные языки.

Рассмотрим схему подготовки задачи к ее решению на ЭВМ с помощью языка программирования (рис. 43).

Решению задачи предшествует выбор метода решения, затем разрабатывается алгоритм, который может быть записан в различ-

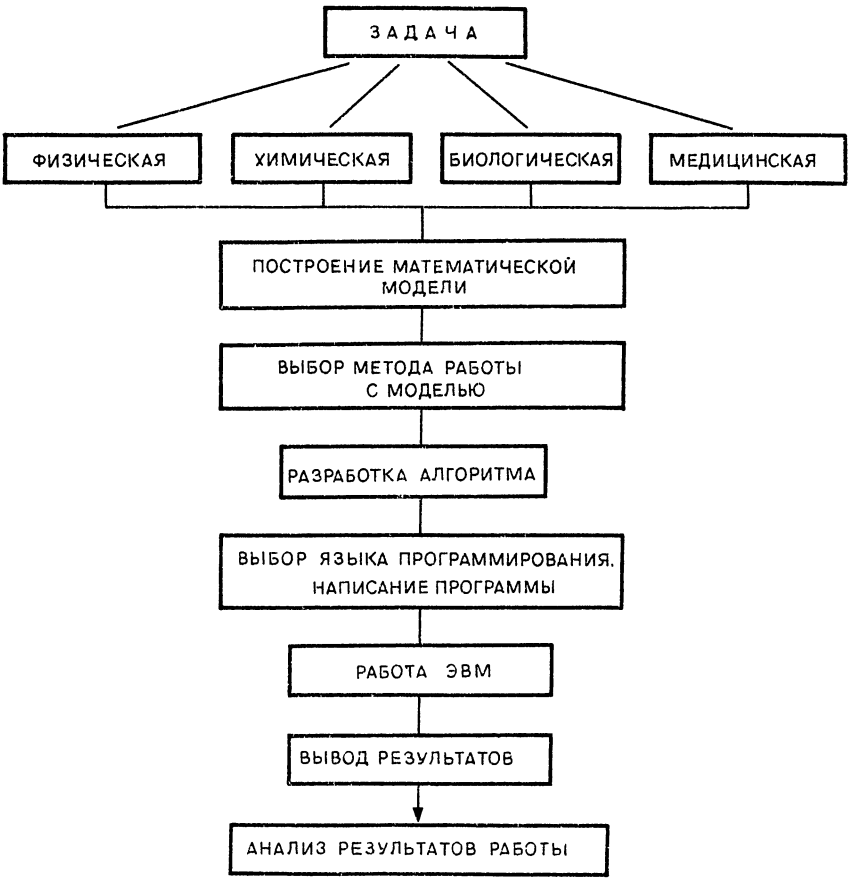


Рис. 43

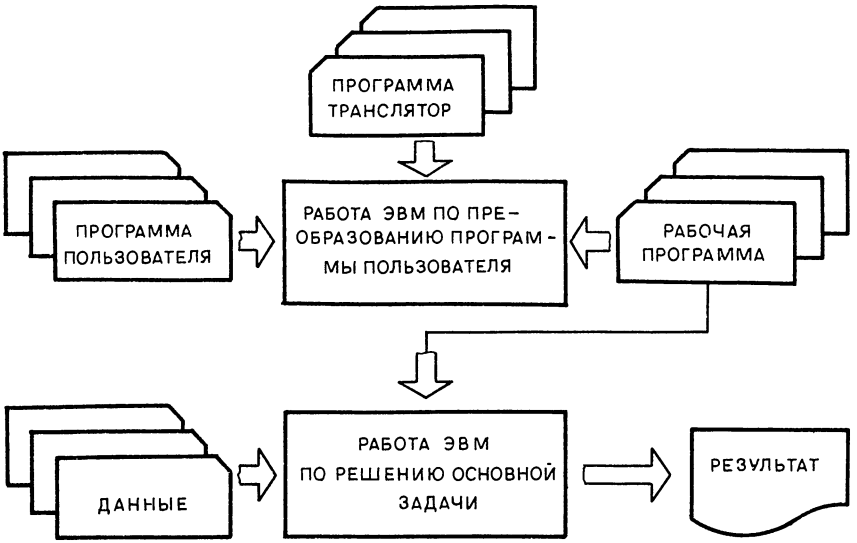


Рис. 44

ных формах: в виде инструкции, в виде схемы или как-то иначе. После этого осуществляется выбор наиболее подходящего для записи полученного алгоритма языка программирования. Затем алгоритм записывается на выбранном языке, и полученный текст называют программой пользователя. Эта программа «набивается» на перфокарты или перфоленту, и с их помощью, а очень часто и просто с помощью пишущей машинки, вводится в оперативную память вычислительной машины.

После того как программа пользователя размещена в памяти, на помощь привлекается сама машина. На рисунке 44 показано, как ЭВМ обрабатывает программу пользователя, привлекая для этого специальную программу, называемую транслятором. Транслятор заранее внесен в память машины.

Подчиняясь указаниям транслятора, ЭВМ переводит текст алгоритма с языка программирования на внутренний язык машины, т. е. на язык кодов ее команд. При этом она сама отводит место в своей памяти не только для текста создаваемой рабочей программы, но и для всех исходных, промежуточных и окончательных результатов предстоящих ей вычислений.

Программа-транслятор решает важную задачу: с помощью ЭВМ переводит текст с языка программирования в текст рабочей программы, действуя по которой эта же ЭВМ и решает основную задачу.

Тексты программ-трансляторов разрабатываются высококвалифицированными специалистами, как правило, это делается вручную.

Приобретая ЭВМ, пользователь, как правило, приобретает и несколько трансляторов. Оригиналы трансляторов записываются на магнитные ленты, а для повседневного применения используются их копии, которые хранятся на магнитных дисках, барабанах и других носителях информации.

На одной и той же ЭВМ можно использовать трансляторы с нескольких языков программирования — это позволяет выбирать для решения на ЭВМ наиболее подходящие языки программирования.

Наряду с программами-трансляторами используются и программы-интерпретаторы — о них будет рассказано ниже.

§ 36. СХЕМА И АЛГОРИТМ РАБОТЫ ЭВМ

Напомним, что всякая вычислительная машина с программным управлением состоит из следующих узлов (устройств):

устройства ввода или, если это крупная машина, нескольких устройств ввода различного типа;

устройства управления (УУ);

арифметико-логическое устройство (процессора);

оперативно запоминающего устройства (ОЗУ);

внешнего запоминающего устройства или нескольких типов такого устройства (ВЗУ);

устройства вывода или нескольких типов такого устройства.

Схематически взаимосвязь между всеми узлами можно изобразить так, как показано на рисунке 45.

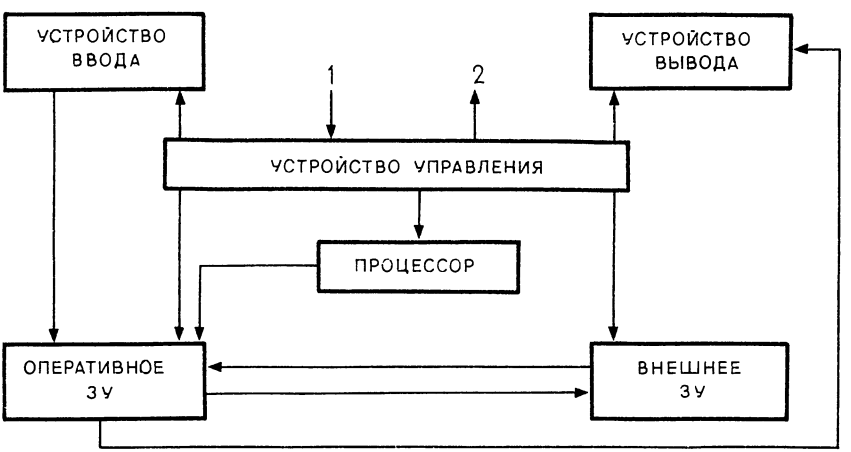


Рис. 45

Устройства ввода служат для занесения в оперативную память ЭВМ как текста программы, так и всех исходных данных.

В настоящее время разработан ряд типов устройств ввода. Если информация цифровая и буквенная, то для ввода используют клавиатуру с достаточным числом клавиш на пульте. Скорость ввода информации с помощью клавиатуры ограничена возможностями человека — примерно 200 символов в минуту.

Если необходимо ввести графическую информацию, поместить в память какую-нибудь линию, используют панели графического ввода с контактными карандашами или световые перья. Имеются и устройства ввода речевой информации.

Большое распространение получили устройства ввода, в которых информация считывается с перфокарты или с перфоленты. Такой способ более громоздок: ведь сначала всю информацию необходимо «набить» на перфокарту или перфоленту и лишь затем использовать в ЭВМ. Достоинство такого способа в том, что как перфоленту, так и перфокарту можно использовать многократно и считывание с них информации осуществляется с большой скоростью. Устройства для занесения информации на перфоленту являются дополнительными узлами, в каждом из таких устройств осуществляется двоичное кодирование каждого символа; на перфокарту фиксируются коды этих символов. Устройства ввода с перфокарты и с перфоленты работают с такими скоростями: с перфоленты в среднем за одну секунду считывается 100 символов; с перфокарты считывается за одну минуту до 700 перфокарт.

Устройство управления (УУ) обеспечивает координацию действий всех узлов машины в соответствии с программой.

В процессоре осуществляются все арифметические и логические операции над закодированной информацией. Современные процессоры выполняют отдельные операции с гигантскими скоростями — в среднем до сотен миллионов операций в секунду.

Оперативное запоминающее устройство (ОЗУ) служит для хранения выполняемой программы и основной части обрабатываемой информации. ОЗУ, как правило, самые быстрые из запоминающих устройств; в них применяются ячейки памяти на электронных триггерных регистрах или магнитных элементах (ферритовых сердечниках).

ОЗУ состоит из ячеек, каждая из которых имеет свой номер (адрес). Отыскание числа по его адресу и считывание этого числа выполняются с большой скоростью. Если ОЗУ изготовлено на магнитных сердечниках, то любое число будет найдено и считано за 2 миллионные доли секунды; если же ОЗУ использует память на электронных триггерах, то время обращения к памяти не более полумиллионной доли секунды. Совершенствование оперативных ОЗУ продолжается.

К внешним запоминающим устройствам относят магнитные барабаны, магнитные диски, ленточные магнитофоны. На таких носителях информации можно записывать большие объемы как

числовой, так и буквенной информации. Так, магнитный диск серийного образца может вместить до 100 Мбайт. Магнитный барабан имеет объем от 100 до нескольких сот Мбит, скорость обращения не более 100 м/с. Более медленными являются накопители информации на магнитных лентах.

Каждое из внешних устройств с помощью каналов связи подключается к устройству управления и процессору ЭВМ.

Среди устройств вывода довольно широко применяется автоматическое алфавитно-цифровое печатающее устройство (АЦПУ). По окончании работы над задачей ЭВМ с помощью АЦПУ выводит на широкую бумажную ленту таблицы, текст, различные сводки. В одной строке размещается до 128 символов.

Результат работы ЭВМ может быть показан на экране дисплея. (Более подробно такой способ вывода информации обсуждается дальше.)

Если в результате решения задачи следует вычертить кривую линию, или схему, или карту, то в этом случае хорошо использовать графопостроитель. С его помощью ЭВМ сама вычерчивает линии на бумаге.

Результаты работы ЭВМ могут быть записаны на диски или барабаны. Для этого есть специальные устройства вывода.

§ 37. ПРИНЦИП ПРОГРАММНОГО УПРАВЛЕНИЯ

Выше было сказано, что все исходные данные в виде кодов с помощью устройства ввода размещаются в ячейках внутреннего запоминающего устройства. В этом же запоминающем устройстве находится и рабочая программа. Каждая команда, входящая в программу, хранится, как правило, в одной ячейке. Каждая ячейка ОЗУ независимо от того, что в ней хранится: код числа или код команды,— имеет свой номер, называемый адресом ячейки.

Работа ЭВМ по программе начинается с того, что УУ «заглядывает» в так называемый счетчик адреса команд (СЧАК). Дело в том, что перед началом работы в СЧАК помещается адрес ячейки, в которой хранится самая первая команда программы. С выполнения именно этой команды ЭВМ начинает решение задачи. После выполнения этой команды в СЧАК появляется номер следующей команды, за ним — еще один и т. д., пока в СЧАК не окажется номер команды: «конец работы по программе»,— выполнив которую ЭВМ останавливается.

Устройство управления находит ячейку по ее адресу, затем расшифровывает команду и настраивает арифметическое устройство на ее выполнение. УУ обеспечивает подачу в арифметическое устройство (АУ) тех чисел, над которыми выполняется операция. Эти числа находятся по адресам, которые содержатся в команде.

Устройство управления после того, как оно снабдило арифметическое устройство необходимыми ему числами и настроило

его на выполнение конкретной операции, на некоторое время «отходит в тень»: дает время арифметическому устройству на выполнение операции.

Арифметическое устройство выполняет порученную ему операцию над кодами чисел. Полученный результат АУ демонстрирует на своих выходах. Окончив работу, АУ дает УУ сигнал об этом. Устройство управления направляет результат в ячейку памяти, адрес которой предусмотрен либо программистом, либо транслятором. После этого УУ вновь «заглядывает» в СЧАК и начинает рассмотренный цикл сначала.

Может оказаться так, что очередная команда есть команда вывода информации. В этом случае устройство управления обращается к тому из устройств вывода, которое предусмотрел программист, и настраивает его на работу. По окончании вывода всей подготовленной для этого информации устройство вывода дает сигнал УУ.

Может оказаться и так, что очередная команда, к выполнению которой должна приступить ЭВМ, есть указание о вводе дополнительной информации. В этом случае УУ настраивает на работу требуемое устройство ввода. По окончании ввода информации УУ вновь «заглядывает» в СЧАК, и рабочий цикл повторяется еще раз.

Так, следуя программе, выполняя одну команду за другой, УУ координирует совместную работу всех блоков ЭВМ.

В этом случае самоуправление ЭВМ при ее работе осуществляется программой.

§ 38. ПОДГОТОВКА ЗАДАЧ К РЕШЕНИЮ НА ЭВМ. МАТЕМАТИЧЕСКИЕ МОДЕЛИ

Задачи по физике, химии, геометрии и биологии очень часто решаются алгебраическим путем. При решении задачи полезно, а иногда и необходимо составлять уравнение или неравенство, в ряде случаев — систему уравнений или неравенств.

Все количественные связи между постоянными и переменными величинами выражены в составленных уравнениях. В математике эту часть работы над задачей называют созданием математической модели. Если разработанная модель, например система уравнений, очень сложная, то для работы с моделью привлекается ЭВМ.

После того как математическая модель создана, необходимо принять решение о методе решения полученного уравнения или системы уравнений. Перед использованием ЭВМ выбранный метод необходимо сформулировать в виде алгоритма. Часто на этом этапе работы над задачей вычерчивают схему алгоритма.

Следующим шагом в решении задачи является подготовка текста программы на подходящем языке программирования.

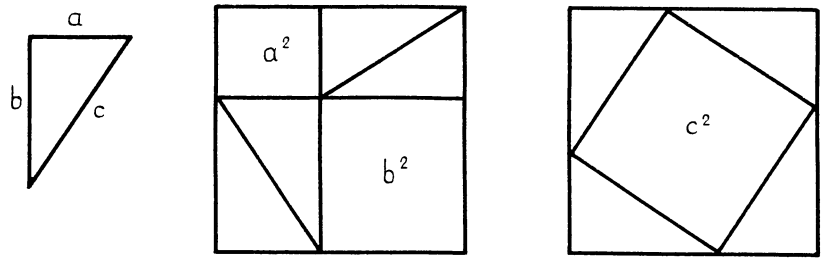


Рис. 46

После этого в работу вступает ЭВМ, которая действует в соответствии с программой и выдает результат.

Затем человек полученные с помощью ЭВМ ответы сопоставляет с условием задачи и принимает окончательное решение о существе дела в целом.

Весь описанный процесс подготовки задачи к ее решению на ЭВМ можно представить так, как показано на рисунке 43.

Еще раз подчеркнем, что разработка модели — чрезвычайно важный этап. От того, какая будет построена модель, зависит успех в решении задачи. Школьники должны знать, что создание модели — это творческий акт; только человек, глубоко разобравшийся в существе задачи и при этом обладающий воображением, может придумать хорошую модель.

На рисунке 46 приводится геометрическая модель для доказательства теоремы Пифагора. Чертежи настолько ясны, что теорема $c^2 = a^2 + b^2$ кажется очевидной.

Наряду с алгебраическими и геометрическими моделями применяются и логические модели (пример будет рассмотрен ниже). Все эти модели относятся к математическим моделям.

Решение задач на моделях — один из сильнейших методов решения задач вообще. На практике используются модели очень сложных процессов, изучение которых без построения моделей иногда невозможно. С помощью ЭВМ, используя модели, экономисты изучают экономику отраслей народного хозяйства, биологи моделируют поведение отдельных клеток и некоторых простых живых систем, инженеры-энергетики строят модели гигантских электрических сетей.

Более простым, но не менее важным является метод физического моделирования. Примеры на каждом шагу: портной использует манекен, сапожник шьет обувь по «модели ноги» — по колодке. Испытание самолетной катапульты не сразу поручают человеку — сначала катапультируют манекен. Вместо испытаний самолетного крыла в воздухе его поведение исследуют в аэродинамической трубе. Теория моделей — важный раздел науки.

§ 39. ОБУЧЕНИЕ ЯЗЫКУ ПРОГРАММИРОВАНИЯ

Внимание педагогов язык Бейсик привлек в связи с поисками наиболее пригодного для обучения детей языка программирования. Одной из предпосылок в отборе языка является то, что лучше предложить школьникам менее развитый, но завершенный язык, нежели какое-то подмножество таких развитых языков, как Алгол или Фортран.

Язык Бейсик считается хорошо продуманным с дидактических позиций, с точки зрения обучения детей. Обучение этому языку во многом в глазах учащихся напоминает изучение родного, естественного языка. Школьники знакомятся с алфавитом, с тем, как из отдельных букв составляются слова, узнают смысл каждой языковой конструкции. Каких-либо принципиальных особенностей в ознакомлении школьников с этим языком программирования учитель не встречает.

Однако, как это часто бывает, не успел появиться исходный вариант языка, созданный на основе педагогических концепций обучения, как сейчас же (часто под влиянием коммерческих стимулов) начинают возникать новые «усовершенствованные» версии Бейсика. Язык дополняется весьма разнообразными средствами, и его педагогическая направленность начинает теряться в массе дополнений.

Досадно, что новые версии языка подчас столь сильно отличаются друг от друга, что программы, написанные на разных версиях языка, не могут «стыковаться». Примерами могут быть болгарская версия языка Бейсик, реализованная для ЭВМ «Правец», и версия Бейсик-MSX, реализованная для ПЭВМ японской фирмы «Ямаха». Программы, написанные на Бейсике-MSX, не работают на ЭВМ «Правец», и наоборот. Аналогичная ситуация и с версией Бейсик-6, используемой в словацких ПЭВМ MPD-2.

Из сказанного следует, что важно ясно представлять себе основную часть языка; понимать, что является дополнительными средствами, и знать, какой минимум сведений о языке Бейсик должен быть усвоен учащимися. Конечно, преподаватель сам должен знать ту версию языка, которая реализована на применяемых им ЭВМ.

Ниже приводятся некоторые советы для учителей по отбору содержания изучаемого предмета, а также основных средств языка.

Еще раз подчеркнем, что на первых занятиях при изучении алфавита и общей структуры языка Бейсик учителю рекомендуется активно использовать сопоставление с родным языком школьника.

Уже на первых уроках учитель имеет возможность очень активно воздействовать на воображение учащихся. Так, разъясняя понятие «алфавит», он формирует в сознании школьника более

общее и современное представление об алфавите, закрепляет те представления об алфавите, которые уже вводились ранее при рассказе об алфавитном способе представления информации. Понятие «буква» в сознании школьника заменяется более общим и обогащенным понятием — «символ алфавита». В связи с этим школьник более просто относится и к таким необычным утверждениям, что целое число 2136 в языке Бейсик есть слово, которое может подвергаться преобразованию. Такое переосмысливание хорошо известных понятий: «алфавит», «буква», «слово» — играет существенную роль в формировании алгоритмической культуры учащегося.

Может быть, для некоторых мысль, которую мы намерены подчеркнуть, будет казаться тривиальной — не сказать об этом нельзя при изучении основных средств языка, нельзя продвигаться далее, если нет уверенности в том, что все рассмотренное изучено и закреплено. У этой мысли есть и научно-педагогическое основание: языки программирования обладают малой избыточностью, в них нет ничего лишнего (в основных средствах), они обладают повышенной, образно говоря, «чувствительностью к синтаксическим ошибкам».

Приступая к изучению понятия «арифметическое выражение», учитель вместе со школьниками должен вспомнить, как это понятие вводится в математике. Следует помнить, что учащиеся часто называют выражениями и неравенства, и уравнения. При изучении Бейсика (и других языков программирования) понятие «арифметическое выражение» не только уточняется, но и расширяется.

При рассмотрении этой темы учитель хорошо продумывает систему упражнений. Школьники должны научиться уверенно «переводить» с языка обычной математики на язык Бейсик любые арифметические выражения, хорошо пользоваться скобками, задавая требуемую последовательность операций, а также научиться формировать сложные функции и включать их в состав арифметических выражений. В этом разделе занятий учитель может использовать опыт, приобретенный на уроках алгебры.

Преподавателю нужно помнить, что в некоторых версиях языка Бейсик наряду с арифметическими выражениями рассматриваются и логические выражения. В предлагаемой методике эти возможности языка не входят в число основных, обязательных для всех учащихся. В данной книге рассматриваются методы замены логических выражений их арифметическими моделями.

Приступая к ознакомлению учащихся с основными операторами языка Бейсик, учитель опирается на их знания об алгоритмах и подчеркивает, что текст программы есть новая форма записи алгоритма.

Изучение основных операторов языка может строиться по такой схеме:

формируется задача, требующая введения в язык особой

конструкции; задается повод для введения в состав средств языка оператора конкретного назначения;

новое средство разъясняется на примерах;

дается формальная структура оператора.

Конечно, в этой схеме учитель найдет возможность что-то изменить. Это лишь рекомендуемая схема.

Арифметический оператор присваивания LET изучается на первом уроке. Это один из основных операторов языка; опыт показывает, что два урока на изучение и закрепление этого оператора необходимы.

Школьник должен хорошо разобраться во всех возможных применениях оператора LET: снабжение переменных числовыми значениями (10 LET N=25.3), вычисление значений отдельных выражений и присваивание полученных значений переменным. Здесь учащиеся знакомятся с так называемым форматом оператора. Формат задает структуру оператора — это краткое яркое определение оператора. Школьник впервые встречается с понятием «метка» или «номер оператора», под которым этот оператор входит в текст программы.

Нужно помнить, что в некоторых версиях Бейсика служебное слово LET можно опускать, однако в начале изучения языка школьникам об этом говорить не следует.

Второй урок рекомендуется завершить небольшой контрольной работой.

Изучению оператора вывода результатов работы ЭВМ на экран или на печать — PRINT — следует уделить особое внимание. Учителю нужно заготовить несколько ярких примеров, иллюстрирующих возможности этого оператора, чтобы школьники увидели, как ЭВМ может размещать числа и тексты на экране дисплея. В результате учащиеся будут уверенно использовать экран дисплея и печатающее устройство (принтер). Взаимодействие школьников с ЭВМ не должно тормозиться из-за плохого или неуверенного применения оператора PRINT во всех его возможностях.

В курсе информатики проблема оформления результатов на экране рассматривается специально: школьники знакомятся с тем, как формируются на экране таблицы. Это сложная для учащихся часть курса, и поэтому она поставлена в конец.

При первом знакомстве с оператором PRINT учитель имеет возможность показать ряд простых, но практически очень важных его применений. Завершая уроки по этой теме, учитель показывает учащимся пример хорошо организованного вывода результатов вычислений, при этом следует использовать только изученные операторы — LET и PRINT.

Завершая знакомство с операторами LET и PRINT, учитель может подчеркнуть для учащихся, что с помощью этих операторов пишутся программы, соответствующие любым линейным алгоритмам.

При знакомстве учащихся с операторами передачи управления — GOTO и IF — преподавателю нужно учитывать, что в теоретическом плане включение этих операторов в систему средств языка Бейсик делает эту систему средств алгоритмически полной. Иначе говоря, располагая операторами LET, PRINT, GOTO, IF и END, можно записать программу, соответствующую любому алгоритму. Это не означает, что изучаемые далее средства языка являются второстепенными — нет, это важные средства: с их помощью программы записываются кратко, удобно для обозрения; с их помощью обеспечивается оперативное взаимодействие человека, решающего задачу, и ЭВМ.

В языке Бейсик условиями ветвления являются арифметические отношения, хорошо знакомые учащимся. Особых затруднений при освоении операторов IF и GOTO, как правило, не возникает. Большую роль играет система упражнений, которую использует преподаватель. Здесь, однако, есть ряд вопросов, с которыми учителю следует познакомиться.

Рассмотрим простой пример — пусть дан фрагмент программы:

```
5 INPUT "N="; N
10 P=1 : I=1
15 P=P*I
20 I=I+2
25 IF I>N THEN 35
30 GOTO 15
35 PRINT "P="; P
40 END
```

Выполняя эту программу, ЭВМ вычислит произведение первых нечетных чисел, самое большое из которых не превышает заданного числа *N*.

Внимательный школьник может сообразить, что программу можно упростить, если условие ветвления — оператор IF — записать иначе. Вот какой вариант программы может быть:

```
5 INPUT "N="; N
10 P=1 : I=1
15 P=P*I
20 I=I+2
25 IF I<N THEN 15
26 PRINT "P="; P
30 END
```

В полученном варианте на один оператор меньше, а именно на один оператор GOTO. В первом варианте один оператор (IF I > N THEN 35) обходил другой оператор (GOTO); такие элементы вносят путаницу в программу, при анализе программы оператор

Рис. 47

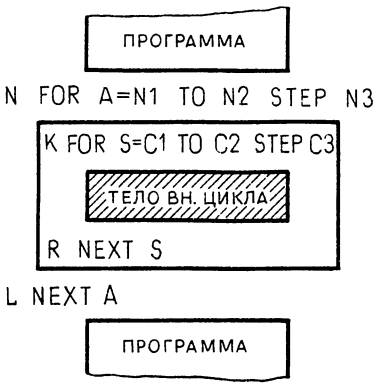


Рис. 48

GOTO заставляет делать большие «скачки» по тексту программы. Если таких случаев, когда один оператор (GOTO) обходит другой, много, то в совокупности они создают большие трудности в чтении программы.

Учащимся можно посоветовать, как можно меньше использовать оператор GOTO. Заметим, что высказанные соображения будут использованы и при обсуждении методики изучения оператора цикла FOR — NEXT.

Начиная изучение оператора цикла, преподаватель подчеркивает, что этот оператор является одним из составных средств языка. С помощью оператора цикла можно более кратко записать целый фрагмент программы. Рисунки 47 и 48 выполняются школьниками. На рисунке 48 изображен один цикл внутри другого.

Учитель обращает особое внимание на понятие «параметр цикла» или управляющая переменная. Ученикам нужно научиться использовать то обстоятельство, что шаг цикла (приращение управляющей переменной) может быть целым и дробным, положительным и отрицательным.

Ранее при рассмотрении циклических алгоритмов обращалось внимание на две структуры таких алгоритмов. Схемы их приведены на рисунке 49.

Учитель обращает внимание если не всех учащихся, то по крайней мере наиболее сильных, на структуру цикла, изображенного на рисунке 49,*б*. Этот тип цикла, когда условие проверяется на входе в «тело цикла», называется цикл ПОКА: «пока условие α истинно, все операторы, образующие тело цикла, выполняются». Учащиеся должны увидеть в таком цикле обобщение оператора GOTO, когда в нем нет обхода одним оператором GOTO другого такого же оператора. Школьникам рекомендуется использовать циклы типа ПОКА и избегать применения циклов типа ДО. Здесь учитель, не используя термин «структур-

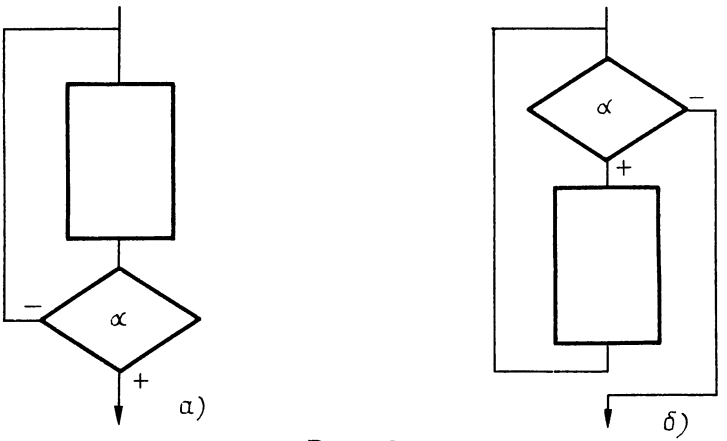


Рис. 49

ное программирование», начинает вводить его элементы в сознание юного программиста.

Заметим, что учитель должен разобраться в причинах появления структурного программирования, понимать, что структурное программирование применяется при составлении программ, в частности при написании больших программ. Со структурным программированием можно знакомить школьников на конкретных примерах, но это следует делать обоснованно и не торопясь.

Подчеркнем, что уже при изучении схем алгоритмов всех трех типов показывалось, что каждый модуль имеет один вход и один выход. На всех рисунках модули помещались в отдельные, обведенные рамкой прямоугольники.

Следующими близкими по своему назначению являются операторы READ — DATA и INPUT. Эти операторы позволяют снабжать переменные, входящие в программу, различными значениями. Иногда термин DATA разъясняется как своеобразное хранилище информации. Учителю рекомендуется разобрать это понятие с учениками более подробно.

Информационное хранилище DATA можно представить себе в виде некоторого колодца, в который один за другим опускаются числа. Колодец заполняется, и после этого из него можно черпать информацию, извлекая числа одно за другим в соответствии с правилом: последнее положенное число извлекается первым. Учитель подчеркивает, что это правило следует учитывать при размещении чисел в DATA. Еще одно обстоятельство важно: число, извлеченное из DATA, открывает путь для извлечения следующего числа. Из DATA извлекают числа, а не их копии. Когда из DATA будет извлечено число, которое туда было помещено первым, больше никакой информации из DATA получить не удастся.

Чтение чисел из DATA осуществляется с помощью оператора READ; при этом один оператор может дать указание снабдить числовым значением не только одну, но и несколько переменных.

Уверенное использование операторов READ — DATA позволяет формировать самые разнообразные массивы из исходных чисел. От подготовки данных часто во многом зависит простота в решении задачи.

Оператор INPUT в Бейсике позволяет реализовать диалоговый режим работы ЭВМ с человеком. Такого рода демонстрации учитель заготавливает заранее. В числе демонстрации должны быть простые игры, например «Побеждает чет» или «НИМ» и др.

Приступая к ознакомлению учащихся с оператором DIM, учитель повторяет понятия «переменная с индексом» и «переменная с двумя индексами». В основном варианте Бейсика переменные с бóльшим числом индексов, чем два, не применяются. Школьники должны уверенно формировать вручную двумерные массивы из данных им чисел, не путать понятия «строка» и «столбец». Следует хорошо закрепить представление о размещении массива в памяти ЭВМ.

Только после этого учитель может учить формированию массивов в памяти ЭВМ с применением оператора DIM. Рекомендуется хорошо проработать совместное применение оператора FOR — NEXT и оператора DIM в сочетании с операторами READ — DATA и INPUT. Закрепление умения применять оператор DIM следует осуществить в лабораторной работе.

Изучение встроенных функций в Бейсике должно перекликаться с теми рекомендациями, которые учащиеся получили при изучении функций, реализованных на микрокалькуляторе. Следует повторить главное из уже известного детям, напомнить об ошибках округления, о том, что значения аргумента для тригонометрических функций нельзя брать произвольными. Желательно вычисление функции свести к вычислению значения функции от произвольного значения аргумента, от аргумента, соответствующего условию.

Для некоторых школьников могут оказаться незнакомыми такие функции, как ABS (X), SGN (X) и RND (X). Учителю нужно подчеркнуть, что эти функции имеют большое значение для программистов: они часто используются при программировании задач невычислительного характера.

ЭВМ может вычислять значения сложных функций или функций от функций. Конечно, следует заранее заготовить две-три простые демонстрации. Примером вычисления сложных функций может быть такая программа:

```
 5  INPUT X
10  PRINT COS (ATN (SQR (X)))
15  PRINT LOG (EXP (SQR (X)))
20  END
```

На эту тему при изучении средств языка достаточно отвести один урок.

Несколько особо среди средств языка Бейсик стоят операторы GOSUB — RETURN. Изучение этих операторов проводится в конце курса, после того как все ранее рассмотренные средства языка усвоены и закреплены. Понятие «подпрограмма» и понятие оператор «передачи управления с возвратом» (GOSUB — RETURN) — самые сложные для школьников средства языка, и поэтому их изучение начинается только при полной уверенности учителя в прочных знаниях школьников по другим средствам языка.

Рассмотрим пример, который учитель может использовать для закрепления материала.

Пусть необходимо подсчитать величину биномиального коэффициента C_n^m. В качестве расчетной формулы будем использовать формулу

$$C_n^k = \frac{n!}{k!\,(n-k)!}.$$

Ясно, что вычисление чисел $n!$, $k!$ и $(n-k)!$ осуществляется с помощью подпрограммы. Возможный вариант программы в целом может быть таким:

```
10 '------------- ВЫЧИСЛЕНИЕ БИНОМИАЛЬНОГО КОЕФФ. -------
20 INPUT N, M
30 K=N
40 GOSUB 140
50 A=L
60 K=M
70 GOSUB 140
80 B=L
90 K=N-M
100 GOSUB 140
110 PRINT "CNM="; A/(B*L)
120 END
130 '----------------- ПОДПРОГРАММА ---------------------
140 '--------------- ВЫЧИСЛЕНИЕ ФАКТОРИАЛА --------------
150 L=1
160 FOR I=1 TO K
170 L=L*I
180 NEXT I
190 RETURN
```

К столь же сложным средствам мы относим и оператор DEF, позволяющий доопределять набор функций, которые используются при решении задач. Разъясняя этот оператор, учитель приоткрывает для школьников дверь в высшие этажи искусства программирования. Речь идет о том, что в развитых языках про-

граммирования с помощью операторов, аналогичных оператору DEF, можно доопределять набор средств и действий ЭВМ. Можно, например, сначала «научить» ЭВМ строить квадрат, а затем «поручить» ей такое действие: «изобрази на экране квадрат со стороной *a*...» Языки, в которых возможно расширение набора используемых средств, называются расширяющими языками.

Подчеркнем, что в Бейсике с помощью оператора DEF можно определять не только функции от одной переменной. Последний раздел в изучении средств языка Бейсик посвящен уже понятию о программах в целом. Задача учителя — сформировать у школьников ясные представления о том, как решается задача на ПЭВМ, включая сведения о математическом обеспечении такого процесса. Школьники должны разобраться в том, что такое директива, как с помощью конкретных директив организуется процесс решения задач на персональных и больших ЭВМ.

Эта часть курса носит практический характер. Учитель разъясняет и показывает в действии все основные директивы, демонстрирует различные варианты обнаружения ошибок в тех программах, которые написали школьники. Учащиеся научатся сами разбираться в сообщениях об ошибках. Без этого выходить на ЭВМ с серьезной программой школьникам не рекомендуется. Вообще, учитель следит за тем, чтобы начатая школьником работа с программой непременно была закончена.

Предполагается, что школьник будет иметь возможность работать с накопителем информации на магнитных дисках. Учителю следует рассказать о принципе работы мягких дисков, об объеме каждого типа диска, а также как следует пользоваться директивами LOAD и SAVE.

Учащиеся должны быть знакомы с основными приемами вывода информации на печать, а также уметь выводить на печать не только текст программы (директива LLIST), но и результаты работы ЭВМ (директива LPRINT) и другие тексты. Насколько детально следует проводить такое знакомство, учитель решает сам.

§ 40. ПРОГРАММИРОВАНИЕ НА БЕЙСИКЕ

Основная цель занятий по этому разделу курса — научить школьников решать задачи на ЭВМ, воспитать навыки свободного обращения с вычислительной машиной как хорошо освоенным инструментом.

Решение этих вопросов тесно связано с общей проблемой расширения у учащихся представлений о возможностях машинного решения задач.

Достижение поставленных целей организуется на уроках через демонстрацию образцов решения специально подобранных задач в сочетании с самостоятельной работой над задачами таких же классов.

Мы считаем, что при такой методике учителю удастся решить такие частные вопросы:

показать учащимся интересные, практически важные задачи, для решения которых привлечение ЭВМ является обоснованным;

подчеркнуть специфически программистские приемы при разборе конкретных программ, которые школьник может взять на вооружение и применить при решении других задач;

закрепить на практике сведения о всех включенных в курс средствах языка Бейсик;

закрепить умение применять метод моделирования, особенно при решении математических задач.

Учитель строго следит за соблюдением принципа «от простого к сложному», а при наличии класса ПЭВМ старается использовать возможности техники для продуманной индивидуализации обучения.

Подчеркнем, что анализ особенностей решения каждого класса задач должен включать рассмотрение алгоритмической проблемы — это первая и важнейшая часть анализа. Особенности техники программирования, создание «портфеля» программистских приемов — вторая часть анализа любой демонстрационной задачи, предложенной в учебнике.

Демонстрационные задачи решают на каждом ученическом месте. Учитель имеет необходимый раздаточный материал, и каждый ученик «видит» предлагаемый образец программы в действии. Конкретные рекомендации приводятся применительно к каждому параграфу.

1. Суммирование чисел. Приступая к проведению этого урока, учитель ясно представляет общематематическое значение проблемы суммирования членов различных последовательностей.

Напомним, что изобретательность математиков позволила найти изящные формулы для вычисления суммы нечетных чисел, суммы их квадратов и кубов:

$$\sum_{i=1}^{n}(2i-1)=1+3+5+\ldots+(2n-1)=n^2;$$

$$\sum_{i=1}^{n}(2i-1)^2=1^2+3^2+5^2+\ldots+(2n-1)^2=\frac{n(4n^2-1)}{3};$$

$$\sum_{i=1}^{n}(2i-1)^3=1^3+3^3+5^3+\ldots+(2n-1)^3=n^2(2n^2-1).$$

Вывести эти формулы непросто. А как же тогда решить задачу суммирования, если в сумму включаются числа более замысловатого вида: $(3i-1)^2$, $(5i+3)^3$ или какие-либо иные. Найти простую формулу совсем непросто.

Учителю нужно иметь в виду, что ЭВМ с успехом применяются для вычисления частичных сумм бесконечных числовых рядов. Напомним некоторые числовые ряды:

$$1+\sum_{k=1}^{\infty}\frac{1}{k!}=e;\quad 1+\sum_{k=1}^{\infty}(-1)^k\cdot\frac{1}{k!}=\frac{1}{e};$$

$$\sum_{k=1}^{\infty}(-1)^{k-1}\cdot\frac{1}{2k-1}=\frac{\pi}{4};\quad \sum_{k=1}^{\infty}\frac{1}{k^2}=\frac{\pi^2}{6}.$$

Используя ЭВМ, можно суммировать члены таких последовательностей, добиваясь необходимой точности.

Основная цель урока: показать, что задача суммирования членов последовательности с программистской точки зрения — это единая задача. Отыскание суммы членов арифметической прогрессии осуществляется так же, как и суммирование членов геометрической прогрессии,— это пример общности в решении задач, которые ранее рассматривались как разные. Если к моменту проведения этого урока школьники на уроках математики уже познакомились с арифметической и геометрической прогрессиями, то учитель непременно должен показать отличие в подходе к решению задач. На уроках математики ставилась задача суммирования первых n членов каждой прогрессии, а на уроках по программированию в основу суммирования берется последовательное сложение генерируемых машиной слагаемых. При этом общая схема алгоритма едина для самых разнообразных слагаемых.

Учитель подчеркивает, что на уроках математики ставится цель, совершенно недоступная ЭВМ: сегодня мы не можем ожидать от ЭВМ открытия формул типа формулы общего члена последовательности 1, 9, 17, 25, ... Не можем мы ожидать от ЭВМ и анализа членов последовательности, проводимого самой ЭВМ, для получения, например, формулы суммы и первых ее членов. В то же время учитель показывает, что любая вычислительная задача из тех, что решались на уроках математики, решается и на ЭВМ.

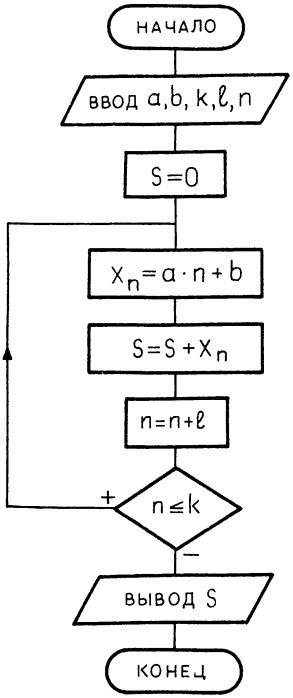
Выделим несколько задач, которые полезно проанализировать с точки зрения их решения на ЭВМ.

Прежде всего рекомендуем учителю подготовить схему алгоритма, изображенную на рисунке 50.

Используя эти схемы (они задают алгоритм суммирования n первых членов арифметической и геометрической прогрессий), в основную схему вместо элемента $x=an+b$ необходимо включить элемент $x=aq^n$, где a — первый член прогрессии; b — разность прогрессии; k — число членов; q — знаменатель геометрической прогрессии; l — приращенные номера члена прогрессии.

Учитель может дать учащимся такие задачи:

1. Составить программу, соответствующую основной схеме, и получить результаты при $a=2$, $b=1$, $k=10$, $l=1$. Какая работа выполнялась на ЭВМ? Какие числа суммировались? Какой результат получен?



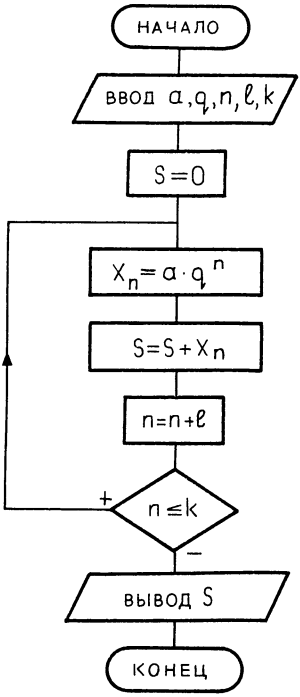


Рис. 50

2. Дана последовательность 3, 7, 11, 15, ... Какие значения следует придать величинам a, b, k, l, чтобы найти сумму 10 членов этой последовательности? Какой будет эта сумма?

3. Какие изменения следует внести в основную схему, чтобы наряду с общей суммой на экране выводился очередной член арифметической последовательности?

4. Дана последовательность 11, 18, 25, 32, ... Составить программу для вычисления суммы $a_1+a_4+a_7+a_{10}$. Чему равна эта сумма?

5. Дана последовательность 5, 9, 13, 17, ... Сколько слагаемых следует взять, чтобы получить сумму, равную 324? Какова будет программа?

6. Дана геометрическая прогрессия 3, 6, 12, 24, ... Чему будет равна сумма 13 ее членов? Сумму вычислить на ЭВМ.

7. Известно, что первый член геометрической прогрессии 3 и ее знаменатель $q=3$. Составить программу для вычисления количества членов прогрессии, дающих в сумме 1092.

8. Дана геометрическая прогрессия, где первый член $a=0{,}32$, знаменатель $q=0{,}4$. Какой номер имеет ее член, величина которого равна $4{,}294967\cdot 10^{-4}$?

9. Дана геометрическая прогрессия 2, 8, 32, 128, ... Составить программу для вычисления суммы $a_1+a_3+a_5+a_7$. Чему равна эта сумма?

10. Составить программу для получения таблицы значений функции, задаваемой формулой $y=2^x$, где $x_0=0$ и $\Delta x=0{,}2$. В таблице должно быть 20 значений.

Еще раз подчеркнем, что все указанные задачи и весь урок — это урок за пультом ПЭВМ. К этому времени учащиеся уверенно пользуются клавиатурой и твердо знают основные конструкции языка Бейсик.

К этому же классу задач по программированию следует отнести и программирование с применением рекуррентных формул.

Школьникам часто предлагаются задачи на вычисление чисел Фибоначчи по формуле

$$a_n=a_{n-1}+a_{n-2},$$

где $a_1=a_2=1$, или задача приближенного вычисления квадратного корня $\sqrt{2}$:

$$x_{n+1}=\frac{x_n}{2}+\frac{1}{x_n}\ (x_0=1).$$

Более интересной, требующей сообразительности и настойчивости в этом классе задач является задача вычисления чисел Бернулли; в основу вычисления кладется соотношение

$$C_k^0\cdot B_0+C_k^1\cdot B_1+C_k^2B_2+\ldots+C_k^{k-1}\cdot B_{k-1}=0,$$

где $B_0=1$.

В программе требуется неоднократное вычисление факториалов различных чисел, биномиальных коэффициентов (C_n^m). Такую программу по силам составить увлеченно работающим школьникам, а также на занятиях кружка.

Известен исторический факт, что А. Лавлейс в своей первой работе, посвященной написанию программ, в качестве задачи рассматривала проблему получения чисел Бернулли. Числа Бернулли относятся к важному для математики типу констант. О числах Бернулли можно прочесть в математической энциклопедии.

В литературе числа Бернулли приводятся в виде обыкновенных дробей: $B_0=1$, $B_1=-\frac{1}{2}$, $B_2=\frac{1}{6}$, $B_3=0$, $B_4=-\frac{1}{30}$, $B_5=0$, $B_6=-\frac{1}{42}$, $B_7=0$, $B_8=-\frac{1}{30}$, $B_9=0$, $B_{10}=\frac{5}{66}$, ...

Числа Бернулли с нечетными номерами, кроме B_1, равны нулю.

2. Задачи на суммирование. Разъяснить способы использования операторов DATA — READ можно на несложных демонстрационных примерах. Учащиеся познакомятся с тем, какие приемы используют программисты при выборке чисел из DATA.

В первой демонстрационной программе (2.1) суммированию подвергаются первые K чисел из общего количества N чисел, хранящихся в DATA. Контроль за количеством выбранных чисел осуществляется в условном операторе 30 IF I > K THEN 40. Счетчик I = I + 1 увеличивает количество извлеченных чисел после каждого извлечения числа из DATA.

```
5 '----------------- СУММИРОВАНИЕ 2.1 -----------------
10 DATA 5, 11, 13, -2, 7, 9, -13, 8, -14, 10
15 M=10 : K=7
20 READ A
25 S=S+A : I=I+1
30 IF I>K THEN 40
35 GOTO 20
40 PRINT "ИСКОМАЯ СУММА S="; S
```

Во второй демонстрационной программе (2.2) суммированию подвергаются только отрицательные числа, которые предшествуют положительным. Контроль за правильностью суммирования осуществляется с помощью «фильтра», в роли которого используется оператор 40 IF A > 0 THEN 80.

```
10 '--------------- СУММИРОВАНИЕ 2.2 -------------------
20 DATA -5, -11, -13, -2, -7, -13.8, -14, 10
30 READ A
40 IF A>0 THEN 80
50 S=S+A
70 GOTO 30
80 PRINT "ИСКОМАЯ СУММА S="; S
90 END
```

В третьей программе (2.3) демонстрируется распространенный прием использования DATA, когда для ограничения набора выбираемых чисел из DATA вводится число, заметно отличающееся от выбираемых. В приводимой программе таким числом является число 1000. Суммирование чисел продолжается до тех пор, пока не будет выбрано и «узнано» это число.

После таких демонстраций учитель может дать более сложные задания.

```
10 '----------------- СУММИРОВАНИЕ 2.3 -----------------
20 DATA -5, -11, -13, -2, -7, -9, -13.8, -14, 1000
30 READ A
40 IF A=1000 THEN 70
50 S=S+A
60 GOTO 30
70 PRINT "ИСКОМАЯ СУММА S="; S
80 END
```

В качестве примера рассмотрим задачу: пусть извлечение квадратного корня из числа 2 осуществляется по итерационной формуле $x_{n+1}=\frac{1}{2}x_n+\frac{1}{x_n}$. Необходимо выяснить, сколько шагов итерации (N) следует осуществить, чтобы достичь требуемой точности 10^{-k}, если начальное значение корня (первая итерация) задается вручную и равно X.

Возможный вариант программы приводится ниже.

```
10 '------------------- ЗАДАНИЕ ------------------------
20 DATA 1E-1, 1E-2, 1E-3, 1E-4, 1E-5, 1E-6, 1E-7, 1E-8,
1E- 9,1E-10
30 PRINT SQR(2) : INPUT "K="; K
35 IF K<=0 OR K>10 THEN END ELSE INPUT "X="; X
40 FOR I=1 TO K: READ EPS: NEXT I
50 Y=X: N=0: PRINT "EPS="; EPS
60 '-----------------------------------------------------
70 WHILE ABS(Y-(.5*Y+1/Y))>EPS
80 Y=.5*Y+1/Y: N=N+1
90 WEND
100 '----------------------------------------------------
110 PRINT "Y="; "N="; N
115 RESTORE: GOTO 30
120 END
```

Работает ЭВМ следующим образом: сначала на экран выдается значение $\sqrt{2}$, вычисленное по встроенному алгоритму. Иначе говоря, ЭВМ находит значение SQR (2). Затем вручную вводится число K, которое указывает, что вычисления должны быть выполнены с точностью до 10^{-k}; после этого вручную вводится значение X.

Хранилище DATA содержит числа 0,1; 0,01; 0,001; ...; 10^{-k}; 10^{-10}. В процессе работы ЭВМ по значению K выберет из DATA требуемую константу и присвоит ее значение переменной EPS, положит EPS=1·E—K. Выборка требуемой константы осуществляется в цикле (строка программы 40).

В строках программы 70—90 выписан цикл типа ПОКА. Его содержание в том, что, пока разность двух приближенных значений $\sqrt{2}$ больше заданного EPS, использование итерационной формулы продолжается, ведется вычисление очередного приближеного значения $\sqrt{2}$.

По окончании цикла ПОКА (WHILE) на экран выводится приближенное, вычисленное с точностью до EPS=10^{-k} значение $\sqrt{2}$ и N — количество шагов итерации, потребовавшееся для достижения заданной точности. Выведя значение Y=$\sqrt{2}$ и N, ЭВМ не остановится: она будет ждать продолжения исследований, будет ждать вывода новых значений K и X. Подчеркнем, что

после каждого использования программы DATA восстанавливается, это осуществляется оператором RESTORE.

Обратим внимание учителя, что в описанной программе с помощью трех операторов — DATA, READ и RESTORE — удалось реализовать последовательный доступ к любому (по счету) числу, хранящемуся в DATA. По существу, была построена модель информационного файла с последовательным доступом. Оператор READ вступает в действие только тогда, когда оператор FOR отсчитает требуемое количество шагов. Оператор RESTORE по окончании работы ЭВМ с выбранным из DATA числом восстанавливает содержимое DATA.

Суммируя сказанное по теме «Использование хранилища DATA», преподаватель может подчеркнуть, что выборка элементов из DATA производится последовательно: один элемент за другим или группами элементов (например, парами — X и Y); при этом выборка может быть начата с любого из хранящихся элементов.

Следует указать, что хранилище DATA целесообразно использовать для хранения физических констант или иных расчетных коэффициентов.

3. Задачи на суммирование (продолжение). Ввод данных, которые сразу же включаются в сумму, на практике встречаются часто. Нетрудно представить себе картину, когда несколько наблюдателей независимо друг от друга передают сведения оператору, а он с пульта ЭВМ в режиме INPUT ведет суммирование и иную обработку поступающей информации. Так бывает, например, на спортивных соревнованиях.

Примером наиболее простой задачи может быть такая: вручную в режиме INPUT ввести N чисел, а ЭВМ вычислит куб их суммы. Такие задачи должны быть выполнены каждым школьником.

Следующий шаг — шаг к усвоению методики формирования одномерных и многомерных массивов и их обработка на ЭВМ. Среди обязательных упражнений — формирование двумерного массива; при этом учитель сообщает учащимся, что элементами массивов могут быть и числа, и символьные величины.

Ученикам следует продемонстрировать конкретные ведомости, в которых требуется суммировать большое количество чисел как «по вертикали» — по столбцам, так и «по горизонтали» — по строкам. Это могут быть производственные ведомости с фамилиями и числовыми данными.

Умение формировать массивы в памяти ЭВМ тесно связано с умением выводить их на печать, с умением свободно обращаться к любому элементу массива. На массив следует смотреть как на математическую модель запоминающего устройства со свободным доступом к любому элементу, хранящемуся в нем.

Три приводимые задачи помогут учителю разработать аналогичные.

Задача 1. Сформировать вручную массив A из четного количества $2N$ элементов и распечатать его в две строки: в первой строке — первые N чисел, во второй строке — вторые N чисел.

Возможный вариант программы.

```
10 '--------------- ЗАДАНИЕ 1 --------------------------
20 INPUT K
30 N=2*K : DIM A(N)
40 FOR I=1 TO N : INPUT A(I) : NEXT I
50 FOR I=1 TO N
60 PRINT A(I);
70 IF I=K THEN PRINT
80 NEXT I
90 END
```

Задача 2. Сформировать вручную два массива и вывести каждый в отдельной строке.

```
10 '----------------- ЗАДАНИЕ 2 ----------------------
20 INPUT N
30 DIM X(N), Y(N)
40 FOR I=1 TO N : INPUT X(I), Y(I) : NEXT I
50 FOR I=1 TO N : PRINT X(I);      : NEXT I
60 PRINT
70 FOR I=1 TO N : PRINT Y(I);      : NEXT I
80 END
```

Задача 3. Сформировать вручную массив из N^2 элементов и вывести его на печать в виде квадратной таблицы.

```
10 '----------------- ЗАДАНИЕ 3 ----------------------
20 INPUT N
30 DIM A(N, N)
40 FOR I=1 TO N
50 FOR J=1 TO N
60 INPUT A(I, J)
70 NEXT J : NEXT I
80 '----------------- ВЫВОД --------------------------
90 FOR I=1 TO N
100 FOR J=1 TO N
110 PRINT A(I, J);
120 NEXT J : PRINT : NEXT I
```

Более трудные задачи и варианты решений предлагаются для изучения учителю.

Задача 1. Сформировать из N элементов одномерный массив A и вывести его на печать в виде K столбцов.

Приведенная программа включает фрагменты: формирование массива A (строка программы 30), проверку правильности ввода

параметра K (число столбцов) — вывод массива A начинается только в том случае, если $K>0$, $K<N$ и $N \vdots K$ (число элементов массива A нацело делится на K). Условия контроля приведены в строках 50 и 60.

Вывод элементов массива на печать осуществляется строка за строкой (строки 90—120). Окончание работы ЭВМ вызывается нарушением условий (строки 50—60), вводом неправильного значения K, например $K=0$.

```
10 '------------------ ЗАДАНИЕ 1 ------------------------
20 INPUT "N="; N : DIM A(N)
30 FOR I=1 TO N : INPUT A(I) : NEXT I
40 INPUT "КОЛ-ВО СТОЛБЦОВ"; K: IF K=0 THEN END
50 IF (K<0 OR K>N) THEN 40
60 IF (N MOD K) <>0 THEN 40
70 '----------------- ВЫВОД -----------------------------
80 FOR I=1 TO N
90 PRINT A(I);
100 IF (I MOD K=0) THEN PRINT
110 NEXT I
120 GOTO 40
```

Приведем распечатку работы ЭВМ с массивом из 12 чисел.

```
К-ВО СТОЛБЦОВ? 4
 10 11 12 13
 14 15 16 17
 18 19 20 21
К-ВО СТОЛБЦОВ? 6
 10 11 12 13 14 15
 16 17 18 19 20 21
К-ВО СТОЛБЦОВ?
 OK
```

З а д а ч а 2. Сформировать массив A из N^2 элементов, вывести его на печать и после этого поменять в нем местами строки с номерами K и L.

```
10 '--------------------- ЗАДАНИЕ 2 ---------------------
  20 INPUT N, K, L
  30 DIM A(N, N)
  40 FOR I=1 TO N
  50 FOR J=1 TO N
  60 INPUT A(I, J)
  70 NEXT J: NEXT I
  80 '-------------------- ВЫВОД А ------------------------
  90 FOR I=1 TO N
  95 FOR I=1 TO N
```

```
100 PRINT A(I, J);
110 NEXT J: PRINT: NEXT I: PRINT
120 '---------------- ОБМЕН A(K, J) И A(L, J) ----------
130 FOR J=1 TO N5 SWAP A(K, J), A(L, J)
140 NEXT J
150 '---------------- ВЫВОД B --------------------------
160 FOR I=1 TO N
170 FOR J=1 TO N
180 PRINT A(I, J);
190 NEXT J: PRINT: NEXT I
```

В программе используется оператор SWAP, с помощью которого две числовые или символьные переменные могут обменяться значениями. В данном случае оператор SWAP используется в цикле (строка 130).

Задача 3. Сформировать из N^2 элементов массив A (N, N) и вывести его на экран в виде квадратной таблицы. В данном массиве поменять местами строки и столбцы, полученный массив B вывести на экран.

```
10 '---------------- ЗАДАНИЕ 3 ------------------------
20 INPUT N
30 DIM A(N, M)
40 FOR I=1 TO N
50 FOR J=1 TO N
60 INPUT A(I, J)
70 NEXT J : NEXT I
80 '---------------- ВЫВОД A --------------------------
90 FOR I=1 TO N
100 FOR J=1 TO N
110 PRINT A(I, J);
120 NEXT J: PRINT: NEXT I: PRINT
130 '---------------- ВЫВОД B --------------------------
140 FOR J=1 TO N
150 FOR I=1 TO N
160 PRINT A(I, J);
170 NEXT I: PRINT: NEXT J
```

4. Простейшие задачи статистики. Появление ЭВМ в школе, несомненно, вызовет выдвижение новых по тематике задач для учащихся. Это будут задачи, решение которых на ЭВМ будет в глазах школьника естественным и оправданным. В числе таких задач станут простейшие задачи статистики; можно предположить, что роль статистики в познании мира может скоро заметно измениться. Такие понятия, как генеральная совокупность, выборка, вариационный ряд, характеристики вариационного ряда, станут для школьной математики столь же обычными и не менее важными понятиями, как медиана, апофема, вписанный угол, монотонность функции, ее максимум или производная.

Урок посвящается одной задаче — напомнить и уточнить представления учащихся о средних нескольких чисел; показать, что при программировании лучше воспользоваться формулой (*) (с. 133), из которой при различных значениях k получаются расчетные формулы для вычисления среднего арифметического, квадратического, кубического и гармонического.

Учитель привлекает внимание школьников к используемой формуле, еще раз подчеркнув, что отыскание такой обобщающей формулы есть творческий акт.

Было бы очень хорошо, если бы учитель на этом же уроке смог показать в работе подготовленную заранее демонстрационную программу по статистике. В этой программе можно было бы показать вариационный ряд, гистограмму, вычислить или увидеть несколько характеристик вариационного ряда. Может быть, можно было проиллюстрировать понятие о дисперсии. Шестивосьминутная программа с ярким использованием экрана была бы весьма полезна.

Пусть величина x имеет n значений: x_1, x_2, ..., x_n; тогда вычисление среднего арифметического осуществляется по формуле

$$\bar{x}=\frac{1}{n}\sum_{i=1}^{n} x_i=\frac{x_1+x_2+\ldots+x_n}{n}.$$

Среднее квадратическое этих же чисел вычисляется по формуле

$$\bar{x}_q=\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}=\sqrt{\frac{x_1^2+x_2^2+\ldots+x_n^2}{n}}.$$

Аналогично вычисляется среднее кубическое:

$$\bar{x}_k=\sqrt[3]{\frac{1}{n}\sum_{i=1}^{n} x_i^3}=\sqrt[3]{\frac{x_1^3+x_2^3+\ldots+x_n^3}{n}}.$$

Важной в статистике является и среднее гармоническое:

$$\bar{x}_h=n:\sum_{i=1}^{n}\frac{1}{x_i}=n:\left(\frac{1}{x_1}+\frac{1}{x_2}+\ldots+\frac{1}{x_n}\right).$$

Формула общая, пригодная для вычисления всех средних, имеет вид

$$M=\left(\sum_{i=1}^{n} x_i^k/\,n\right)^{1/k}. \qquad (*)$$

Оказывается, что при $k=1$ ведется вычисление среднего арифметического, при $k=2$ и $k=3$ вычисляется среднее квадратическое и среднее кубическое. Наконец, при $k=-1$ по этой же формуле вычисляется среднее гармоническое.

Программа для вычисления по формуле (*) может быть такой:

```
5 '---------------------- ВЫЧИСЛЕНИЕ СРЕДНИХ ------------
10 INPUT "N="; N
15 DIM A(N)
20 '-------------- ВВОД ЧИСЛОВЫХ ДАННЫХ ----------------
25 FOR I=1 TO N
30 INPUN A(I)
35 NEXT I
40 '------------- ОПРЕДЕЛЕНИЕ ТИПА СРЕДНЕЙ -------------
45 INPUT "K="; K
50 S=0
55 FOR I=1 TO N
60 S=S+A(I)^K
65 NEXT I
70 '----------------- ВЫВОД РЕЗУЛЬТАТА -----------------
75 SR=(S/N)^(1/K) : PRINT SR
80 END
```

Учитель обращает внимание учащихся на то, что по одной общей программе вычисляется четыре типа средних. В основе программы лежит математическая формула. Программист лишь привлек на помощь ЭВМ. Это яркий пример того, как уже известные методы в математике начинают работать эффективнее, если используется ЭВМ. Естественно, что учитель в заключение поставит перед школьниками задачу и вычисления среднего геометрического по формуле

$$x_g = \sqrt[n]{x_1 \cdot x_2 \cdot \ldots \cdot x_n}\,.$$

Здесь же для наиболее любознательных школьников можно поставить исследовательскую задачу — составить программу для проверки истинности неравенства Коши:

$$\frac{x_1 + x_2 + \ldots + x_n}{n} \geqslant \sqrt{x_1 \cdot x_2 \cdot \ldots \cdot x_n}$$

или аналогичного неравенства

$$\frac{x_1^2 + x_2^2 + \ldots + x_n^2}{n} \leqslant \left(\frac{x_1 + x_2 + \ldots + x_n}{n}\right)^2 + \frac{x}{4},$$

где x — наибольшее из неотрицательных чисел: x_1, x_2, ..., x_n.

В качестве задания более подготовленным школьникам можно предложить составление программы для вычисления математического ожидания и дисперсии. Исходные данные представлены в виде таблицы.

x_i	x_1	x_2	...	x_n
p_i	p_1	p_2	...	p_n

Расчетные формулы:

$$M(x)=\sum_{i=1}^{n} x_i p_i;\ D(x)=M(x_i-M(x))^2.$$

Такое задание закрепит умение школьников образовывать суммы разнообразного характера. Умение программировать вычисления по формулам, аналогичным формуле для вычисления дисперсии, полезно и в общематематическом смысле: школьники научатся разворачивать компактно записанные формулы в суммы.

Возможный вариант программы предлагается ниже. Обращаем внимание на то, что таблица данных формируется вручную, значения переменной X и соответствующие вероятности P вводятся в режиме INPUT.

```
10 '---------.---- МАТЕМАТ. ОЖИДАНИЕ И ДИСПЕРСИЯ ---------
20 INPUT N
30 DIM X(N), P(N)
40 MX=0 : DX=0
50 PRINT "ВВОД ИСХОДНЫХ ДАННЫХ X(I) И P(I)"
60 FOR K=1 TO N
70      INPUT X(K), P(K)
80 NEXT K
90 FOR I=1 TO N
100     MX=MX+X(I)*P(I)
110 NEXT I
120 PRINT "МАТЕМ  ОЖИДАНИЕ MX="; MX
130 FOR I=1 TO N
140     DX= DX+(X(I)-MX)^2*P(I)
150 NEXT I
160 PRINT "ДИСПЕРСИЯ DX="; DX
170 END
```

5. Вычисление значений многочлена. Заметим, что задача вычисления значений многочлена обладает большой общностью. К вычислению значений многочлена сводится немало других задач, например задача замены чисел, данных в одной системе, на равные им числа из другой системы счисления. Вычисление значений ряда замечательных математических констант также требует умения вычислять значения многочлена:

$$a_n \cdot x^n + a_{n-1} \cdot x^{n-1} + \ldots + a_1 x + a.$$

Пример. Вычислить при $x=2$ значение многочлена

$$3x^5+2x^4-x^3-4x^2+x-1.$$

Перед вычислением многочлена его преобразуют по схеме Горнера, т. е. записывают в виде

$$(\ldots(a_n x + a_{n-1})\,x + a_{n-2})\,x + \ldots + a_1)\,x + a_0. \qquad (**)$$

Программирование вычислений сводится к программированию последовательности действий, заданных формулой (**). Отдельно следует обсудить способ представления многочлена в памяти ЭВМ. Это можно сделать, например, так:

с помощью оператора DATA ввести сначала показатель степени (n), значение x;

с помощью DATA ввести все $(n+1)$ коэффициенты начиная с a_n.

Схема программы может быть такой, какой изображена на рисунке 51. Текст программы приведен рядом со схемой.

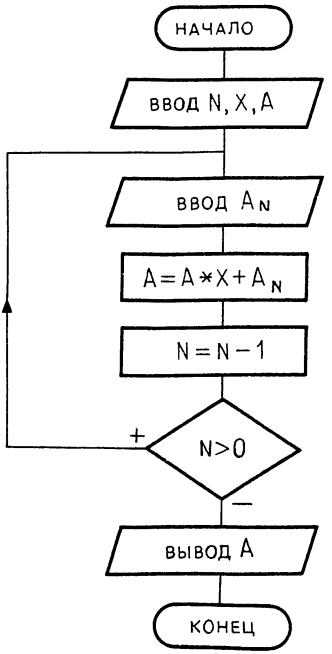


Рис. 51

```
5 DATA N, X, AN,..., AO
10 READ N, X
15 LET A=0
20 READ A: LET A=A*X+A
25 LET N=N-1
30 IF N>1 THEN 20
35 PRINT A
40 END
```

Ниже приводятся задания, которые расширят представления школьников о возможных применениях рассмотренного алгоритма и программы.

1. Составить программу для вычисления значений многочлена вида $a_n x^{2n} + a_{n-1} \cdot x^{2n-2} + \ldots + a_2 x^2 + a_0$, например $5x^8 + 3x^6 - 2x^4 + x^2 + 1$.

2. Составить программу для вычисления значений многочлена, в котором некоторые степени переменной x отсутствуют, например $5x^7 - 3x^5 + x^4 + 3x^3 - 2x + 1$.

3. Составить программу для вычисления значений многочлена вида $a_n x^{2n-1} + a_{n-1} x^{2n-3} + \ldots + a_1 x + a_0$.

4. Составить программу для вычисления значений многочлена вида $\frac{a_n}{x^n} + \frac{a_{n-1}}{x^{n-1}} + \ldots + \frac{a_1}{x} + a_0$.

5. Составить программу для вычисления значений многочлена вида

$$n! + (n-1)! + (n-2)! + \ldots + 2! + 1!$$

6. Данное число 235768_8 заменить равным ему десятичным числом.

7. Данные двоичные числа 1011101_2, 1101111_2, 1000111_2 заменить равными им десятичными числами.

8. Составить программу для вычисления значения многочлена вида

$$\frac{a_n}{n!}+\frac{a_{n-1}}{(n-1)!}+\frac{a_{n-2}}{(n-2)!}+\ldots+\frac{a_1}{1!}.$$

9. Составить программу для вычисления суммы

$$(2n-1)\ldots 3\cdot 1+\ldots+5\cdot 3\cdot 1+3\cdot 1+1.$$

10. Составить программу для вычисления суммы

$$(2n)\ldots 4\cdot 2+\ldots+6\cdot 4\cdot 2+4\cdot 2+2.$$

6. Решение уравнений. С этого урока учащиеся приступают к изучению методов программирования решения уравнений различного типа. Рассмотрим уравнения вида $F(x)=a$, для решения которых можно использовать для функции $F(x)$ вычисление значений обратной ей функции $\varphi(x)$ при $x=a$. Простейшими примерами таких уравнений являются:

$$\ln(x)=2;\ x^3=a;\ e^{x-3}=18{,}2;\ \ln(5x-2)=3{,}2.$$

Решать такие уравнения с помощью ЭВМ нетрудно, так как для основных элементарных функций в числе встроенных функций есть им обратные.

Учитель, начиная этот урок, должен проверить, насколько хорошо учащиеся разбираются в понятии «взаимно обратные функции». Учащиеся должны уверенно переходить от исходного уравнения $F(x)=a$ к выражению $x=\varphi(a)$.

Рассмотрим пример: пусть дано уравнение $e^{3x-1}=4$. В решении выделяем такие шаги:

определение обратной функции $\varphi(x)$;

получение выражения $x=\varphi(a)$;

составление программы для вычисления выражения $\varphi(a)$ и проверка найденного корня.

В данном случае имеем:

для функции $y=e^n$ обратной является функция $u=\ln y$; преобразуем данное уравнение. Логарифмируя, получаем $3x-1=\ln 4$. Отсюда находим $x=(\ln 4+1)/3$.

Программа может быть такой, какая приведена ниже. Конечно, программа охватывает не одно, а целый класс уравнений вида $e^{ax+b}=c$.

Текст программы:

```
10 '----------------- РЕШЕНИЕ УРАВНЕНИЙ -----------------
20 INPUT A, B, C
30 X=(LOG(C)-B)/A
40 PRINT "X="; X
50 '------------ ПРОВЕРКА РЕШЕНИЯ ---------------------
```

```
60 PRINT "C1="EXP(A*X+B)
70 IF ABS(C1-C)>0.00001 THEN 90
80 PRINT "ОТВЕТ X= "; X
90 PRINT "В РЕШЕНИИ ЕСТЬ ОШИБКА, ПОВТОРИТЕ РЕШЕНИЕ"
```

Из программы видно, что ее автор считает уравнение решенным верно, если вычисленное в ходе проверки значение отличается от заданного значения C не больше чем на 0,00001. Подчеркнем, что вычисления значений всех функций — и прямых, и им обратных — осуществляется приближенно и поэтому сопоставление результатов при решении уравнений необходимо. Учащиеся должны знать, что проверка входит в процедуру решения и что программа решения уравнения должна включать проверку результатов.

Изучение предложенного метода решения уравнений расширяет кругозор школьников, знакомит их с самыми разнообразными уравнениями. При этом ученики не боятся, что они их «решать не умеют». При решении уравнений изложенным выше методом они прикидывают, какие взаимно обратные функции можно применить. Основная трудность при решении уравнений этим методом заключена в правильном выборе обратной функции и составление выражения, которое предстоит вычислять ЭВМ.

Примером сложного уравнения может быть такое: $\operatorname{tg}(\ln x)=c$. Решая его, получим $\ln x=\operatorname{arctg}(c)$, откуда $x=e^{\operatorname{arctg}(c)}$.

Расширяя представления учащихся об уравнениях, учитель показывает, какие практически важные задачи приводят к изучаемым уравнениям. Таким примером может быть задача: через сколько лет вклад в N рублей при начислении P рублей процента годовых удвоится? Ответ сводится к вычислению выражения

$$x=\ln(1+p/100)/\ln 2.$$

Данный урок может пройти как урок по решению уравнений за пультом ЭВМ. Дело в том, что все программы однотипны. Главное — верно выписать формулу для вычисления искомой величины.

В качестве уравнений, которые учитель может предложить для решения на уроке, могут быть такие:

$\sin(ax+b)=c$	$\sin(2x-3)=0{,}6$,
	$\sin(2-x)=0{,}4$,
	$\sin(5-3x)=-0{,}45$;
$\cos(ax+b)=c$	$\cos(2x+1)=-0{,}4$,
	$\cos(5-3x)=0{,}2$,
	$\cos(3x-2)=0{,}8$;
$\ln(ax+b)=c$	$\ln(5x-6)=2$;
$e^{ax+b}=c$	$e^{3x+5}=11{,}2$,
	$e^{5-x}=0{,}65$,
	$e^{2x-1}=2{,}9$.

При решении уравнений в программу следует вписать все проверки на допустимость вычислений. Если необходимо вычислить $\ln(ax+b)$, то в программе должен быть оператор IF $ax+b>0$, и т. д.

7. Решение уравнений (продолжение). На решение уравнений можно отвести еще два урока. Первый из них следует посвятить алгоритмическим вопросам решения уравнений, в частности хорошо разобраться в логике алгоритма решения уравнения методом половинного деления отрезка, содержащего единственный корень (рис. 52).

С помощью рисунка учащимся полезно напомнить, что для монотонной функции, имеющей на отрезке $[a; b]$ единственный корень, справедливо утверждение $f(a)\cdot f(b)<0$ (рис. 53).

Особого внимания учителя требует рассмотрение части схемы алгоритма, изображенной на рисунке.

Учащиеся должны понимать суть таких операторов, как LETB=X, LETA=X и LET Y1=Y, и уметь иллюстрировать их. На рисунке 53,*а* имеет место случай, когда $y_1\cdot y_0<0$, и поэтому точка *b* сдвигается влево, в точку *x*. На рисунке 53, *б* имеет место случай, когда $y_1\cdot y_0>0$, и поэтому точка *a* сдвигается вправо.

Завершая этот урок, целиком посвященный анализу особенностей рассмотренного выше алгоритма, учитель может сообщить учащимся, что существуют и другие методы решения уравнений, которые хорошо реализуются на ЭВМ.

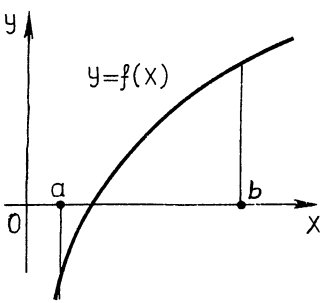


Рис. 52

На втором уроке по этой теме в центре внимания стоят вопросы программирования, а также перехода от схемы алгоритма к программе. Так как к этому времени учащиеся овладели операторами LET и IF, то текст программы они могут написать самостоятельно.

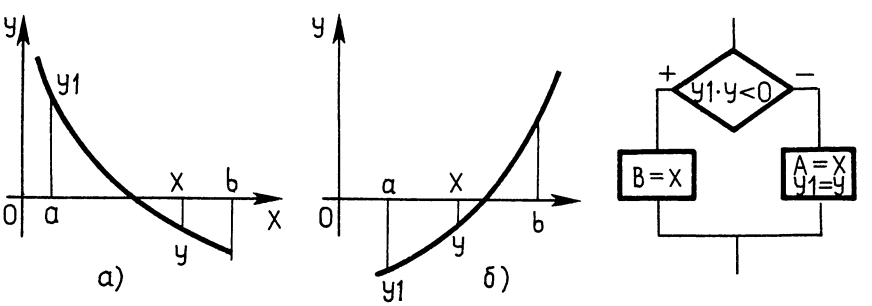


Рис. 53

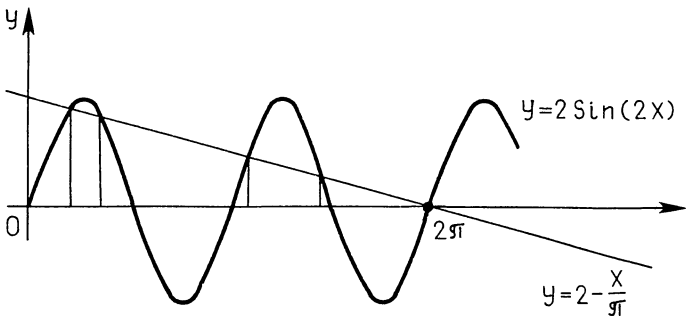


Рис. 54

Ученик на уроке по общей программе решает уравнение и продумывает систему машинной проверки найденного решения.

Опыт показывает, что учитель на этом же уроке успевает рассмотреть задачу и об отделении корней и показать всем школьникам пример, привлекая для демонстрации индивидуальные дисплеи. Учителю желательно иметь плакат со схемой алгоритма.

Еще раз подчеркнем, что если уравнение на некотором отрезке имеет не один, а несколько корней, то в этом случае возникает задача разделить бóльший отрезок на меньшие так, чтобы каждый маленький из отрезков содержал бы уже только один корень.

Один из способов отделения корней продемонстрируем на примере: пусть дано уравнение $2\sin(2x)+\frac{x}{\pi}-2=0$.

Попробуем решить его графически, для чего построим графики функций $y_1=2\sin(2x)$ и $y_2=2-\frac{x}{\pi}$ (рис. 54).

Из рисунка видно, что часть корней данного уравнения принадлежит отрезку $[0;\ 2\pi]$. Выделение из этого отрезка отрезков, содержащих уже по одному корню, основано на следующей идее: исходный отрезок $[0;\ 2\pi]$ разбивается на k маленьких отрезков длиной H, где $H=2\pi/k$. На каждом из маленьких отрезков проверяется, содержит ли этот отрезок корень, и если содержит, то координаты его концов выводятся на экран (на печать).

Предварительно построенные графики позволяют сделать предварительную оценку для k. Все детали алгоритма, основанного на такой идее, можно разобрать по схеме алгоритма, изображенной на рисунке 55.

Учитель может предложить учащимся для решения методом деления отрезка пополам следующие уравнения:

1. $x+\sqrt{x}+\sqrt[3]{x}-2{,}5=0$
2. $\sqrt{1-x}-\operatorname{tg}(x)=0$ $[0;\ 1]$.
3. $e^x+\sqrt{1+e^{2x}}-2=0$ $[-1;\ 0]$.
4. $\sin(\ln x)-\cos(\ln x)+2\ln x=0$ $[1;\ 3]$.
5. $\cos(x)-e^{-\frac{x^2}{2}}+x-1=0$ $[1;\ 2]$.

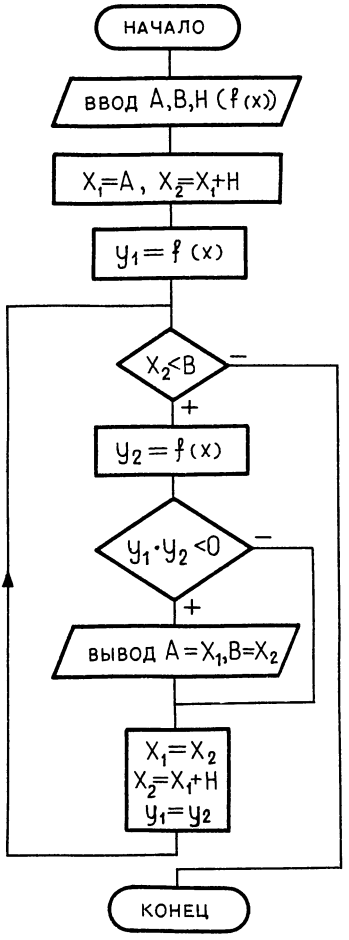


Рис. 55

8. Решение неравенств на ЭВМ. Каким образом поручить ЭВМ решение неравенств? Как следует разрабатывать алгоритм для решения неравенств вида $f(x)>0$, где $f(x)$ — функция достаточно сложная? Следует ли предварительно заняться классификацией неравенств и сначала, например, научится решать линейные неравенства, а затем квадратные, кубические и т. д., все усложняя и усложняя задачу?

Наводящую идею можно почерпнуть, если обратиться к ранее рассмотренному алгоритму отделения корней. Действительно, если для $f(x)$ корни удалось отделить, то это сразу же даст возможность оценить знак функции в каждом промежутке между корнями.

На рисунке 56 схематически показано, что для уравнения $f(x)=0$ удалось найти четыре промежутка, содержащих по одному корню: $[a_1; b_1]$, $[a_2; b_2]$, $[a_3; b_3]$ и $[a_4; b_4]$. Известно, что поиск промежутков осуществляется на заданном отрезке $[a; b]$.

По существу, найдено пять промежутков, уже не содержащих корня, и на каждом из них функция $f(x)$ сохраняет знак. Решение неравенства $f(x)>0$ сводится теперь к определению знака в каждом из полученных пяти промежутков и фик-

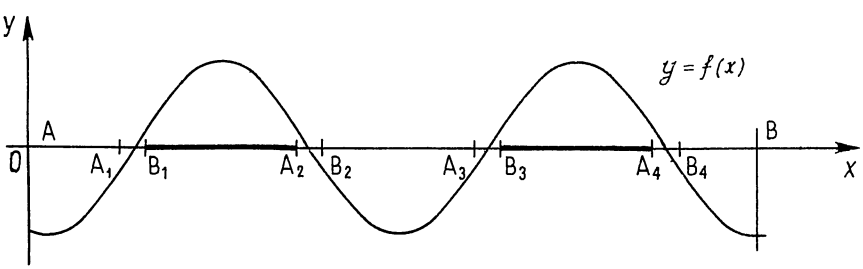


Рис 56

сации (вывод на печать) тех из них, для которых имеет место условие: $f(x)>0$.

Соответствующая программа может быть такой, как приводимая ниже.

```
10 '------------- РЕШЕНИЕ НЕРАВЕНСТВ F(X)>0 -------------
20 CLS: PRINT "В СТРОКУ 60 ВВЕСТИ F(X)"
30 PRINT "ПОСЛЕ ВВОДА - RUN 50"
40 PRINT "В РЕШЕНИИ ЕСТЬ ОШИБКА, ПОВТОРИТЕ РЕШЕНИЕ"
50 LIST 60
60 INPUT "ЗАДАННЫЙ ОТРЕЗОК A, B" A, B
70 INPUT "ТОЧНОСТЬ ВЫЧИСЛЕНИЙ H"; H
80 DEF FNX(X)=
90 FOR X=A TO B STEP H
100 IF FNX(X)*FNX(X+H)>0 THEN 130
110 S=SGN FNX(XL)): XP=X
120 IF S=1 THEN PRINT XL, XP
130 XL=X+H
140 NEXT X: PRINT XL, B
150 PRINT "ПОИСК ОКОНЧЕН": END
```

В качестве примера применения программы рассмотрим решение неравенства

$$x^5-3x^4-5x^3+12x^2+4x-12>0.$$

Исследование будет осуществляться на отрезке $[-12; 12]$, точность при вычислении координат концов отрезка положим равной 0,1. При желании и после того, как будет ясно, что отрезок исследования $[-12; 12]$ задан удачно, точность можно увеличить, а отрезок исследования сузить. Приведем полученные результаты.

```
ЗАДАННЫЙ ОТРЕЗОК A, B       -12, 12
ТОЧНОСТЬ ВЫЧИСЛЕНИЙ H        0.1
-1.99 999                   -1.09999
 1.00001                     1.90001
 3.000009                    12
ПОИСК ОКОНЧЕН
```

Найдено три отрезка, на которых $f(x)>0$. Это отрезки $[-1{,}99; -1{,}09]$, $[1{,}00; 1{,}9]$ и $[3{,}00; 12]$.

В том, что ответ правильный, можно убедиться, разложив на множители $f(x)$. Имеем:

$$x^5-3x^4-5x^3+12x^2+4x-12=(x-1)(x-3)(x+1)(x-2)(x+2).$$

Учитель может предложить школьникам решить следующие неравенства:

1. $(x^2-9)(x^4-5x^2+4)>0$ $[-4; 4]$.
2. $x^4-11x^3+41x^2-61x+30>0$ $[-4; 6]$.

3. $\frac{(x-3)(x^2+4)}{(x+1)(x-5)} > 0$ $[-4; 6]$.

4. $1-e^{x-2} > 0$ $[-1; 3]$.

5. $\sin(3x-1) > 0$ $[-1; 5]$.

Существенным недостатком рассмотренного подхода является необходимость предварительного определения промежутка $[a; b]$, на котором содержатся все корни функции $f(x)$.

9. Системы линейных уравнений. В число задач, которые следует обязательно проработать на уроках информатики, входит задача программирования решения на ЭВМ систем линейных уравнений. Система линейных уравнений является математической моделью многих прикладных задач, и умение грамотно работать с такой моделью необходимо.

Целесообразно начать с решения системы двух линейных уравнений с двумя неизвестными и вместе с учащимися разобрать использование формул Крамера, введя понятие об определителе второго порядка. В последнее время этот метод не включается в число обязательных для изучения на уроках математики.

Систему можно записывать в виде, удобном для дальнейшего программирования:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 = a_{13}; \\ a_{21}x_1 + a_{22}x_2 = a_{23}. \end{cases}$$

Преподаватель вводит понятия об определениях системы и двух вспомогательных определителях. Сделать это целесообразно, рассмотрев решение данной системы методом алгебраического сложения.

Приводим соответствующие выкладки:

$$\begin{array}{l|l} a_{11}x_1 + a_{12}x_2 = a_{13} & a_{22} \text{ (умножаем обе части обоих} \\ a_{21}x_1 + a_{22}x_2 = a_{23} & a_{12} \text{ уравнений)} \end{array}$$

$$\begin{array}{l} a_{11}a_{22}x_1 + a_{12}a_{22}x_2 = a_{13}a_{22} \\ \underline{a_{21}a_{12}x_1 + a_{22}a_{12}x_2 = a_{23}a_{12}} \\ (a_{11}a_{22} - a_{21}a_{12})\,x_1 = a_{13}a_{22} - a_{23}a_{12} \end{array}$$

(вычитаем одно уравнение из другого)

Аналогично получаем

$$(a_{12}a_{22} - a_{21}a_{12})\,x_2 = a_{13}a_{21} - a_{23}a_{11}.$$

Вводим обозначения:

$$D = a_{11}a_{22} - a_{21}a_{12};$$
$$D_{x_1} = a_{22}a_{13} - a_{23}a_{12};$$
$$D_{x_2} = a_{13}a_{21} - a_{23}a_{11}.$$

Отсюда получаем формулы Крамера для вычисления x_1 и x_2:

$$x_1 = \frac{D_{x_1}}{D}; \quad x_2 = \frac{D_{x_2}}{D}.$$

После таких разъяснений составляется схема алгоритма решения системы двух линейных уравнений с двумя неизвестными. При этом обращается внимание на роль определителя системы D; подчеркивается, что если $D=0$, то система единственного решения не имеет.

Разработка программы может стать темой практического занятия для всего класса в целом (схема алгоритма должна быть у каждого учащегося или в виде плаката на стене).

Приводим возможный вариант текста программы:

```
10 '--------------- РЕШЕНИЕ СИСТЕМ УРАВНЕНИЙ ------------
20 DIM A(2, 3)
30 FOR I=1 TO 2
40 FOR K=1 TO 3
50 INPUT A(I, K)
60 NEXT K
70 NEXT I
80 D=A(1, 1)*A(2, 2)-A(2, 1)*A(1, 2)
90 IF D=0 THEN PRINT "ЕДИНСТВЕННОГО РЕШЕНИЯ НЕТ": END
100 DX=A(1, 3)*A(2, 2)-A(2, 3)*A(1, 2)
110 DY=A(1, 1)*A(2, 3)-A(2, 1)*A(1, 3)
120 X=DX/D: Y=DY/D
130 PRINT X, Y
140 END
```

Для закрепления темы рекомендуется приготовить карточки-задания. В карточках-заданиях можно затронуть проблему значительного влияния на решение точности вводимых коэффициентов, особенно если определитель близок к нулю.

Школьникам следует пояснить, что в этом случае небольшое изменение коэффициентов может значительно изменить величину искомых корней. Полезно все рассуждения иллюстрировать геометрически.

На уроках информатики можно остановиться на методе Гаусса решения систем линейных уравнений. Методы подстановки и алгебраического сложения являются основными, которые рассматриваются на уроках математики.

Ниже приводится возможный вариант текста программы. Учитель должен сам решить, в какой мере он будет его анализировать. Можно ограничиться демонстрацией решения конкретной системы из трех или четырех уравнений. Можно подробно рассмотреть каждую строку программы и главным считать технику подготовки таких программ. В этом случае учителю вновь рекомендуется разработать схему алгоритма.

Текст программы, приводимый ниже, может быть использован для самостоятельной отработки; от учащегося в этом случае требуется дать письменные разъяснения по тексту программы, написать развернутый комментарий.

```
10 '--------------- ГАУСС(ПРОСТЕЙШИЙ АЛГОРИТМ) ---------
20 INPUT "N="; N: DIM A(N,N), D(N), B(N), X(N)
30 '----------------- ВВОД ----------------------------
40 FOR I=1 TO N:FOR J=1 TO N
50 PRINT "A("; I;","; J;")=";
60 INPUT A(I,J); NEXT J
70 PRINT "B("; I;")= ";: INPUT B(I): NEXT I
80 '--------------- ИСКЛЮЧЕНИЕ НЕИЗВЕСТНЫХ ------------
90 FOR K=1 TO N-1
100 FOR I=K+1 TO N
110 M=A(I, K)/A(K, K)
120 FOR J=K+1 TO N
130 A(I, J)=A(I, J)-M*A(K, J)
140 NEXT J
150 B(I)=B(I)-M*B(K)
160 NEXT I
170 NEXT K
180 '-------------- ОБРАТНАЯ ПОДСТАНОВКА ---------------
190 X(N)=B(N)/A(N, N)
200 FOR I=N-1 TO 1 STEP -1: S=0
210 FOR J=I+1 TO N: S=S+A(I, J)*X(J): NEXT J
220 X(I)=(B(I)-S)/A(I, I): NEXT I
230 FOR I=1 TO N: PRINT "X("; I;")="; X(I): NEXT I: END
```

Для примера можно рассмотреть решение методом Гаусса ниже приводимых систем:

а) $\begin{cases} 2{,}1x_1+3{,}4x_2=-4{,}522; \\ 3{,}5x_1-2{,}3x_2=9{,}751; \end{cases}$ б) $\begin{cases} 2{,}73x_1-6{,}2x_2+1{,}7x_3=29{,}1; \\ 2{,}56x_1+3{,}2x_2-2{,}5x_3=3{,}805; \\ 1{,}79x_1-2{,}59x_2+6{,}54x_3=24{,}049. \end{cases}$

§ 41. ГЕОМЕТРИЯ И ГРАФИКА НА ЭВМ

Использование ЭВМ в работе с геометрическими и графическими объектами можно осуществить в различных формах.

Работа с геометрическими объектами требует хороших представлений учащихся о методе координат. В частности, учитель может опираться на то, что школьники знают, как вычисляется расстояние между двумя точками, заданными своими координатами; знают уравнение прямой, окружности; понимают и умеют находить координаты точек пересечения линий на плоскости. Конечно, учитель может опираться и на умение школьников схематически изображать графики элементарных функций.

Среди средств Бейсика MSX имеются операторы, предназначенные для использования в рамках метода координат. Охарактеризуем основные из них:

PSET (X, Y), C — размещение окрашенной точки на экране в точке с координатами (X, Y); $0 \leqslant C \leqslant 15$ — цвет точки.

LINE (X1, Y1) — (X2, Y2), C — проведение отрезка цвета *C* между точками (X1, Y1) и (X2, Y2).

LINE (X1, Y1) — (X2, Y2), C, B — вычерчивание прямоугольной рамки; (X1, Y1) и (X2, Y2) — противоположные вершины одной диагонали рамки, C — цвет рамки.

LINE (X1, Y1) — (X2, Y2), C, BF — вычерчивание закрашенного прямоугольника; (X1, Y1) и (X2, Y2) — противоположные вершины одной диагонали прямоугольника; C — цвет закраски.

CIRCLE (X, Y), R, C — вычерчивание окружности с центром в точке (X, Y) радиуса R; C — цвет окружности.

CIRCLE (X, Y), R, C, A1, A2 — вычерчивание дуги окружности с центром в точке (X, Y) радиуса R; C — цвет линии дуги; A1 — угол (начало дуги); A2 — угол (конец дуги).

CIRCLE (X, Y), R, C,,, O — вычерчивание эллипса; здесь O — отношение осей.

Используя эти средства, учитель может показать, как ЭВМ может выполнять геометрические преобразования плоскости: параллельный перенос, осевую симметрию, центральную симметрию, поворот вокруг точки и гомотетию.

Разработку программ реализации каждого из указанных геометрических преобразований учитель может взять как тему одного из уроков или рассматривать как задание для самостоятельной работы.

При решении поставленных задач и других задач геометрии учитель имеет возможность еще раз привлечь внимание школьников к использованию моделей, к моделированию как методу решения задач. В этом случае решение геометрических задач переносится в область алгебры: для каждой такой задачи создается ее алгебраическая модель, которая и обрабатывается на ЭВМ.

Рассмотренный подход один из основных, ниже дается два примера использования такого подхода. Однако при желании учитель может вооружиться «циркулем» и «линейкой» (с делениями или без делений). Речь идет о том, что в роли линейки выступит отрезок, координатами концов которого можно управлять с кла-

виатуры, давая приращения каждой из координат левого и правого концов отрезка. Это значит, что «линейка» сможет перемещаться параллельно самой себе, скользить вдоль избранной прямой, поворачиваться вокруг одного из концов и многое другое. Располагать «циркулем» — это значить уметь быстро с клавиатуры указать центр окружности и ее радиус, провести эту окружность.

Манипулируя выделенными для этого кнопками, учитель сможет вести традиционные построения на экране быстрее, чем на доске, и использовать при этом различные цвета.

Умение моделировать на ЭВМ традиционными средствами для построения геометрических фигур (циркулем и линейкой) подсказывает методическую идею вооружения пользователя ЭВМ необычными геометрическими инструментами. Можно представить себе учителя, ведущего урок геометрии и использующего «эллипсопостроитель» или «параболопостроитель». В этом случае количество решаемых задач значительно расширится, особенно задач на построение.

Приведем примеры: пусть необходимо построить треугольник, сумма боковых сторон которого равна S длина основания c и длина медианы, проведенной к стороне, m_c.

Напомним, что эллипс есть множество точек плоскости, сумма расстояний которых до двух фокусов есть величина постоянная. Учитывая это определение, построение можно осуществить по плану:

1. Построить эллипс: $\frac{x^2}{\left(\frac{S}{2}\right)^2}+\frac{y^2}{\left(\frac{S}{2}\right)^2-\left(\frac{c}{2}\right)^2}=1$.
2. Построить окружность: $x^2+y^2=m_c^2$.
3. Найти точки пересечения окружности и эллипса, соединить фокусы эллипса с одной из найденных точек. Полученный треугольник искомый.

Естественно, что все элементы анализа и решения задач на построение, которые присутствуют при построениях с помощью циркуля и линейки, имеют место и при построении фигур с применением других инструментов. Еще раз подчеркнем, что моделирование новых инструментов с помощью ЭВМ создает основу для необычайно сильных и эффективных приемов в преподавании геометрии.

Так, при желании учитель может показать учащимся решение одной из знаменитых задач геометрии — задачи об удвоении куба. Эту задачу решить с помощью только линейки и циркуля невозможно. Однако задача очень просто и красиво решается, если использовать построение параболы.

З а д а ч а. Дан куб. Построить новый куб, объем которого в два раза больше исходного.

Решить задачу — это значит по данной стороне s исходного куба построить сторону x искомого куба.

Р е ш е н и е. Объем искомого куба x^3, объем данного куба s^3. Имеем $x^3=2\cdot s^3$.

Отсюда получаем $x=s\cdot\sqrt[3]{2}$; следовательно, искомая сторона равна данной s, умноженной на $\sqrt[3]{2}$.

Рассмотрим две параболы: $x^2=ay$ и $y^2=bx$. Координаты a и b подберем, учитывая, что

$$y=\frac{x^2}{a} \text{ и } \frac{x^4}{a^2}=bx;\ x^3=a^2b,\ x=\sqrt[3]{a^2b}.$$

Возьмем $s=1$, $a=1$ и $b=2$, и тогда искомая сторона $x=\sqrt[3]{2}$.

Для построения x используем параболы $y^2=2x$ и $x^2=y$. Найдем абсциссу их точек пересечения. Рисунок 57 поясняет решение.

Упомянем еще и о дополнительных возможностях работы с изображенными на экране объектами, но которые не принято относить к подлинно геометрическим. Речь идет о мозаиках, паркетах, узорах, как содержащих правильные геометрические многоугольники, так и не содержащих таковых, об орнаментах.

Пользователь имеет возможность создавать специальные, им самим формируемые фигуры (спрайты); полученные фигуры можно очень просто перемещать по поверхности экрана, окрашивать в различные цвета, изменять размеры. Такие возможности не являются обязательными для овладения всеми учащимися — учителю же ознакомиться с ними необходимо. Овладев приемами машинной графики, учитель может создать немало полезных наглядных пособий по многим учебным дисциплинам.

1. Метод координат — реализация на ЭВМ. Все особенности использования метода координат разъясним на примере решения конкретных задач.

З а д а ч а 1. На координатной плоскости проведена прямая $Ax+By+C=0$. На этой же плоскости даны две точки: $M(x_1, y_1)$ и $N(x_2, y_2)$. Составить программу, работая по которой ЭВМ выяснит, лежат ли точки M и N по одну сторону от данной прямой.

Задача является геометрической, но решение ее будет осуществляться средствами алгебры. Следовательно, необходимо создать алгебраическую модель, работая с которой можно получить ответ на вопрос задачи. Иначе говоря, какие действия с уравнением прямой и координатами точек M и N следует осуществить, чтобы узнать, лежат ли точки M и N по одну сторону от прямой $Ax+By+C=0$.

Рассуждения могут быть такими: известно, что если координаты точки удовлетворяют уравнению прямой, то точка лежит на прямой; иначе: точка лежит по какую-то сторону от этой прямой в одной из полуплоскостей, на которую прямая делит плоскость. Ясно, что точки, лежащие в одной и той же полуплоскости, при подстановке их координат в уравнение прямой дают числовые значения многочлена $Ax+By+C$ одного и того же знака. Иначе говоря, если $(Ax_1+By_1+C)(Ax_2+By_2+C)>0$, то обе точки: $M(x_1, y_1)$ и $N(x_2, y_2)$ — лежат по какую-то одну сторону от прямой $Ax+By+C=0$.

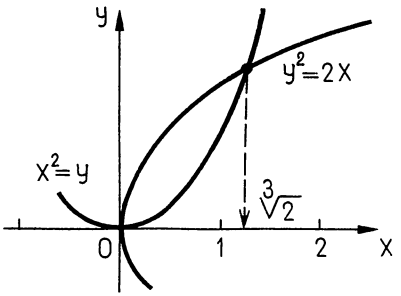


Рис. 57

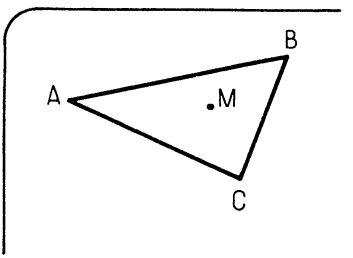


Рис. 58

Алгебраическая модель построена — это неравенство

$$(Ax_1+By_1+C)(Ax_2+By_2+C)>0.$$

«Заглядывая» в неравенство, ЭВМ «увидит», как расположены точки.

Программа, соответствующая такой модели, может быть такой:

```
5 DATA
10 READ A, B, C, X1, Y1
15 LET Z1=A*X1+B*Y1+C
20 READ X2, Y2
25 LET Z2=A*X2+B*Y2+C
30 IF Z1*Z2>0 THEN 45
35 PRINT "ТОЧКИ ЛЕЖАТ В РАЗНЫХ ПОЛУПЛОСКОСТЯХ'
40 GOTO 50
45 PRINT "ТОЧКИ ЛЕЖАТ В ОДНОЙ ПОЛУПЛОСКОСТИ"
50 END
```

Рассмотренную задачу учитель может считать подготовительной для решения более сложной задачи, предлагаемой для самостоятельного решения.

З а д а ч а 2. На координатной плоскости расположен треугольник, уравнения сторон которого известны:

$$AB\colon\ A_1x+B_1y+C_1=0;\ BC\colon\ A_2x+B_2y+C_2=0;$$
$$AC\colon\ A_3x+B_3y+C_3=0.$$

На плоскости дана точка $M(x_4, y_4)$. Выяснить, лежит ли точка M внутри треугольника ABC.

Построение алгебраической модели можно базировать на следующих соображениях: если точка M лежит внутри треугольника ABC, то она находится в одной полуплоскости с вершиной C, если сторону AB считать прямой, образующей две полуплоскости. Аналогично: если BC — линия, делящая плоскость на полуплоскости, то точка M находится в одной полуплоскости с вершиной A. И наконец, точка M находится в одной полуплоскости с вершиной B относительно прямой AC. Рисунок 58 поясняет эти соображения.

Алгебраической моделью задачи являются три последовательно рассматриваемых соотношения:

$$(A_1x_3+B_1y_3+C_1)(A_1x_4+B_1y_4+C_1)>0;$$
$$(A_2x_1+B_2y_1+C_2)(A_2x_4+B_2y_4+C_2)>0;$$
$$(A_3x_2+B_3y_2+C_3)(A_3x_4+B_3y_4+C_3)>0.$$

Если хотя бы одно из соотношений окажется ложным, то проверку можно прекратить и ЭВМ может выдать сообщение: «Точка M не лежит внутри треугольника ABC».

Составление программы для решения такой задачи — это небольшой экзамен по технике программирования. Учащиеся имеют возможность проявить фантазию.

В качестве еще одного примера решения геометрической задачи с помощью ЭВМ рассмотрим задачу о симметрии точек относительно прямой. Остановимся на простейшей задаче — построении точки A_1, симметричной данной точке A, относительно прямой $Ax+By+C=0$.

Предлагается следующий подход к решению задачи и разработке программы для ЭВМ. Прежде всего после ввода исходных данных: координат данной точки (x_A, y_A), коэффициентов уравнения оси симметрии (A, B и C)— на экран ЭВМ выводятся изображения точки A (в виде небольшого кружочка) и отрезка MN оси симметрии.

Затем ЭВМ вычисляет координаты искомой точки A_1 и вновь обращается к подпрограмме вывода на экран изображений точек A, A_1 и оси симметрии.

Следует учесть, что если в уравнении прямой $Ax+By+C=0$ коэффициент $B=0$, то ось симметрии вертикальна и искомые координаты точки A_1 находятся по формулам $y_{A_1}=y_A$, а $x_{A_1}=2\left(-\frac{C}{A}\right)-x_A$. В программе такое расположение оси симметрии учтено.

Следует помнить, что не всякая прямая проходит через первую четверть координатной плоскости и не любая точка лежит в этой части плоскости. Задача как демонстрационная имеет ограничения, связанные с размером экрана.

В предлагаемой программе в строках 60—120 реализуется вывод отрезка MN оси симметрии, который может быть изображен на экране:

```
10 '-------------- СИММЕТРИЯ ОТНОСИТЕЛЬНО ПРЯМОЙ --------
20 SCREEN 0, 0, 0: CLS
30 PRINT "ВВОД ДАННЫХ XA, YA, A, B, C"
40 INPUT XA, YA, A, B, C
50 XA1=XA: YA1=YA
60 FOR I=1 TO 200 STEP 5
70 Y=(-A/B)*I-C/B
80 IF Y>0 THEN XN=I: YN=Y: GOTO 100
```

```
90 NEXT I
100 FOR I=5 TO 1 STEP -1
110 XM=XN+I*100:Y=(-A/B)*XM-C/B
120 IF (Y<0 OR Y>400) THEN NEXT I
130 YM=Y
140 GOSUB 230
150 XO=(B^2*XA-A*(C+YA*B))/(A^2+B^2)
160 YO=(A^2*YA-B*(A*XA+C))/(A^2+B^2)
170 '--------------- ВЫВОД ИСКОМЫХ КООРДИНАТ ------------
180 XA1=2*XO-XA:YA1=2*YO-YA
190 PRINT "XA1="; XA1,  "YA1="; YA1
200 FOR I=1 TO 5000 : NEXT I
210 COSUB 230
220 END
230 '---------------- ПОДПРОГРАММА ----------------------
240 CLS: SCREEN 3
250 CIRCLE (XA, YA), 1, 3
260 CIRCLE (XA1, YA1)M 1, 3
270 LINE (XM, YM)-(XN, YN), 3
280 LINE (XA, YA)-(XA1, YA1), 5
290 FOR I=1 TO 15000 : NEXT I
300 CLS: RETURN
```

Изображение на экране появляется два раза. Один раз демонстрируются исходная точка A, отрезок оси симметрии MN. Второе изображение включает дополнительно искомую точку A_1 и отрезок, соединяющий симметричные точки.

1. Составить программу для отыскания координат вершин треугольника ABC по данному основанию b, высоте h и произведению двух боковых сторон: $a \cdot c = m$. Рекомендация. Ось x совмещать со стороной AC, ось y провести через середину этой же стороны.

2. Составить программу, работая по которой ЭВМ ответит на вопрос: в какой четверти координатной плоскости раеположена точка $A(x, y)$?

3. Составить программу, работая по которой ЭВМ выяснит, лежит ли точка $A(x, y)$ выше или ниже прямой $y = ax + b$. Ответ дать в развернутой форме, например: точка A лежит ниже прямой $ax + b$.

4. Составить программу, работая по которой ЭВМ сможет ответить на вопрос: лежит ли точка A внутри острого угла, образованного прямыми $y = a_1x + b_1$ и $y = a_2x + b_2$? (Рис. 59.)

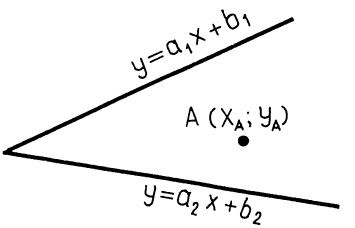


Рис. 59

5. Составить программу, работая по которой ЭВМ ответит

на вопрос: лежит ли точка A внутри окружности, заданной уравнением $x^2+y^2=R^2$?

6. Составить программу, работая по которой ЭВМ ответит на вопрос: лежит ли точка A в квадрате со стороной a, центр которого расположен в точке $O(x, y)$?

7. Составить программу, работая по которой ЭВМ ответит на вопрос: лежит ли точка $M(x, y)$ внутри треугольника, заданного координатами вершин: $A(x_A, y_A)$; $B(x_B, y_B)$; $C(x_C, y_C)$?

8. Составить программу, работая по которой ЭВМ ответит на вопрос: лежит ли точка M внутри треугольника, заданного уравнениями сторон $AB: A_1x+B_1$; $BC: A_2x+B_2$; $AC: A_3x+B_3$?

9. Составить программу, работая по которой ЭВМ ответит на вопрос: является ли данный многоугольник правильным?

10. Составить программу для выяснения: является ли данный многоугольник выпуклым?

Решение последней задачи поясним. При ее решении потребуется знание уравнения прямой, проходящей через две точки. Решение задачи можно начать с рассмотрения чертежа, на котором изображены два пятиугольника: один из них выпуклый, другой невыпуклый.

По определению все вершины выпуклого многоугольника лежат по одну сторону от любой прямой, содержащей одну из сторон данного многоугольника. Отсюда следует идея алгоритма: рассматривать одну за другой все n таких прямых и проверять, лежат ли $n-2$ вершины по одну сторону от проверяемых прямых. В рассмотрение не включаются две вершины, через которые проводится очередная прямая.

О том, как это делается, было рассказано выше. Таким образом, в задаче используется уже рассмотренный алгоритм.

Подчеркнем некоторые особенности программы. Точки рассмариваются парами, и при этом исключаются точки, через которые проходит прямая, относительно которой ведется проверка взаимного расположения точек. Следует следить и за тем, чтобы какая-либо из проверяемых двух точек не попала на прямую в качестве второй из проверяемых. Все эти проблемы решены в строках программы 130—146. Точки лежат по разные стороны от проверяемой прямой, если условие (146) выполняется.

Завершив анализ расположения точек относительно всех прямых, ЭВМ выводит сообщение: «Многоугольник выпуклый» или «Многоугольник невыпуклый» — и прерывает работу. Возникает пауза — и учитель может вызвать на экран изображение данного многоугольника.

Для вызова изображения или для его снятия с экрана достаточно нажать на клавишу «Пробел».

```
10 '--------------- ВЫПУКЛЫЕ МНОГОУГОЛЬНИКИ -------------
20 SCREEN 0, 0, 0: CLS
30 INPUT "ВВЕДИТЕ КОЛИЧЕСТВО ВЕРШИН N"; N
```

```
40 DIM X(N+1), Y(N+1), Z(N), A(N), B(N), C(N), R(N)
50 FOR I=1 TO N
60 PRINT "ВВЕДИТЕ КООРДИНАТЫ"; I; "ВЕРШИН"
70 INPUT X(I), Y(I)
80 NEXT I
90 X(N+1)=X(1): Y(N+1)=Y(1)
100 '-------- ПРОВЕРКА УСЛОВИЙ ВЫПУКЛОСТИ ---------------
110 FOR I=1 TO N
120 A(I)=Y(I+1)-Y(I): B(I)=X(I+1)-X(I): C(I)=B(I)*Y(I)-
    A(I)*X(I)
130 FOR J=1 TO N-1
135 IF (J=I OR J=I+1) THEN 150
136 IF I=N THEN 142
140 R(J)=SGN(A(I)*X(J)-B(I)*Y(J)+C(I))
142 IF J+1=I THEN GOTO 150
144 R(J+1)=SGN(A(I)*X(J+1)-B(I)*Y(J+1)+C(I))
146 IF R(J) <> R(J+1) THEN 230
150 NEXT J
210 NEXT I
220 PRINT "МНОГОУГОЛЬНИК ВЫПУКЛЫЙ": GOTO 240
230 PRINT МН-К НЕВЫПУКЛЫЙ"
240 F$=INPUT $(1) 'ПАУЗА, ДЛЯ СНЯТИЯ НАЖАТЬ ПРОБЕЛ
250 SCREEN 3:CLS
260 PSET (X(1), Y(1)), 3
270 FOR F=2 TO N+1
280      LINE -(X(F), Y(F)), 3
290      'LINE(X(1), Y(1))-(X(F), Y(F)), 3
300 NEXT F
310 F$=INPUT $(1) 'ПАУЗА, ДЛЯ СНЯТИЯ НАЖАТЬ ПРОБЕЛ
320 CLS: END
```

Заметим, что можно продолжить работу над темой «Выпуклые и невыпуклые многоугольники на экране ЭВМ». В качестве следующей и более трудной задачи можно предложить осуществить триангуляцию данного многоугольника, т. е. разбиение его на треугольники. Далее можно предложить вычислить площадь многоугольника, уже триангулированного.

Предупредим, что решение задачи триангуляции для случая выпуклого многоугольника — значительно более простое дело, нежели триангуляция невыпуклого многоугольника.

2. Линейка на экране ЭВМ. Геометрические построения, проводимые на доске, можно осуществлять и на экране ПЭВМ. Каким образом это осуществляется?

Можно составить программу, демонстрирующую какое-нибудь сложное построение. В этом случае ЭВМ шаг за шагом с требуемой скоростью будет вычерчивать на экране отрезки, проводить окружности, закрашивать в требуемые цвета выделенные для этого области; при этом школьники видят заранее заго-

товленный графический или геометрический сюжет. Конечно, можно приостанавливать процесс построения, давать какие-то пояснения, а затем продолжать демонстрацию. Пользоваться такими педагогическими заготовками очень просто, и они полезны. Особенно при объяснении задач по стереометрии.

Можно предложить другой путь. Речь идет о создании управляемой линейки на экране, о вручении пользователю экранного циркуля. Что это значит?

Разрабатывается программа моделирования действий с линейкой и циркулем на экране. Изображенные на экране отрезок и окружность перемещаются по указаниям человека, передающего свои воздействия через клавиатуру. В этом случае учитель может сам переместить отрезок так, как это требует построение, может его удлинить или повернуть, может изменить вручную с клавиатуры место расположения центра окружности и ее радиус.

Остановимся для примера на создании линейки на экране. Ниже приводится программа реализации, написанная на языке Бейсик.

```
10 '--------------- ЛИНЕЙКА НА ЭКРАНЕ -------------------
20 DIM LI(2, 2, 20)
30 CLS: SCREEN 3
40 X1=1: Y1=10: X2=400: Y2=10
50 CK=1: N=0
60 '------------- УСТАНОВКА ИСХОДНОГО ОТРЕЗКА -----------
70 LINE (X1, Y1)-(X2, Y2), 2
80 IF N=0 THEN 120
90 FOR I=1 TO N
100      LINE (LI(1, 1, I), LI(1, 2, I)) -
         (LI(2, 1, I), LI(2, 2, I)), 5
110 NEXT I
120 '------------- УПРАВЛЕНИЕ ЛИНЕЙКОЙ -----------------
130 S$=INKEY$: IF S$=" " THEN 130
140 IF S$="Z" THEN LINE (X1, Y1) - (X2, Y2), 0: Y1=Y1+CK
150 IF S$="X" THEN LINE (X1, Y1) - (X2, Y2), 0:
                   Y1=ABS(Y1-CK)
160 IF S$="C" THEN LINE (X1, Y1) - (X2, Y2), 0: Y2=Y2+CK
170 IF S$="V" THEN LINE (X1, Y1) - (X2, Y2), 0:
                   Y2=ABS(Y2-CK)
180 IF S$="A" THEN LINE (X1, Y1) - (X2, Y2), 0: X1=X1+CK
190 IF S$="S" THEN LINE (X1, Y1) - (X2, Y2), 0:
                   X1=ABS(X1-CK)
200 IF S$="D" THEN LINE (X1, Y1) - (X2, Y2), 0:
                   X2=ABS(X2-CK)
210 IF S$="F" THEN LINE (X1, Y1) - (X2, Y2), 0: X2=X2+CK
220 IF S$="G" THEN LOCATE 23, 60: INPUT "CK="; CK:
              'СКОРОСТЬ ДВИЖЕНИЯ
```

```
230 IF S$="N" THEN 260: 'ЗАПОМИНАНИЕ НОВОЙ ЛИНИИ
240 IF S$="B" THEN 280: 'ЗАВЕРШЕНИЕ РАБОТЫ
250 GOTO 70
260 N=N+1: LI(1, 1, N)=X1:LI(1, 2, N)=Y1:
    LI(2, 1, N)=X2: LI(2, 2, N)=Y2
270 GOTO 70
280 IF N=0 THEN 330
290 'ВЫВОД КООРДИНАТ ПОСТРОЕННЫХ ЛИНИЙ
295 CLS
300 FOR I=1 TO N
310 PRINT "X1="; LI(1, 1, I); "Y1=";
    LI(1, 2, I); "X2="; LI(2, 1, I);
    "Y2="; LI(2, 2, I)
320 NEXT I
330 END
```

В самом начале работы на экране появляется короткая линейка. Левый конец ее совпадает с точкой экрана (1, 10), правый — совпадает с точкой (400, 10). Линейка расположена горизонтально — это ее исходное положение.

Управлять линейкой — это значит изменять значение каждой координаты обоих ее концов. Нажатием определенных кнопок на клавиатуре можно придавать каждой из координат (X1, Y1; X2, Y2) приращение ($\pm$CK). Величина приращения в процессе работы может быть изменена вручную. От величины приращения CK зависит скорость перемещения линейки по плоскости экрана.

Назначение каждой управляющей клавиши приведено ниже.

Клавиша	Вызываемое действие
Z	Y1=Y1+CK
X	Y1=ABS (Y1−CK)
C	Y2=Y2+CK
V	Y2=ABS (Y2−CK)
S	X1=ABS (X1−CK)
F	X2=X2+CK
D	X2=ABS (X2−CK)
A	X1=X1+CK
G	Ввод нового значения CK
N	Запоминание линии
B	Завершение работы

При необходимости любая из полученных прямых может быть вычерчена. После нажатия на кнопку N на экране изменяется цвет линейки, и этот новый отрезок как бы «застывает» на месте. Рабочая же линейка может быть перемещена. Координаты концов зафиксированного отрезка запоминаются (строка 260).

Для окончания работы необходимо нажать на кнопку B, после этого на экране высвечиваются координаты всех зафиксированных в ходе работы отрезков (строки 300—320). Всего может быть зафиксировано 20 прямых, для этого открыт массив LI (2, 2, 20).

Аналогично можно изготовить и другие специальные инструменты для построения различных парабол, эллипсов, гипербол и других геометрических объектов.

Возможность вести построения вручную придает уроку геометрии особую доказательность. Школьники с большим желанием ведут построения различных кривых на экране. Разъяснение программ конструирования геометрических инструментов несет немало поучительного для курса программирования. Важно рассказать школьникам об использовании видеопамяти ЭВМ.

Ниже дополнительно предлагается решение человеком с помощью ЭВМ задачи Аполлония: построить окружность, касающуюся трех данных окружностей.

Основная идея решения состоит в том, что центр любой из окружностей есть точка пересечения трех вспомогательных окружностей. Для каждой из трех данных окружностей строится вспомогательная, ей концентрическая. На рисунке 60 показаны три исходные окружности и три вспомогательные (они изображены пунктиром).

Центр искомой окружности $O\,(x, y)$ лежит в точке пересечения трех вспомогательных окружностей с радиусами R_1-d, R_2-d, R_3-d, где R_1, R_2, R_3 — радиусы данных окружностей. В данном случае радиусы вспомогательных окружностей на одну и ту же величину (d) меньше, чем радиусы исходных окружностей.

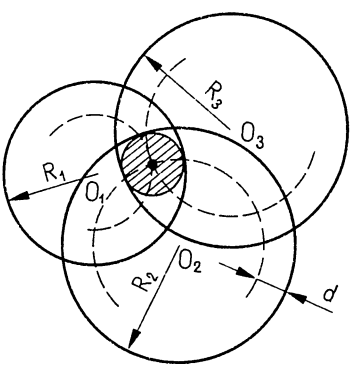


Рис. 60

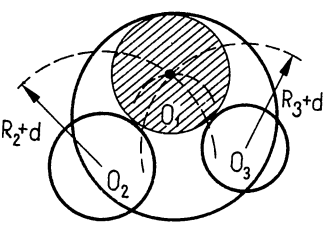


Рис. 61

На рисунке 61 показан случай, когда радиус одной из вспомогательных окружностей ($R_1 d$) меньше радиуса исходной окружности (R_1), а радиусы двух других вспомогательных окружностей (R_2+d, R_3+d) больше радиусов исходных окружностей.

По существу, координаты (x, y) центра одной из искомых окружностей и ее радиус R есть решение одной из восьми возможных систем уравнений:

$$\begin{cases} (x-x_1)^2+(y-y_1)^2=(R_1\pm d)^2; \\ (x-x_2)^2+(y-y_2)^2=(R_2\pm d)^2; \\ (x-x_3)^2+(y-y_3)^2=(R_3\pm d)^2. \end{cases}$$

Если поставить задачу программирования поиска корней таких систем, то это означает и программирование решения задачи Аполлония методами алгебры и аналитической геометрии.

Ниже предлагается иной подход. Обратим внимание на то, что приращения радиусов (d) одинаковы. Это обстоятельство подсказывает мысль организовать управление изменением радиусов с клавиатуры с одновременным вычерчиванием трех вспомогательных окружностей. Предлагается создать модель на экране: подчиняясь указаниям клавиатуры, все три вспомогательные окружности либо увеличивают свои радиусы и «растут», или одновременно «убывают», или одни «убывают», а другие «растут».

Решение задачи сводится к тому, что человек, решающий задачу, управляет изменением радиусов вспомогательных окружностей и фиксирует (останавливает изменение всех трех окружностей) в тот момент, когда три вспомогательные окружности пересекаются в одной точке.

Затем вводится в действие подвижный вспомогательный курсор. Управление курсором осуществляется с клавиатуры ПЭВМ. Курсор вначале размещается в центре экрана, а затем шаг за шагом продвигается в точку пересечения вспомогательных окружностей; в любой момент движения по экрану координаты курсора известны. Совместив курсор с точкой пересечения вспомогательных окружностей, человек, решающий задачу, выводит на экран координаты центра, радиус найденной окружности, и ЭВМ вычерчивает ее.

Может оказаться, что на экране обнаружилась еще одна точка пересечения вспомогательных окружностей. В этом случае снимаются и ее координаты, вычисляется радиус и строится еще одна искомая окружность.

Полученные координаты центров и радиусы найденных окружностей заносятся в созданные для этого массивы и в конце решения задачи используются для построения всех найденных окружностей. Возможно получение восьми окружностей.

Текст программы, снабженный развернутыми комментариями, приводится ниже. Предложенное решение является хорошим примером быстрого машинного подхода к решению геометрических задач (ранее так задачи не решались).

```
10 '------------------ ЗАДАЧА АПОЛЛОНИЯ -----------------
20 '---------- ДИАЛОГОВЫЙ КОНСТРУКТИВНЫЙ ПОДХОД ---------
30 SCREEN 3: CLS: DIM XC(2, 8), YC(2, 8), RC(2, 8)
40 KEY 6, " ": KEY 7, " ": KEY 8, " ": KEY 9, " ":
   KEY 10, " "
50 DEV1=1 : DEV2=1 : DEV3=1 : D=1
60 INPUT "ВВОД X1, Y1, R1 -"; X1, Y1, R1
70 INPUT "ВВОД X2, Y2, R2-"; X2, Y2, R2
80 INPUT "ВВОД X3, Y3, R3 -"; X3, Y3, R3
90 CLS
100 FOR DEV1=-1 TO 1 STEP 2
110 FOR DEV2=-1 TO 1 STEP 2
120 FOR DEV3=-1 TO 1 STEP 2
130 RR1=R1 : RR2=R2 : RR3=R3
140 '-------- ФОРМИРОВАНИЕ ИНФОРМАЦИОННОГО ОКНА ---------
150 LINE (460, 10) - (617, 168), 1, B
160 LINE (458, 8) - (619, 170), 1, B
170 LOCATE 2, 60: PRINT "D1 D2 RX1 RX2"
180 LOCATE 3+OC, 60: PRINT DEV1; DEV2; DEV3
190 '-------- ИЗОБРАЖЕНИЕ ИСХОДНЫХ ОКРУЖНОСТЕЙ ----------
200 CIRCLE(X1, Y1), R1, 1
210 CIRCLE(X2, Y2), R2, 3
220 CIRCLE(X3, Y3), R3, 5
230 '----- УПРАВЛЕНИЕ ВСПОМОГАТЕЛЬНЫМИ ОКРУЖНОСТЯМИ -----
240 LOCATE 24, 2: PRINT SPACE$ (40);
250 LOCATE 24, 2: PRINT "ДЛЯ ОТМЕНЫ ПОИСКА НАЖМИТЕ (N)";
260 LOCATE 3+OC, 60: PRINT DEV1; DEV2; DEV3
270 A$=INKEY$: IF A$=" " THEN 260
280 IF LEN(A$)=2 AND ASC(RIGHT$(A$, 1))=72
    THEN RH1=RR1: RH2=RR2: RH3=RR3: RR1+RR1+DEV1:
    RR2=RR2+DEV2: RR3=RR3+DEV3: GOTO 320
290 IF LEN(A$)=2 AND ASC(RIGHT$(A$, 1))=80
    THEN RH1=RR1: RH2=RH2: RH3=RR3: RRI=RR1-DEV1:
    RR2=RR2-DEV2: RR3-RR3-DEV3: GOTO 320
300 IF ASC(A$)=27 THEN GOTO 400
310 IF A$="N" OR A$="H" THEN OC=OC+1:
    COSUB 800: GOTO 670 ELSE 260
320 CIRCLE(X1, Y1), ABS(RH1), 0
330 CIRCLE(X2, Y2), ABS(RH2), 0
340 CIRCLE(X3, Y3), ABS(RH3), 0
350 CIRCLE(X1, Y1), ABS(RR1), 2
360 CIRCLE(X2, Y2), ABS(RR2), 2
370 CIRCLE(X3, Y3), ABS(RR3), 2
380 GOTO 200
390 '------- УПРАВЛЕНИЕ ВСПОМОГАТЕЛЬНЫМ КУРСОРОМ --------
400 X=300: Y=200: LOCATE 23, 65: PRINT "D="; D
410 CIRCLE(X, Y), 1, 15
420 A$=INKEY$: IF A$=" " THEN 420
```

```
430 IF LEN(A$)=2 AND ASC(RIGHT$(A$, 1))=64 THEN D=D+1
440 IF LEN(A$)=2 AND ASC(RIGHT$(A$, 1))=65 THEN D=D-1
450 IF D<=0 OR D>20 THEN D=1
460 LOCATE 23, 65: PRINT "D="; D
470 IF LEN(A$)=2 AND ASC(RIGH$(A$, 1))=72
    THEN CIRCLE(X, Y), 1, 0: Y=Y-D: GOTO 410
480 IF LEN(A$)=2 AND ASC(RIGHT$(A$, 1))=80
    THEN CIRCLE(X, Y), 1, 0: Y=Y+D: GOTO 410
490 IF LEN(A$)=2 AND ASC(RIGHT$(A$, 1))=75
    THEN CIRCLE(X, Y), 1, 0: X=X-D: GOTO 410
500 IF LEN(A$)=2 AND ASC(RIGHT$(A$, 1))=77
    THET CIRCLE(X, Y),·1, 0: X=X+D; GOTO 410
510 '------ С'ЕМ ПАРАМЕТРОВ ИСКОМЫХ ОКРУЖНОСТЕЙ ---------
520 IF ASC(A$) <> 27 THEN 420 ELSE ON W+1 GOTO 530, 850
530 OC=OC+1: RC(W+1, OC)=ABS(RR1-R1): XC(W+1, OC)=X:
    : YC(W+1, OC)=Y: W=1
540 CIRCLE(XC(W, OC), YC(W, OC)), RC(W, OC) 15
550 LOCATE 23, 4: PRINT "X="; X, "Y="; Y, "R="; RC(W, OC)
560 LOCATE 24, 2: PRINT SPACE$(40);
570 LOCATE 24, 2: PRINT "С'ЕМ НОВЫХ КООРДИНАТ (Y/N)";
580 A$=INKEY$: IF A$=" " THEN 580
590 IF A$="Ы" OR A$="Y" THEN CIRCLE(XC(W, OC), YC(W, OC)),
    RC(W, OC),: CIRCLE(XC(W, OC), YC(W, OC)), 1, 0:
    GOTO 270
600 IF A$ <> "Н" AND A$ <> "N" THEN BEEP: GOTO 580
610 GOSUB 800
620 A$=INKEY$: IF A$=" " THEN 620
630 CIRCLE(XC(W+1, OC), YC(W+1, OC)), RC(W+1, OC), 0
640 CIRCLE(XC(W+1, OC), YC(W+1, OC)), 1, 0
650 LOCATE 2+OC, 72: PRINT RC(W+1, OC): W=0
660 '--------- ПОСТРОЕНИЕ ПЕРВОЙ ОКРУЖНОСТИ -------------
670 NEXT DEV3: NEXT DEV2: NEXT DEV1
680 '------------ ИТОГОВЫЙ РЕЗУЛЬТАТ --------------------
690 CLS
700 CIRCLE(X1, Y1), R1, 1
710 CIRCLE(X2, Y2), R2, 3
720 CIRCLE(X3, Y3), R3, 5
730·FOR OC=1 TO 8
740 FOR W=0 TO 1
750 CIRCLE(XC(W+1, OC), YC(W+1, OC)), RC(W+1, OC) 15
760 NEXT W
770 NEXT OC
780 END
790 '-------------- ПОДПРОГРАММЫ -----------------------
800 CIRCLE(X1, Y1), ABS(RR1), 0
810 CIRCLE(X2, Y2), ABS(RR2), 0
820 CIRCLE(X3, Y3), ABS(RR3), 0
830 RETURN
```

```
840 '-------------------------------------------------
850 RC(W+1, OC)=ABS(RR1-R1): XC(W+1, OC)=X: YC(W+1, OC)=Y
860 LOCATE 2+OC, 68: PRINT RC(W+1, OC);
870 CIRCLE(XC(W+1, OC), YC(W+1, OC)), RC(W+1, OC), 15:
    W=1
880 LOCATE 23, 4: PRINT "X="; X, "Y="; Y, "R="; RC(W+1,
    OC)
890 'PRINT РЦ(2, ОЦ), ОЦ, В
900 GOTO 610
```

Школьники могут исследовать возможности программы, рассмотрев, например, задачу Аполлония, когда радиусы исходных окружностей малы ($R_1 = R_2 = R_3 = 1$). В этом случае задача сводится к проведению окружности через три данные точки (окружность радиуса 1 может рассматриваться как точка).

Аналогично можно получить решение задачи: провести окружность через данную точку и касающуюся двух данных окружностей или провести окружность через две данные точки, касающуюся данной окружности.

Столь тесное взаимодействие с ЭВМ позволяет эффективно решать многие задачи. Художник-модельер может быстро подбирать рисунок необходимых ему тканей; дизайнер работает над формой изделия; учитель быстро ведет построение изображений на экране, используя при этом различные инструменты и цвет. Такой режим активного взаимодействия человека и ЭВМ называется интерактивным.

§ 42. ДИАЛОГ ЧЕЛОВЕКА И ПЭВМ. ПРОГРАММИРОВАНИЕ ИГР

Важно понимать, как организуется диалог между человеком и вычислительной машиной, знать возможности такого диалога и представлять задачи, в которых диалог необходим.

Можно рассказать учащимся об информационно-справочных системах, об учебных курсах, реализуемых через ЭВМ, об играх с ЭВМ. Во всех этих случаях взаимодействие человека с машиной или вычислительной системой требует диалога.

Мы рекомендуем один урок посвятить рассказу о программировании процесса игры между человеком и ПЭВМ. На таком уроке еще раз обращается внимание на тщательность разработки алгоритма и показываются некоторые элементарные приемы организации диалога.

Известно, что ЭВМ может успешно соперничать с человеком в различных интеллектуальных играх. Ниже на примере показывается, как создаются такие программы, обсуждается, какие математические и программистские задачи при этом приходится решать.

У с л о в и я и г р ы. К началу игры в коробке находится некоторое число (N) камешков. Два соперника (человек и ЭВМ)

поочередно извлекают из коробки любое количество камешков, не превышающее, однако, половины имеющихся в ней на момент хода. Выигрывает тот, кто берет последний камешек.

Беспроигрышная стратегия в этой игре известна:

если к началу игры количество камешков выражается числом вида $3 \cdot 2^k - 1$, где $k = 0, 1, 2, 3, \ldots$, то в игру следует вступить вторым, т. е. первый ход уступить сопернику;

при любом ходе нужно брать столько камешков, чтобы количество оставшихся выражалось числом вида $3 \cdot 2^k - 1$.

Уточним постановку задачи на программирование. Необходимо составить такую программу, чтобы ЭВМ полностью имитировала поведение человека в игре. Это значит, что машина должна правильно оценить начальное количество предметов (N) и самостоятельно принять решение: начать игру или же уступить первый ход сопернику-человеку. В ходе игры машина должна безошибочно оценивать ситуацию, правильно рассчитывать и выполнять свой очередной ход. Конечно, ЭВМ должна следить за тем, соблюдает ли правила игры соперник.

Очень важно научить ЭВМ действовать в «трудной» обстановке, складывающейся в игре. Например, при начальных условиях, когда число камешков $N = 19$, в игру ей следовало бы вступить первой, но по жребию игру начал соперник-человек, и теперь ЭВМ придется внимательно следить за его поведением и ждать, когда он, выполняя свой ход, допустит ошибку. В этот момент машина должна сделать такой ход, чтобы инициатива в игре перешла к ней и довести игру до своей победы.

Может, однако, оказаться так, что человек ошибки не допустит, будет играть в соответствии со стратегией. В этом случае машина должна играть до конца, но затем признать свое поражение.

Для придания поединку всех человеческих черт нужно научить ЭВМ подавать реплики, необходимые в игре. Среди таких реплик могут быть замечания человеку о том, что он нарушает правила игры или сообщение: «Сдаюсь, благодарю за игру» или «Вы проиграли. Не хотите ли сыграть еще одну партию?» и т. д.

Ясно, что моделирование процесса игрового поединка на ЭВМ требует от программиста продумывания многих деталей.

Рассмотрим наиболее важные вопросы. Прежде всего программист обдумывает структуру программы в целом, представляет ее схематически в виде крупных блоков (рис. 62).

Четыре блока соответствуют наиболее важным разделам программы. В «блоке контроля за выполнением алгоритма» содержатся команды, выполняя которые машина проверяет, следует ли человек беспроигрышной стратегии. Если на каком-то ходу человек принял ошибочное решение, то управление процессом игры передается «блоку выигрыша». Выполняя команды этого блока, ЭВМ действует строго в соответствии с беспроигрышной стратегией и побеждает.

Если же человек действует в игре безупречно, то машине остается только выжидать в надежде поймать человека на ошибке. Поведение ЭВМ в таких ситуациях определяется командами из «блока проигрыша» и «блока контроля за выполнением алгоритма».

Контроль за соблюдением правил игры человеком ЭВМ ведет, руководствуясь командами из «блока контроля за правильностью хода».

Игра может начаться с того, что ЭВМ предложит человеку самому назвать начальное количество камешков (N). Это можно запрограммировать так:

```
 5 PRINT "укажите начальное количество камешков N"
10 INPUT "N=", N
```

Далее ЭВМ предлагает сопернику вопрос:

```
15 PRINT   "Каким по очереди хотите вступить в игру: первым или вторым?
           Введите 1 или 2"
20 PRINT   "Вступаю в игру": INPUT R
```

Одной из центральных задач в создании данной программы является проверка, относится ли конкретное число камешков к числам вида $3 \cdot 2^k - 1$. Эта задача вполне по силам учащимся. Возникают и другие программистские задачи.

Все рассмотренное дает некоторое представление о тех вопросах, которые приходится решать программисту при моделировании процесса игры на ЭВМ.

Нужно заранее позаботиться о разработке программы (это под руководством учителя может сделать и школьник), и учащиеся должны сыграть несколько партий. При этом обратить внимание учащихся на полезность разработки таких программ и

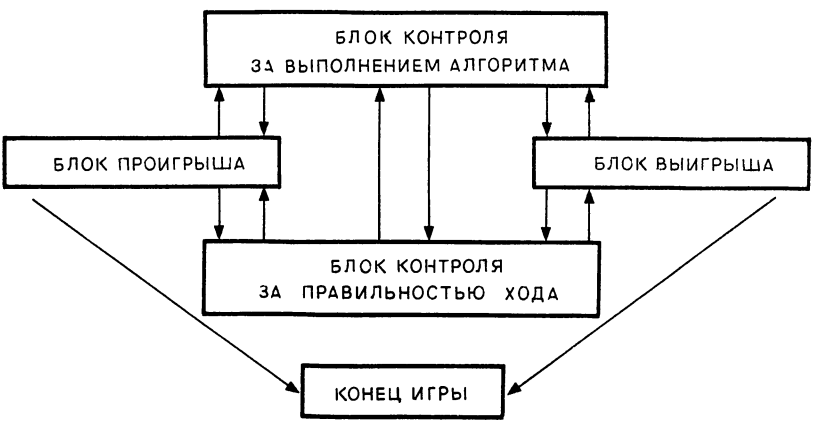


Рис. 62

такого типа игр. В таких играх на первом месте — стратегия, ее моделирование. Такие игры учитель должен противопоставлять играм типа «Гонка» и «Космический бой».

Дополнительно можно дать учащимся следующие задания.

1. Каким может быть фрагмент программы для ответа на вопрос: является ли число N числом вида 2^k-1, $k \in N$?

2. Каким должен быть фрагмент программы для ответа на вопрос: является ли число N числом вида 2^k+1, $k \in N$?

3. В коробке содержится K предметов. Ход человека: он берет X предметов. Какой может быть фрагмент программы, проверяющий условие: $x=3 \cdot 2^k-1$, $k \in N$?

4. Каким будет алгоритм, если в описанной выше игре будет изменено правило определения победителя: победителем считается тот, кто оставит сопернику последний предмет?

5. Разработайте программу моделирования игры Баше.

§ 43. СОРТИРОВКА ЭЛЕМЕНТОВ МАССИВА

Среди практических задач, к решению которых привлекаются ЭВМ, может быть, одними из важнейших являются задачи сортировки. Под сортировкой понимается перераспределение элементов массива в соответствии с наперед заданным порядком.

Например, в массиве из N чисел осуществить их перестановку (нумерацию) так, чтобы после перестановки в массиве первым элементом было самое маленькое число, каждое следующее было бы больше предшествующего, а последнее — самое большое из чисел данного массива.

Если сортируется массив не чисел, а, например, слов, то сортировку можно производить, например, в лексикографическом порядке или выделить все слова, начинающиеся или кончающиеся какой-то буквой или слогом.

Близкими к сортировке являются задачи включения новых элементов в уже упорядоченный массив или исключения из упорядоченного массива нескольких элементов с последующей перенумерацией так, чтобы в массиве не было пропусков, не было «пустых» элементов. К этому же классу относятся задачи выделения в массиве элементов с заданным свойством. Может потребоваться выборка всех чисел, кратных трем, или выборка фамилий мужчин старше 30 лет или элементов с другими свойствами.

Все сказанное можно обобщить и сформулировать так, как это принято в профессиональном программировании. Сортировке подвергаются файлы записей в соответствии с их ключами. Под файлом, образно говоря, следует понимать некоторую картотеку, заведенную, например, на сотрудников предприятия, или картотеку изделий, которые выпускает предприятие. Все сведения о каждом сотруднике или изделии оформляются в виде записей. Каждая запись состоит из полей или небольших порций информа-

ции. Если речь идет о сотруднике, то полями в записи являются фамилия, специальность, год рождения, квалификационный разряд, стаж работы и другие показатели. Если речь идет об изделии, то полями в записи могут быть стоимость изделия, цвет, вес, регистрационный номер, номер ГОСТа или иной показатель.

При сортировке файла одно из полей рассматривается как ключ. Если ключом является фамилия, то файл может сортироваться с целью расположения фамилий в порядке алфавита. Если ключом является вес изделия, то файл можно сортировать в порядке возрастания или убывания веса.

Методов сортировки очень много, существует хорошо проработанная теория сортировок и методика их реализации на ЭВМ. Упомянем о некоторых основных идеях. Если перед сортировкой весь сортируемый файл удается поместить в оперативном ЗУ, то осуществляемую затем сортировку называют внутренней. Часто, однако, сортируемые файлы настолько велики, что лишь их часть помещается в оперативном ЗУ. В этом случае в процессе сортировки идет обмен между внутренним и внешним ЗУ. Такую сортировку называют внешней.

В школьном курсе рассматриваются внутренние сортировки, сортируются записи, предварительно объединенные в массив. Наиболее простым и, по-видимому, всегда сообщаемым школьникам является метод сортировки перестановкой элементов массива.

Суть метода в том, что на первом шаге сортировки сравнивается второй элемент с первым, и если необходимо, то эти элементы меняются местами. Затем третий элемент сравнивается со вторым, и вновь при необходимости они меняются местами (получают новые индексы). Таким образом, за $(N-1)$ шаг самый большой (или самый малый) элемент будет продвинут в конец сортируемого массива, всплывет, как пузырек, «вверх».

Весь цикл продвижения элементов «вправо» повторится еще несколько раз, пока все элементы не расположатся либо в порядке возрастания, либо в порядке убывания.

Описанный подход положен в основу программы, составленной школьником:

```
10 'СОРТИРОВКА ЧИСЕЛ
20 PRINT "ИЗ СОРТИРУЕМЫХ ЭЛЕМЕНТОВ СФОРМИРУЙТЕ МАССИВ
   A(N)"
30 INPUT "ВВОДИТЕ N"; N: DIM A(N)
40 PRINT "ВВОД ДАННЫХ"
50 FOR I=1 TO N: INPUT A(I): NEXT I
60 PRINT "МАССИВ СФОРМИРОВАН"
70 PRINT "УКАЖИТЕ ПОРЯДОК СОРТИРОВКИ"
80 PRINT "ПО ВОЗРАСТАНИЮ, ДА ИЛИ НЕТ"
90 INPUT A$
100 IF A$="ДА" THEN 160
```

```
110 FOR I=1 TO N-1
120 FOR J=1 TO N
130 IF A(I)<A(F) THEN SWAP A(I), A(J)
140 NEXT J: NEXT I
150 PRINT "МАССИВ ОТСОРТИРОВАН" GOTO 200
160 FOR I=1 TO N-1
170 FOR J=I+1 TO N
180 IF A(I)>=A(J) THEN SWAP A(I), A(J)
190 NEXT J: NEXT I: GOTO 150
200 FOR K=1 TO N
210 PRINT A(K); : NEXT K
220 END
```

Описанный метод называется «методом пузырька». Если предположить, что каждая из перестановок двух соседних элементов равновероятна, то требуется $n^2/4$ сравнений и перестановок элементов.

Существуют другие алгоритмы, в которых общее число сравнений и перестановок не превышает $n \cdot \log_2 n$. Эти алгоритмы более быстры (сравните числа $n \log_2 n$ и $n^2/4$), нежели «метод пузырька». Однако изучение этого метода — дело хотя и очень интересное, но может увести учителя с его учениками далеко в сторону.

Преподавателю рекомендуем, положив в основу приведенную программу сортировки, дать задание учащимся — внести изменения в программу, с тем чтобы ЭВМ могла осуществить сортировку букв латинского алфавита. В отсортированном массиве буквы должны быть расположены в алфавитном порядке (лексикографическом).

Решение такого задания расширит представления о возможностях ЭВМ в задачах упорядочения элементов.

Текст программы может быть таким, как приведенный ниже:

```
10 'СОРТИРОВКА СИМВОЛЬНЫХ ВЕЛИЧИН
20 PRINT "ИЗ СОРТИРУЕМЫХ ЭЛЕМЕНТОВ СФОРМИРУЙТЕ МАССИВ
   A$(N)"
30 INPUT "ВВОДИТЕ N"; N: DIM A$(N)
40 PRINT "ВВОД СОРТИРУЕМЫХ СИМВОЛОВ"
50 FOR I=1 TO N: INPUT A$(I): NEXT I
60 PRINT "МАССИВ СФОРМИРОВАН"
70 PRINT "УКАЖИТЕ ПОРЯДОК СОРТИРОВКИ"
80 PRINT "ПО ВОЗРАСТАНИЮ, ДА ИЛИ НЕТ"
90 INPUT A$
100 IF A$= "ДА" THEN 160
110 FOR I=1 TO N-1
120 FOR J=I TO N
130 IF A$(I) < A$(J) THEN SWAP A$(I), A$(J)
140 NEXT J: NEXT I
150 PRINT "МАССИВ ОТСОРТИРОВАН": GOTO 200
```

```
160 FOR I=1 TO N-1
170 FOR J=I+1 TO N
180 IF A$(I)>=A$(J) THEN SWAP A$(I), A$(J)
190 NEXT J: NEXT I: GOTO 150
200 FOR K=1 TO N
210 PRINT A$(K);: NEXT K
220 END
```

Из методов сортировки рассмотрим еще так называемый метод сортировки с возвратом. Рассказать о методе можно в форме комментария к готовой программе:

```
10 '------------------ СОРТИРОВКА С ВОЗВРАТОМ ------------
20 PRINT "В ОПЕРАТОР DATA ВВОДИТЕ ДАННЫЕ"
30 PRINT "ПОСЛЕ ВВОДА RUN 60"
40 LIST 50
50 DATA
60 INPUT "КОЛ-ВО ЭЛЕМЕНТОВ N="; N
70 DIM A(N)
80 FOR I=1 TO N
90 READ A(I): PRINT A(I);
100 NEXT I
110 PRINT
120 I=1
130 IF A(I)<A(I+1) THEN 200
140 SWAP A(I), A(I+1)
150 K=I
160 IF A(K-1)<=A(K) THEN 200
170 SWAP A(K), A(K-1)
180 K=K-1
190 IF K <>1 THEN 160
200 I=I+1
210 IF I<N THEN 130
220 FOR I=1 TO N
230 PRINT A(I);
240 NEXT I
250 END
```

В строках 10—100 выписаны операторы формирования вручную массива A (N) из числовых данных. Все исходные данные размещаются в хранилище DATA — это позволяет неоднократно использовать их для решения различных задач. Для восстановления DATA потребуется использование оператора RESTORE.

В операторах 130—140 реализуется сравнение элементов массива попарно и продвижение большего из сравниваемых чисел «вправо». В операторах 160—170 сравнению подвергаются числа, продвигаемые «влево».

В строках 200—210 выписаны операторы контроля за завершенностью алгоритма сортировки — сортировка продолжается,

пока не осуществлено сравнение двух последних элементов массива: A (N) и A (N—1).

В операторах 220—240 реализован вывод отсортированного массива A.

Приводим пример отсортированного массива:

```
Кол-во элементов N=? 10
11.87 23.76—56.32—34.89.43 1.89 23 76.78 41.76—34
—56.32 —34.89 —34. 43 1.89 11.87 23 23.76 41.76 76.78
OK
```

Задания для учащихся по теме «Сортировка элементов массива»:

1. Составить программу вывода на экран самого большого (самого малого) элемента массива A (массив A находится в памяти ЭВМ).

2. Составить программу сортировки массива A, находящегося в памяти, по убыванию абсолютных величин его элементов. Отсортированный массив вывести на экран.

3. В DATA хранится неупорядоченный набор из N чисел. Составить программу, работая по которой ЭВМ сформирует из этого набора чисел массив A и затем в этом массиве расположит числа в порядке их возрастания.

4. Дан упорядоченный массив A — чисел, расположенных в порядке их возрастания. Дано некоторое число a, которое необходимо вставить в данный массив, чтобы упорядоченность массива сохранилась. Составить программу для решения этой задачи.

5. Составить программу для быстрой перестройки данного массива A, в котором элементы расположены в порядке возрастания, так, чтобы после перестройки эти же элементы оказались расположенными в порядке убывания.

6. Дан массив A, содержащий как отрицательные, так и положительные числа. Составить программу исключения из него всех отрицательных чисел, а оставшиеся положительные расположить в порядке их возрастания.

7. Составить программу, работая по которой ЭВМ будет из DATA брать одно число за другим и формировать из них массив A, располагая числа в порядке возрастания.

8. Дан список авторов в форме массива A$. Составить программу формирования указателя авторов в алфавитном порядке и вывести его на экран.

9. Имеется N абонентов телефонной станции. Составить программу, в которой формируется список по форме: номер телефона, фамилия (номера идут в порядке возрастания).

10. Имеется N слов различной длины, все они занесены в ОЗУ. Составить программу упорядочения слов по возрастанию их длин.

Глава VII. НЕКОТОРЫЕ ВОПРОСЫ МЕТОДИКИ КУРСА В ЦЕЛОМ

§ 44. ОБЩИЕ ПРОБЛЕМЫ ВЕДЕНИЯ КУРСА

Первый опыт преподавания в школе курса «Основы информатики и вычислительной техники» показал, что этот курс имеет немало специфических особенностей.

Обратим внимание на некоторые.

Курс не может считаться курсом «по выбору». Основанием для такого утверждения является фундаментальность самого феномена «информация», значимость которого сравнима со значимостью такого понятия, как «энергия».

Упомянутая фундаментальность выражается в необычайно сильно выраженных межпредметных связях курса. Из этого, кстати, вытекает требование к высокой культуре и эрудиции преподавателя.

Можно предположить, что курс «Основы информатики и вычислительной техники» не будет столь консервативным, как курсы математики, языка и естественных наук. Стремительность прогресса в вычислительной технике, изменения в формах общения человека и ЭВМ требуют современного отражения в методике преподавания. Труд учителя, ведущего этот курс, специфичен по ряду параметров.

Особое значение, и может быть главенствующее, приобретает общее психолого-педагогическое обеспечение всего процесса компьютеризации школьного образования.

Нужно считаться с тем, что «эффект новизны» в деле внедрения информатики в школу уходит в прошлое; на первое место взамен выдвигаются более глубокие мотивы, особую роль играет склонность школьников к занятиям точными науками. В этом случае личность учителя переоценить нельзя.

От учителя требуется ясное и взвешенное осознание роли ЭВМ во всем учебном процессе. Нам представляется, что педагогическое мастерство учителя именно в преподавании основ информатики особенно значимо. Вся техника и создаваемая ею среда — это лишь поле и средства деятельности педагога.

При ведении курса, особенно в его лабораторно-практической части, изменяется весь характер работы учителя. В частности, резко возрастает нагрузка на учителя непосредственно в ходе занятия, индивидуализация при работе за отдельной ПЭВМ при-

водит к высокому уровню активности работы группы в целом. Немалую роль играет требование школьников получить ответ на любой вопрос «сразу», а при работе с текстами программ и оригинальными алгоритмами это по силам не каждому учителю.

Возрастает нагрузка и при подготовке к уроку. В новом курсе повышается роль дидактического обеспечения курса. Здесь учитель встречается с нетрадиционными средствами, с системами, реализуемыми через ЭВМ. Речь идет о материалах для закрепления умений работы на ЭВМ как на персональной машине. Речь идет и о системах контроля, когда контролирующая и оценивающая функции передаются ЭВМ.

В этих условиях учителю нужно самому разрабатывать дидактические материалы более высокого интеллектуального уровня и действительно знать, чему ему удалось научить школьников, чему не удалось.

Здесь очень важна педагогическая умеренность; завышение требований к учебной подготовке, несоответствие требованиям программы недопустимы. В равной мере недопустимы и занижение требований или увлечение одним разделом за счет другого. Курс называется «Основы информатики и вычислительной техники», и поэтому учитель не должен преувеличивать значение программирования за счет поверхностного изучения основ вычислительной техники. Исходить, вероятно, следует из того, что учащиеся в будущем вправе избирать путь «в конструкторы ЭВМ» и «в программисты». Школа готовит их к этим двум направлениям, а не к одному из них.

Особую роль играют медицинские требования к организации работы школьников за пультом современных ПЭВМ. Учителю надо учитывать, что у детей и подростков центральная нервная система и зрительные органы находятся в стадии созревания, незрелая центральная нервная система у детей находится в состоянии повышенной судорожной готовности по сравнению со взрослыми; поэтому для них высока опасность развития фотоэпилепсии, связанной с мерцанием экрана при частоте 50 Гц. Это только одна из ряда причин, по которой Минздрав СССР и Минпрос СССР в 1987 г. ввели «Временные рекомендации по режимам занятий учащихся за видеотерминалами ЭВМ». Об этих рекомендациях и изменениях в них учитель должен помнить. Заметим, кстати, что требования эти сегодня достаточно жестки: еженедельная длительность работы за пультом в X—XI классах не более 20 мин.

Важной особенностью работы учителя по новому курсу являются новые требования к формированию его личного учебного архива, набора дидактических материалов. Учителю нужно иметь персональное ЗУ для хранения всех учебных и демонстрационных программ, т. е. «архив на гибком диске», хорошо организованный и квалифицированно поддерживаемый. Это требует дополнительной работы.

Учитель информатики может в целом изменить свое педагоги-

ческое делопроизводство. Современная техника позволяет учителю вести самый разнообразный учет всей работы и учет результатов учащихся. Он может создавать автоматизированную информационно-поисковую систему на два года, на весь цикл обучения новому предмету.

§ 45. О ДИДАКТИЧЕСКИХ МАТЕРИАЛАХ. КАРТОЧКИ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Работа в классе, снабженном комплектом ПЭВМ, требует разработки специфических дидактических материалов. Ниже предлагаются в качестве примера образцы карточек-заданий, рекомендуемых для использования на занятиях по изучению темы «Программирование на Бейсике».

Карточки предлагаются для занятий по всем темам, рекомендуемым для закрепления навыков в составлении простейших программ. В каждой карточке учащийся работает с текстом программы, закрепляет умения обращаться с текстом, учится анализировать тексты, вносить в них изменения, пробует силы в составлении простейших программ. Карточки составлены по темам:

1. Суммирование чисел на ЭВМ.
2. Суммирование чисел.
3. Вычисление средних.
4. Умножение чисел.
5. Решение уравнений.
6. Выделение отрезков, содержащих единственный корень уравнения.
7. Решение неравенств.
8. Решение систем линейных уравнений.
9. Символьные величины и их обработка.
10. Решение геометрических задач.
11. Формирование таблиц.
12. Реализация процесса простейших игр.

Предлагаемые карточки являются лишь вариантом возможных материалов, которые можно использовать в классе, снабженном ПЭВМ. Объем работы непосредственно за пультом электронно-вычислительной машины (после проработки заданий) составляет 20—22 мин.

Конечно, каждый учитель найдет свои формы и содержание такого типа учебных материалов. Это лишь примеры, взятые из опыта.

Суммирование чисел на ЭВМ

Дана программа суммирования первых 50 натуральных чисел:

```
5 '----------------- СУММИРОВАНИЕ ЧИСЕЛ -----------------
10 S=0: N=0
15 S=S+N
20 N=N+1
30 IF N<50 THEN 15
40 PRINT S
50 END
```

Выполните программу на ЭВМ.

а) Внесите изменения в программу, с тем чтобы ЭВМ вычислила сумму первых 35 натуральных чисел. Проверьте программу в работе.

б) Внесите изменения в программу, с тем чтобы ЭВМ вычислила сумму первых 40 четных чисел. Проверьте программу в работе.

в) Внесите изменения в программу, с тем чтобы ЭВМ вычислила сумму первых 30 нечетных чисел. Проверьте программу в работе.

г) Внесите изменения в программу, с тем чтобы ЭВМ вычислила сумму квадратов первых 20 натуральных чисел. Проверьте программу в работе.

д) Внесите изменения в программу, с тем чтобы ЭВМ вычислила сумму квадратов первых 15 положительных чисел, каждое из которых при делении на 5 дает в остатке 3. Проверьте программу в работе.

Суммирование чисел

1. В последовательности Фибоначчи каждое число, начиная с третьего, вычисляется по формуле

$$a_n = a_{n-1} + a_{n-2};\ a_1 = a_2 = 1.$$

Дана программа вычисления N первых чисел Фибоначчи:

```
10 '----------------- ЧИСЛА ФИБОНАЧЧИ -----------------
20 INPUT "КОЛ-ВО ЧИСЕЛ"; N
30 DIM F(N)
40 F(1)=1: F(2)=1
50 FOR I=3 TO N
60 F(I)=F(I-1)+F(I-2)
70 PRINT I; "-Е ЧИСЛО РАВНО"; F(I)
80 NEXT I
90 END
```

Проверьте программу в работе.

а) Как следует изменить программу, чтобы ЭВМ вывела на печать числа только с нечетными номерами?

б) Как следует дополнить программу, чтобы ЭВМ вычислила члены последовательности, заданные формулами

$$a_n = a_{n-1} + a_{n-2} + a_{n-3};\ a_1 = a_2 = a_3 = 1?$$

2. Какую работу выполняет ЭВМ, действуя по программе, приведенной ниже?

```
10 '---------------- ЧИСЛА ФИБОНАЧЧИ --------------------
20 INPUT "КОЛ-ВО ЧИСЕЛ"; N
30 DIM F(N)
40 F(1)=1; F(2)=1: S=2
50 FOR I=3 TO N
60 F(I)=F(I-1)+F(I-2)
70 PRINT I; "-Е ЧИСЛО РАВНО"; F(I)
80 S=S+F(I)
90 NEXT I
100 PRINT "S="; S
110 END
```

Вычисление средних

1. Дана программа вычисления среднего арифметического чисел.

```
10 '----------------- ВЫЧИСЛЕНИЕ СРЕДНИХ ---------------
20 INPUT "N="; N
30 DIM A(N)
40 FOR I=1 TO N
50 INPUT A(I)
60 NEXT I
70 INPUT "K="; K
80 S=0
90 FOR I=1 TO N
100 S=S+A(I)^K
110 NEXT I
120 M=(S/M)^(1/K)
130 PRINT M
140 END
```

Проверьте программу в работе при $N = 10$ и $K = 1$. Вычисление какого типа средней при этом ведется?

а) Внесите изменения в программу, с тем чтобы ЭВМ вычислила среднее квадратичное первых 8 чисел.

б) Внесите изменения в программу, с тем чтобы ЭВМ вычислила среднее кубическое всех чисел. Проверьте программу в работе.

в) Внесите изменения в программу, с тем чтобы ЭВМ вычислила среднее гармоническое первых 5 чисел. Проверьте программу в работе.

2. Составьте программу вычисления среднего геометрического первых 5 положительных чисел из N чисел. Вычисления осуществите по формуле

$$x = \sqrt{x_1 \cdot x_2 \cdot x_3 \cdot x_4 \cdot x_5}.$$

Умножение чисел

1. Дана программа умножения нескольких чисел, содержащихся в DATA:

```
10 DATA 12.3, 20.1, 13.2, 5.6, 1.7
20 P=1
30 FOR I=1 TO 5
40 READ X
50 P=P*X
60 NEXT I
70 PRINT P
80 END
```

Выполните программу на ЭВМ.

2. Дополните программу так, чтобы в результате было получено и выведено дополнительно на экран значение Q кубического корня из произведения.

3. Дополните программу так, чтобы в результате было получено произведение не чисел, хранящихся в DATA, а их квадратных корней.

4. Предлагается под номером 65 ввести оператор RESTORE. Что это даст? Какую роль играет оператор RESTORE в программах?

Решение уравнений

1. Дана программа решения уравнения $2/(p-3)-2=0$ на отрезке [3,5; 6]:

```
10 '-------------------- РЕШЕНИЕ УРАВНЕНИЙ --------------
20 A=3.5: B=6: E=.0001
30 X=(A+B)/2: Y=2/(X-3)-2
40 YA=2/(A-3)-2
50 IF Y=0 THET 80
60 IF Y*YA>0 THEN A=X ELSE B=X
70 IF (B-A)>E THEN 30
80 PRINT X
90 END
```

Выполните программу на ЭВМ.

а) Проверьте работу программы на отрезке [2; 3]. Как ведет себя ЭВМ? Объясните поведение машины.

б) Включите в программу «часы» (таймер) и узнайте время решения уравнения.

2. Дан текст программы, предназначенной для решения уравнения $(x+2)^2-3=0$ методом половинного деления отрезка, содержащего корень:

```
10 '-------------------- РЕШЕНИЕ УРАВНЕНИЙ --------------
20 PRINT "В СТРОКУ 50 ВВЕДИТЕ УРАВНЕНИЕ"
30 PRINT "ПОСЛЕ ВВОДА RUN 50"
40 LIST 50
50 DEF FNF(X)=(X+2)^2-3
60 INPUT A, B, E
70 X=(A+B)/2: Y=FNF(X)
80 YA=FNF(A)
90 IF Y=0 THEN 120
100 IF Y*YA>0 THEN A=X ELSE B=X
110 IF (B-A)>E THEN 70
120 PRINT X
130 END
```

Научитесь пользоваться подпрограммой. Решите уравнения:

а) $e^{x-2}-2=0$ на отрезке $[0; 3]$;
б) $(x+2)^2-3=0$ на отрезках $[-2; 2]$ и $[-5; -2]$;
в) $\sin(2x-1)-0{,}7=0$ на отрезке $[1{,}5; 2{,}5]$;
г) $x-\ln x-1{,}5=0$ на отрезке $[1; e]$.

Выделение отрезков, содержащих единственный корень уравнения

Дана программа выделения отрезков, содержащих единственный корень уравнения $1/4\,(x-3)^4-3\,(x-3)^3-(x-3)^2+1=0$:

```
10 '------------------ РЕШЕНИЕ УРАВНЕНИЙ ----------------
20 CLS: INPUT A, B, H
30 DEF FNX(X)=(X-3)^4/4-3*(X-3)^3-(X-3)^2+1
40 X1=A
50 Y1=FNX(X1)
60 Y2=FNX(X1+H)
70 IF X1>B THEN 110
80 IF Y1*Y2>0 THEN X1=X1+H: GOTO 50
90 PRINT "A="; X1; "B="; X+H
100 X1=X1+H: GOTO 50
110 PRINT "ПОИСК ОКОНЧЕН": END
```

Исходный отрезок $[1; 6]$, шаг $H=0{,}1$.

Выполните программу на ЭВМ. Сколько отрезков, содержащих единственный корень, найдено?

а) Внесите вручную изменения в данную программу, с тем

чтобы можно было выделить отрезки, содержащие по единственному корню уравнения $x^4-13x^2+36=0$.

Исходный отрезок $[-4;\ +4]$, шаг $H=0{,}01$.

б) Внесите вручную изменения в текст исходной программы и выделите отрезки, содержащие по единственному корню уравнения $x^4-3x^3-x^2+1=0$.

Исходный отрезок $[-2;\ 3]$, шаг $H=0{,}01$.

Решение неравенств

1. Дана программа решения квадратных неравенств вида $Ax^2+Bx+C>0$:

```
10 '-------------- РЕШЕНИЕ КВАДРАТНЫХ НЕРАВЕНСТВ --------
20 INPUT A, B, C
30 D=B^2-4*A*C
40 IF D>0 THEN 120
50 IF D=0 THEN 90
60 IF A>0 THEN 80
70 PRINT "РЕШЕНИЙ НЕТ": END LIST
80 PRINT "X - ЛЮБОЕ ДЕЙСТВ. ЧИСЛО": END
90 X3=-B/(2*A)
100 IF A<=0 THEN 70
110 PRINT "РЕШЕНИЕ - ЛЮБОЕ ДЕЙСТВ. X, КРОМЕ X="; X3: END
120 X1=(-B-SQR(D))/(2*A)
130 X2=(-B+SQR(D))/(2*A)
140 IF A>0 THEN 160
150 PRINT X2; "<X<"; X1: END
160 PRINT X1; "<X<"; X2;
170 END
```

Выполните программу на ЭВМ, решив следующие неравенства:

а) $x^2+6x+11>0$; б) $x^2+3x-10>0$.

2. Дана программа решения линейных неравенств вида $Ax+B>0$:

```
5 INPUT A, B
10 X=-B/A
15 IF A>0 THEN 25
20 PRINT "X<"; X: END
25 PRINT "X>"; X: END
```

Выполните программу на ЭВМ, решив следующие неравенства:

а) $2x-3<0$; б) $3{,}21x+19{,}65>0$.

3. Ниже приводится программа решения неравенств вида $F(x)>0$:

```
10 '--------------- РЕШЕНИЕ НЕРАВЕНСТВ F(X) > 0 ---------
20 CLS: PRINT "В СТРОКУ 60 ВВЕСТИ F(X)"
30 PRINT "ПОСЛЕ ВВОДА - RUN 50"
40 LIST 60
50 INPUT "ЗАДАННЫЙ ОТРЕЗОК A, B"; A, B
55 INPUT "ТОЧНОСТЬ ВЫЧИСЛЕНИЙ H"; H
60 DEF FNX(X)=
70 X=A: XL=A
80 FOR X=A TO B STEP H
90 IF FNX(X)*FNX(X+H)>0 THEN 130
100 S=SGN(FNX(XL)): XP=X
110 IF S=1 THEN PRINT XL, XP
120 XL=X+H
130 NEXT X: PRINT XL, B
140 PRINT "ПОИСК ОКОНЧЕН": END
```

Используя программу, решите неравенства:

а) $(x-1)(x-2)(x-3)>0$ на отрезке $[0; 4]$;
б) $e^{x-3,1}-2,82>0$ на отрезке $[0; 4]$;
в) $(x^3-8)(x-5)(2x-1)>0$ на отрезке $[-2; 5,5]$;
г) $\sin(3x-1)>0$ на отрезке $[-1; 5]$.

Решение систем линейных уравнений

Дана программа для решения систем двух линейных уравнений с двумя неизвестными вида

$$\begin{cases} a_{11}x_1+a_{12}x_2=a_{13}; \\ a_{21}x_1+a_{22}x_2=a_{23}. \end{cases}$$

```
10 '--------------- РЕШЕНИЕ СИСТЕМ УРАВНЕНИЙ ------------
20 DIM A(2, 3)
30 FOR I=1 TO 2
40 FOR K=1 TO 3
50 INPUT A(I, K)
60 NEXT K
70 NEXT I
80 D=A(1, 1)*A(2, 2)-A(2, 1)*A(1, 2)
90 IF D=0 THEN PRINT "ЕДИНСТВЕННОГО РЕШЕНИЯ НЕТ": END
100 DX=A(1, 3)*A(2, 2)-A(2, 3)*A(1, 2)
110 DY=A(1, 1)*A(2, 3)-A(2, 1)*A(1, 3)
120 X=DX/D: Y=DY/D
130 PRINT X, Y
140 END
```

Решите системы и проверьте найденные корни:

а) $\begin{cases} 5x-3y=16; \\ 7x+2y=41; \end{cases}$ б) $\begin{cases} 9,21x-5,41y=34,1436; \\ 17,39x+11,56y=47,0372. \end{cases}$

1. Ниже приводится программа формирования массива A$ из символьных величин:

```
10 'ФОРМИРОВАНИЕ МАССИВА СЛОВ
20 INPUT "ВВЕДИТЕ N"; N
30 DIM A$(N)
40 FOR I=1 TO N
50 INPUT "ВВЕДИТЕ СЛОВО A$"; A$
60 PRINT A$
70 NEXT I
80 END
```

Выполните программу на ЭВМ, взяв в качестве элементов создаваемого массива служебные слова Бейсика:
FOR, POINT, IF, LET, ABS, LEN, INPUT, NEXT, GOTO, END, READ, RESTORE, LIST, OR, AND.

Дополните вышеприведенную программу операторами, выписанными ниже:

```
80 FOR K=1 TO N-1
90 FOR J=K+1 TO N
100 IF A$(K)>A$(J) THEN SWAP A$(K), A$(J)
110 NEXT J
120 NEXT K
130 FOR L=1 TO N: PRINT A$(L): " ";
140 NEXT L
150 END
```

Выполните программу на ЭВМ. Какая проделана работа?

2. Дана программа, которая выбирает из DATA одно за другим слова (A$), вычисляет их длину, и если слово имеет длину K, то выводит его на экран.

В DATA перед началом работы помещено N слов:

```
10 PRINT "В DATA  ПОМЕСТИТЕ N СЛОВ"
20 PRINT "ПОСЛЕ ВВОДА RUN 40"
30 LIST 40
40 DATA
50 INPUT "K="; K
60 INPUT "N="; N
70 FOR I=1 TO N
80 READ A$
90 IF LEN(A$)=K THEN PRINT A$
100 NEXT I
110 END
```

Проверьте программу в работе при различных K.

а) Как следует изменить программу, чтобы ЭВМ вывела на экран слова меньшей чем K длины?

б) Введите в программу оператор RESTORE, объясните его содержание.

в) Введите в DATA целые положительные многоразрядные числа (всего 10 чисел) и запустите программу при K=3. Какой результат будет получен?

3. Дана программа формирования массива символьных переменных A$ и выборки из сформированного массива слов, начинающихся на заданную букву X$.

```
10 REM ФОРМИРОВАНИЕ МАССИВА СИМВОЛЬНЫХ ПЕРЕМЕННЫХ
20 INPUT "ВВЕДИТЕ ЧИСЛО ЭЛЕМЕНТОВ МАССИВА-N"; N
30 DIM A$(N)
40 FOR I=1 TO N
50 PRINT "ВВОДИТЕ"; I; "-Е СЛОВО A$";
60 INPUT A$(I)
70 NEXT I
80 REM ВЫБОР ИЗ ДАННОГО МАССИВА СЛОВ С ПЕРВОЙ БУКВОЙ X$
90 INPUT "ВВЕДИТЕ X$"; X$
100 FOR I=1 TO N
110 IF LEFT$(A$(I),1)=X$ THEN PRINT A$(I); " ";
120 NEXT I
```

а) Проверьте программу в работе, задав различные N и A$, X$.

б) Используя программу, сформируйте числовой массив из одноразрядных чисел и выведите числа, входящие в массив, в порядке возрастания.

в) Какие изменения следует внести в программу, чтобы ЭВМ из массива, сформированного из чисел и слов, вывела на печать только числа?

Решение геометрических задач

1. Дана программа построения и изображения на экране точки A_1, симметричной данной точке A относительно точки O.

```
10 '----------- СИММЕТРИЯ ОТНОСИТЕЛЬНО ТОЧКИ ------------
20 INPUT XA, YA, XO, YO
30 XA1=2*XO-XA: YA1=2*YO-YA
40 PRINT XA1, YA1
50 CLS: SCREEN 3: COLOR 1, 15, 3
60 CIRCLE(XA, YA), 2, 3
70 CIRCLE(XO, YO), 2, 5
80 CIRCLE(XA1, YA1), 2, 3
90 LINE(XA, YA)-(XA1, YA1), 5
100 END
```

Выполните программу на ЭВМ, введя конкретные координаты: X_O, Y_O, X_A, Y_A.

а) Какие изменения следует внести в программу, чтобы можно было построить и изобразить треугольник $A_1B_1C_1$, симметричный данному относительно точки O? Закрасьте построенный треугольник.

б) Измените программу так, чтобы можно было получить на экране изображение треугольника $A_1B_2C_2$, симметричного данному относительно вершины A.

2. Составьте программу для решения задачи: основание треугольника ABC лежит на оси x. Вершины A и C имеют координаты $A\,(X_A,\ O)$ и $C\,(X_C,\ O)$. Вершина B лежит в первой четверти плоскости xOy. Найдите координаты точки $D\,(X_D,\ O)$, в которой биссектриса BD пересечет основание треугольника AC.

Р е ш е н и е. Известно, что биссектриса делит противоположную сторону треугольника на отрезки, длины которых пропорциональны длинам прилежащих сторон треугольника:

$$|AD|:|DC| = |AB|:|BC|;\ |AD| = X_D - X_A;\ |DC| = X_C - X_D;$$

$$|AB| = \sqrt{(X_B - X_A)^2 + (Y_B - Y_A)^2};\ |BC| = \sqrt{(X_C - X_B)^2 + (Y_C - Y_B)^2};$$

$$\frac{X_D - X_A}{X_C - X_D} = \frac{\sqrt{(X_B - X_A)^2 + (Y_B - Y_A)^2}}{\sqrt{(X_B - X_C)^2 + (Y_B - Y_C)^2}}.$$

Предлагается программа:

```
10 '-------------- БИССЕКТРИСА -------------------------
20 CLS
30 PRINT "ВВЕДИТЕ КООРДИНАТЫ ВЕРШИН ТР-КА"
40 INPUT XA, YA, XB, YB, XC, YC
50 '----------- ВЫЧИСЛЕНИЕ КООРДИНАТ D(X, Y) ------------
60 R=SQR((XA-XB)^2+YB^2)/SQR((XC-XB^2+YB^2)
70 XD=(R*XC+XA)/(1+R)
80 PRINT "XD="; XD: PRINT "YD="; 0
90 FOR I=1 TO 5000: NEXT I
100 '--------------- РИСУНОК К ЗАДАЧЕ -------------------
110 CLS: SCREEN 3: COLOR 1, 3, 5
120 LINE(XA, YA)-(XB, YB), 3
130 LINE(XA, YA)-(XC, YC), 3
140 LINE(XB, YB)-(XC, YC), 3
150 LINE(XB, YB)-(XD, YD), 3
160 END
```

а) Проверьте программу в работе, введя конкретные координаты.

3. Составьте программу проведения биссектрисы AE.

а) Предложите план работы, если требуется провести биссектрису угла, образованного двумя прямыми, уравнения которых известны.

4. Дана программа построения вписанных в окружность правильных многоугольников:

```
10 '---------------- ПРАВИЛЬНЫЕ М-КИ -------------------
20 INPUT N, R, XO, YO
30 DIM X(N+1), Y(N+1)
40 FOR I=0 TO N
50 Y(I)=R*SIN((6.28318/N)*I)+YO
60 X(I)=R*COS((6.28318/N)*I)+XO
70 NEXT I
80 CLS: SCREEN 3: COLOR 1, 15, 3
85 CIRCLE(XO, YO), R, 5
90 X(N+1)=X(1): Y(N+1)=Y(1)
100 FOR I=1 TO N
110 LINE(X(I), Y(I))-(X(I+1), Y(I+1)), 5
120 NEXT I
130 END
```

Выполните программу на ЭВМ.

а) Постройте вписанный квадрат и вычислите его периметр. Изобразите окружность и вписанный квадрат.

б) Впишите в окружность сначала правильный треугольник, затем шестиугольник и вычислите площадь треугольника.

Формирование таблиц

1. Дана программа формирования квадратной таблицы из N^2 первых натуральных чисел. Числа заполняют «спирально» (по часовой стрелке) таблицу в $(N \times N)$ клеток:

```
10 CLS
20 'ПОСТРОЕНИЕ ЧИСЛОВОЙ СПИРАЛИ
30 INPUT "ВВЕДИТЕ N (<10)"; N
40 DIM A(N, N)
50 F=N: M=N: C=1: R=N
60 FOR I=1 TO N: Q=Q+1: K=I: A(1, K)=Q: NEXT I
70 R=R-1: IF R=0 THEN 160
80 FOR I=1 TO R: Q=Q+1: S=C+I: A(S, K)=Q: NEXT I
90 N=N-1: IF N=0 THEN 160
100 FOR I=1 TO N: Q=Q+1: K=F+1-C-I: A(S, K)=Q NEXT I
110 R=R-1: IF R=0 THEN 160
120 FOR I=1 TO R: Q=Q+1: S=M+1-C-I: A(S, K)=Q: NEXT I
130 N=N-1: IF N=0 THEN 160
140 FOR I=1 TO N: Q=Q+1: K=C+I: A(S, K)=Q: NEXT I
150 C=C+1: GOTO 70
160 FOR S=1 TO M
170 FOR K=1 TO F
180 IF A(S, K)<10 THEN PRINT A(S, K) " "; GOTO 200
190 PRINT A(S, K);
```

```
200 NEXT K
210 PRINT
220 NEXT S
```

Проверьте программу в работе, введя конкретное значение (например, $N=4$ или $N=8$).

а) Опишите суть алгоритма построения таблицы своими словами. В чем главная трудность формирования таблицы?

б) Измените текст программы так, чтобы таблица была заполнена по спирали, но против часовой стрелки.

Реализация процесса простейших игр

Условия игры. К началу игры имеется K групп предметов, содержащих соответственно M_1, M_2, ..., M_k предметов. Играют двое: человек и ЭВМ, ходят поочередно. За каждый ход любой из соперников может разбить любую одну группу на две меньшие.

Ходы выполняются до тех пор, пока во всех группах не останется по одному предмету.

Победителем считается тот, кто сумеет выполнить последний ход.

Разработайте алгоритм ведения игры. Вычертите его схему. Составьте программу. Выполните программу на ЭВМ.

Рекомендация. Составляя алгоритм, сформулируйте сначала ответы на вопросы: каким по очереди вступать в игру: первым или вторым? Зависит ли это от числа К и чисел М? Как рассчитывать свой ход?

§ 46. ИЗБРАННЫЕ ЗАДАЧИ

За многие годы в школьной математике постепенно сформировался «золотой фонд» учебных задач и задач для факультативных занятий. В этот фонд входят задачи по всем разделам школьной математики. Широко известны, например, сборники старинных задач по арифметике, алгебре, геометрии и тригонометрии. Использование старинных задач разных народов и эпох вызывает интерес к математике, побуждает школьников к самостоятельной творческой работе, к проявлению смекалки и инициативе.

В поле зрения учителя сегодня задачи, требующие не только математических знаний, но и умение поставить задачу для ее решения на ЭВМ. В наше время в связи с появлением в школе нового предмета возникает потребность в создании и отборе хороших учебных задач для работы с сильными учениками. Требуются задачи по технике программирования, т. е. задачи, в решении которых школьник овладевает специфическими приемами работы с ЭВМ. Требуются задачи, решение которых без применения ЭВМ неосуществимо, нужны и занимательные задачи.

Можно предположить следующую рубрикацию в классификации задач по курсу «Основы информатики и вычислительной техники»:

I. Информация

Задачи на подсчет количества информации, на закрепление понятия «единицы информации».

Задачи на кодирование информации. Кодирование числовой и символьной информации.

Задачи на преобразование чисел, записанных в одной системе счисления, на равные им в других системах счисления.

Задачи об особенностях представления чисел в различных системах счисления.

II. Алгоритмы

Задачи на разработку численных алгоритмов; задачи, требующие создания арифметической или алгебраической модели.

Решение уравнений и систем уравнений.

Задачи на преобразование символьной информации, алгоритмы обработки слов и текстов.

Лингвистические задачи, решаемые на ЭВМ.

Задачи на совместное использование средств символьной и численной обработки.

Задачи аналитической обработки алгебраических выражений.

Задачи алгебры многочленов.

Исследование функций (простейшие случаи).

Задачи на использование графических средств.

Геометрические инструменты на экране ЭВМ.

Геометрические задачи на построение и их решение на ЭВМ.

Метод координат — задачи из различных предметных областей.

Задачи на формирование таблиц различной формы.

Задачи сортировки.

Логические задачи и задачи математической логики.

III. Преобразователи информации

Задачи анализа простейших схем логических устройств.

Задачи синтеза простейших одноактных логических устройств.

Задачи синтеза простейших устройств, снабженных памятью.

Задачи на программирование для машины Поста.

Задачи на программирование для алгоритмической системы Тьюринга.

Задачи на разработку марковских алгоритмов.

Задачи на разработку алгоритмов в системах с заданным ограничением в средствах.

Задачи на взаимосвязь между алгоритмическими системами.

IV. Занимательные задачи

Задача об удвоении куба и ее решение на ЭВМ.

Задача о трисекции угла и ее решение с помощью ЭВМ.

Задача о квадратуре круга и ее решение на ЭВМ.

Моделирование процесса игры и программирование беспроигрышных стратегий на ЭВМ.

Вычисление замечательных математических констант с большим числом цифр в записи числа.

Решение задач на построение — пример: задача Аполлония.

Моделирование на ЭВМ генераторов чисел Пифагора, Фибоначчи, Каталана, Бернулли, Ферма, Мерсенна.

Построение треугольника Паскаля.

Построение гармонического треугольника Лейбница.

Моделирование решета Эратосфена.

Моделирование машины Поста на ЭВМ.

Моделирование машины Тьюринга на ЭВМ.

Замечательные кривые в полярных координатах.

Предлагаемая рубрикация рассматривается только как вспомогательная, приводимая в помощь учителю. Задания имеют уровень, несколько более высокий, чем задачи учебно-тренировочные. Все эти задачи всего лишь возможные образцы, нужно самостоятельно формировать свой «золотой фонд» задач и заданий.

1. Составить программу преобразования целого положительного числа, записанного в системе с основанием $q=10$, в равное ему число в системе с основаниями $p>10$, $p<10$.

2. Составить программу преобразования целого десятичного числа в равное ему число в системе счисления с основанием p, где p — целое отрицательное число ($p=-2$, $p=-3$...).

3. Составить программу преобразования целого числа в десятичной системе на равное ему число в уравновешенной системе счисления с основаниями $p=3$, $p=5$...

4. Составить программу преобразования десятичного целого положительного числа в равное ему число в системе остаточных классов с модулями $p_1=2$, $p_2=3$, $p_3=5$, $p_4=7$, $p_5=11$, $p_6=13$.

5. Составить программу преобразования целого положительного десятичного числа в равное ему число в системе с основанием $p=20$.

6. Составить программу преобразования целого положительного числа A_q в равное ему число B_p, где p и q — наперед заданные натуральные числа.

7. Составить программу преобразования данного числа в системе с целым отрицательным основанием A_p в равное ему число B_q, где $q \in N$.

8. Составить программу преобразования числа, записанного в системе остаточных классов с модулями $p_1=2$, $p_2=3$, $p_3=5$, $p_4=7$, $p_5=11$, $p_6=13$, в равное ему число B_q в системе счисления с основанием $q \in N$.

9. Составить программу преобразования числа, данного в системе с основанием $q=2i$, в равное ему десятичное число.

10. Составить программу-сумматор натуральных чисел, записанных в системе счисления с основанием $q \in N$ ($q<10$, $q>10$).

11. Составить программу-сумматор чисел, если суммирование цифр задается таблицами:

1.

+	0	1
0	0	1
1	1	110

2.

+	0	1	2	3
0	0	1	2	3
1	1	2	3	130
2	2	3	130	131
3	3	130	131	132

12. Составить программу-сумматор натуральных чисел, записанных в системе счисления с основанием $q=12$.

13. Составить программу-сумматор натуральных чисел, записанных в системе счисления остаточных классов с модулями $p_1=2$, $p_2=3$, $p_3=5$, $p_4=7$, $p_5=11$, $p_6=13$.

14. Составить программу-сумматор натуральных чисел, если суммирование цифр задается таблицей:

+	0	1	$\bar{1}$
0	0	1	1
1	1	$1\bar{1}$	0
$\bar{1}$	$\bar{1}$	0	$\bar{1}1$

15. Составить программу формирования n-го члена арифметической прогрессии, если a — первый ее член; d — разность.

16. Составить программу вычисления n-го члена геометрической прогрессии, если b — ее первый член; q — знаменатель.

17. Составить программу суммирования n первых членов арифметической прогрессии, если a — ее первый член; d — разность.

18. Составить программу суммирования n первых членов геометрической прогрессии, если b — ее первый член; q — ее знаменатель.

19. Составить программу суммирования n первых членов последовательности с общим членом $a_n=kn+l$, а также $a_n=k^n \cdot l$.

20. Составить программу суммировния членов последовательности, задаваемой формулой общего члена: $a_n=2n^2+n-1$. Суммироваться должны члены начиная с номера m по номер k включительно.

21. Составить программу-генератор чисел Фибоначчи.

22. Составить программу-генератор чисел Каталана.

23. Составить программу-генератор чисел, входящих в треугольник Паскаля.

24. Составить программу-генератор чисел Бернулли.

25. Составить программу «Решето Эратосфена» для получения простых чисел.

26. Составить программу-генератор чисел Ферма.

27. Составить программу-генератор чисел Мерсенна.

28. Составить программу-генератор чисел, входящих в треугольник Лейбница.

29. Составить программу-генератор коэффициентов в разложении степеней трехчлена $(1+x+x^2)^n$. Сформулировать правило формирования коэффициентов, удобное для их вычисления.

30. Составить программу разложения натурального числа на простые множители.

31. Составить программу сокращения дроби вида m/n, где $m \in N$; $n \in N$.

32. Составить программу приведения дробей к общему знаменателю.

33. Составить программу получения всех делителей данного натурального числа (простых и составных).

34. Составить программу вычисления значений многочлена

$$a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0.$$

35. Составить программу вычисления значений выражения

$$\frac{a_n}{x^n} + \frac{a_{n-1}}{x^{n-1}} + \ldots + \frac{a_1}{x_1} + a_0.$$

36. Составить программу вычисления суммы

$$1! + 2! + 3! + 4! + 5! + \ldots + n!.$$

37. Составить программу вычисления суммы

$$\frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \ldots + \frac{1}{n!}.$$

38. Составить программу для ответа на вопрос: делится ли нацело данное число a на число k.

39. Составить программу для ответа на вопрос: является ли данное число вида $2^k - 1$ числом вида $2^k + 1$.

40. Сумма первых n членов числовой последовательности, задаваемой общим членом $a_n = f(n)$, равна S. Найти количество членов последовательности, входящих в сумму.

41. Факториал числа x равен p. Найти число n сомножителей, данного p.

42. Произведение n первых нечетных чисел равно p. Сколько сомножителей взято?

43. Составить программу вывода на экран самой большой цифры в записи данного целого десятичного числа.

44. Число N_{10} содержит нечетное количество цифр. Вывести на печать среднюю цифру.

45. Составить программу для определения количества цифр в записи данного десятичного целого числа.

46. Составить программу определения количества цифр в записи данного целого числа, записанного в системе счисления с основанием p.

47. Составить программу-сумматор двух натуральных чисел, если количество цифр в каждом из них велико.

48. Составить программу вычисления степеней чисел вида a^n, если a и n велики.

49. Составить программу вычисления числа $2^{64}-1$, в результате сохранить все цифры.

50. Составить программу вычисления 100!.

51. Составить программу извлечения точного квадратного корня из n-разрядного числа (n велико).

52. Составить программу для проверки утверждения: «Результатами вычислений по формуле x^2+x+17 при $0 \leqslant x \leqslant 15$ являются простые числа». Все результаты вывести на экран.

53. Составить программу для проверки утверждения: «Результатами вычислений по формуле x^2+x+41 при $0 \leqslant x \leqslant 40$ являются простые числа».

54. Составить программу-генератор простых чисел, в основу положить формулу $2x^2+29$ при $0 \leqslant x \leqslant 28$.

55. Составить программу-генератор простых чисел, в основу положить формулу $(2^{2x+1}+1)/3$ при $1 \leqslant x \leqslant 36$.

56. Составить программу-генератор чисел Пифагора, т. е. чисел, удовлетворяющих условию: $a^2+b^2=c^2$; в основу положить формулы $a=2n+1$; $b=2n^2+2n$; $c=2n^2+2n+1$, $n \in N$, и вводить вручную.

57. Составить программу-генератор чисел Пифагора, используя $a=m^2-n^2$; $b=2m \cdot n$; $c=m^2+n^2$, где $m, n \in N$. Результаты вывести на экран в виде таблицы из пяти столбцов: m, n, a, b, c.

58. В данном натуральном n-разрядном числе N_{10} переставить цифры так, чтобы образовалось наибольшее число, записанное этими же цифрами.

59. В данном натуральном n-разрядном числе N_{10} переставить цифры так, чтобы образовалось наименьшее число, записанное этими же цифрами.

60. Составить программу умножения чисел, имеющих в своей записи большое количество цифр.

61. Составить программу, работая по которой ЭВМ восстановит задуманное число $x<100$, если ей сообщат остатки от деления x на 3,5 и 7.

62. Составить программу вывода на экран заполненной таблицы истинности для заданного логического выражения

$$X=F\ (A,\ B,\ C).$$

63. Составить программу для проверки, являются ли тавтологиями нижеприводимые высказывания:

$$A+BC \equiv (A+B)(A+C),\ A \to BC \equiv (A \to B)(A \to C);$$
$$A+(B \to C) \equiv (A+B) \to (A+C),\ A \to (B+C) \equiv (A \to B)+(A \to C);$$

$$A(B \to C) \equiv AB \to AC,\ A \to (B \sim C) \equiv (A \to B) \sim (A \to C);$$
$$A(B \sim C) = AB \sim AC,\ A + (B \sim C) = (A + B) \sim (A + C);$$
$$\overline{A \to B} = \bar{A} \to \bar{B}.$$

64. Составить программу отыскания среди вводимых вручную высказываний тождественно ложных и тождественно истинных.

65. Составить программу выяснения, являются ли два или несколько данных высказываний эквивалентными.

66. Составить программу для решения логической задачи: в соревнованиях по гимнастике участвуют Алла, Валя, Сима и Даша. Болельщики высказали предположения о возможных победителях:

первый будет Сима, Валя будет второй;

второй будет Сима, Даша — третьей;

Алла будет второй, Даша — четвертой.

По окончании соревнований оказалось, что в каждом из предположений только одно из высказываний истинно, другое ложно.

Какое место на соревнованиях заняла каждая из девушек, если все они оказались на разных местах?

67. Можно ли по программе, поставленной для решения задачи 66, решить задачу, в которой речь идет о четырех действующих лицах? Что требуется изменить в тексте программы?

68. Решить задачу, если условие задано системой высказываний:

$$\bar{Л}_1 \to K_1 = 1; \quad A_2 \to \bar{K}_1 = 1; \quad Л_1 K_1 = 0; \quad Л_1 Л_2 = 0;$$
$$\bar{K}_1 \bar{K}_4 \to A_1 = 1; \quad \bar{A}_1 \to \bar{M}_2 = 1; \quad Л_2 K_2 = 0; \quad M_2 M_4 = 0.$$
$$\bar{M}_4 \to K_3 = 1;$$

Придумать условие задачи.

69. Решить логическую задачу, если ее условие задано системой высказываний:

$$A_1 \bar{B}_2 + \bar{A}_1 B_2 = 1; \quad A_1 A_2 = 0; \quad G_3 G_4 = 0.$$
$$A_2 \bar{G}_3 + \bar{A}_2 G_3 = 1; \quad A_2 B_2 = 0;$$
$$C_2 \bar{G}_4 + \bar{G}_2 G_4 = 1; \quad C_2 B_2 = 0;$$
$$C_2 A_2 = 0;$$

70. На математической олимпиаде за три первых места вручены призы победителям. На вопрос, кто оказался призером, получены такие ответы:

1. Константин, Дмитрий и Светлана.
2. Элиза, Владимир и Дмитрий.
3. Григорий, Максим и Богдан.
4. Светлана, Жанна и Дмитрий.
5. Григорий, Жанна и Богдан.
6. Элиза, Максим и Павел.
7. Владимир, Павел и Элиза.

Известно, что в одном ответе все три имени названы правильно; в одном ответе правильно названо только одно имя; в двух ответах правильно названы два имени; в остальных трех все три имени названы неправильно. Составить программу для отыскания трех победителей олимпиады.

71. Составить программу формирования массива слов A из N слов.

72. Составить программу сортировки массива A$ по длине слов, входящих в него (сначала в массиве A$ после сортировки должны быть более короткие слова).

73. Составить программу сортировки слов массива A$ по первым буквам слов в лексикографическом порядке.

74. Составить программу выяснения, является ли введенное слово A$ палиндромом, т. е. словом, которое читается одинаково как слева направо, так и справа налево.

75. Дан массив символов A$, упорядоченный в лексикографическом порядке. Вставить в массив новый элемент так, чтобы лексикографический порядок сохранился.

76. В массиве A$ из N слов найти все слова, заканчивающиеся трехбуквенным словом B$.

77. Выяснить число вхождений слова A$ в данное слово B$.

78. Записаны слова в столбик и рядом вписать те же слова, в обратном порядке.

79. Составить программу преобразования десятичных натуральных чисел в римскую систему нумерации.

80. Составить программу преобразования чисел, записанных в римской нумерации, в числа десятичной системы.

81. Найти количество слов заданной длины в массиве данного слова A.

82. Составить программу приведения подобных членов в данном многочлене от одного переменного (исходный многочлен: $3x^3-2x^2+x-2x^3+x-6+3x$, результат: x^3-2x^2+5x-6).

83. Составить программу умножения одночлена на многочлен (пример: $2x^2\cdot(x^3-3x^2-6)=2x^5-6x^4-12x^2$).

84. Составить программу умножения многочленов от одного переменного (пример: $(2x^4-x^3+x^2-3x+1)(x^3-2x^2)$).

85. Составить программу перемножения многочленов от двух переменных (пример: $(2x^3-3xy+y^2)(2x-y)$).

86. Составить программу вынесения общего буквенного множителя за скобки (пример: $2a^3-3a^2+a=a(2a^2-3a+1)$).

87. Составить программу возведения в степень выражений типа $(1+x)^n$.

88. Составить программу возведения в степень выражений типа $(1+x+x^2)^n$.

89. Составить программу возведения в степень выражений типа $(1+x+x^2+x^3)^n$.

90. Составить программу возведения в степень выражений типа $(1+x+x^2+\ldots+x^k)^n$.

91. Вычислить N значений функции $F(x)$ и построить график по точкам, аргумент x должен получать приращение Δx.

92. Составить программу сдвига графика вдоль оси x.

93. Составить программу сдвига построенного графика вдоль оси y.

94. Составить программу «Управляемый курсор».

95. Разработать программу «Управляемая с клавиатуры линейка на экране» (управление ведется через приращение координат концов отрезка, вводимых с клавиатуры).

96. Разработать программу «Управляемая с клавиатуры парабола» (параболопостроитель).

97. Сконструировать гиперболопостроитель.

98. Сконструировать эллипсопостроитель.

99. Составить программу построения прямоугольного треугольника по заданным длинам медиан, проведенных к катетам.

100. Составить программу формирования таблицы сложения цифр в системе счисления с основанием $q=4$.

101. Составить программу таблицы сложения цифр в системе счисления с основанием $q=-4$.

102. Составить программу формирования треугольника Паскаля (коэффициенты в разложении $(1+x)^n$).

103. Составить программу формирования треугольника из коэффициентов разложения: $(1+x+x^2)^n$.

104. Составить программу, работая по которой ЭВМ заполнит таблицу из $n \times n$ клеток последовательными числами, расположенными по спирали, начиная с левого верхнего угла и продвигаясь по часовой стрелке (против часовой стрелки).

105. На клетчатой доске $(N \times N)$ расставить максимальное количество ферзей «нового типа» так, чтобы они не мешали друг другу. Ферзь «нового типа» имеет право ходить не более чем на K клеток в каждом направлении. Ввод значений K и N — с клавиатуры.

106. В предыдущую задачу ввести изменения в правила хода. Ферзь «нового типа» может ходить по вертикали не более чем на K_1 клеток, по горизонтали — не более чем на K_2 клеток и по диагонали — не более чем на K_3 клеток.

107. Решить кроссворд, если известны слова, его образующие, но неизвестно, как они записываются (по горизонтали или вертикали). Слова для заполнения кроссворда: АЛГОЛ, АВОСТ, СЛОВО, КРОССВОРД.

108. На доске из $N \times N$ клеток расставить максимальное количество шахматных коней так, чтобы они не били друг друга.

109. Составить программу-генератор фигурных чисел. В основу положить расчетную формулу K-го члена n-угольного числа:

$(n)_k = k/2\,(n\,(k-1)-2\,(k-2))$, где $k=1, 2, \ldots; n=3, 4, \ldots$.

ЭВМ должна изображать любое сформированное число на экране.

110. Известно, что прямоугольник размером 32×33 можно составить из девяти квадратов со сторонами 1, 4, 7, 8, 9, 10, 14, 15, 18 единиц. Составить программу вычерчивания этого прямоугольника. Сколько решений нашлось?

111. На плоскости дано N точек. Составить программу построения выпуклого n-угольника, если это возможно.

112. Дана квадратная таблица. Необходимо разрезать ее на четыре части, из которых можно сложить уже магический квадрат. Сумма всех чисел по каждой из горизонталей, вертикалей и диагоналей должна быть равна 34.

1	15	5	12
8	10	4	9
11	6	16	2
14	3	13	7

113. На доске из 4×4 клеток расставлено 8 слонов. Задача состоит в том, чтобы поменять белых и черных слонов местами. При этом ни один из слонов не должен ни разу атаковать слона противоположного цвета. Первый ход делают белые. Составить демонстрационную программу.

114. Поместить ферзя на его собственную клетку шахматной доски (поле $d1$). Сделать им пять ходов так, чтобы его путь был максимальным и при движении ферзь ни разу не пересекал бы свой собственный путь. Составить демонстрационную программу.

115. На доске из 3×3 клеток в угловых клетках стоит по одному коню (из них вверху — два белых, внизу — два черных). Составить программу обмена местами всех коней. Программа должна быть демонстрационной.

116. Составить демонстрационные программы вывода на экран изображений кривых: кардиоиды, циклоиды, гипоциклоиды, улитки Паскаля, астроиды, дельтоиды, конхоиды Никомеда, спирали Архимеда, циссоиды Диоклеса, логарифмической спирали, лемнискаты Бернулли, многолепестковой розы.

117. Построить латинский квадрат (5×5).

118. Построить латинский квадрат $(n \times n)$.

119. Дано два ортогональных латинских квадрата:

1	2	3	4
3	4	1	2
4	3	2	1
2	1	4	3

и

1	2	3	4
4	3	2	1
2	1	4	3
3	4	1	1

Построить магический квадрат с $S=34$.

120. Прямоугольник размером 29×31 покрыть квадратами следующих размеров:

10×10 — 2 шт.;	11×11 — 1 шт.;
6×6 — 2 шт.;	19×19 — 1 шт.;
12×12 — 1 шт.;	1×1 — 1 шт.

Результат продемонстрировать на экране ЭВМ.

121. Даны три окружности различного радиуса, касающиеся друг друга. Провести четвертую окружность, касающуюся всех трех. (Рекомендация. Используйте условие: $a^2+b^2+c^2+d^2=\frac{1}{2}(a+b+c+d)^2$, где a, b, c — кривизны данных окружностей; d — кривизна искомой.)

122. Данной прямой касаются две окружности радиусами R_1 и R_2. Найти радиус окружности, которая касается данных окружностей и прямой: $ax+by+c=0$.

123. Построить спираль Архимеда: $\rho=a\cdot\varphi$. Составить программу построения спирали одного витка.

124. Располагая спиралью Архимеда, разделить данный угол на три части (задача о трисекции угла).

125. Располагая спиралью Архимеда, разделить данный угол на части в отношении $m:n:p$.

126. Найти наименьшее натуральное число, дающее при делении на 2 остаток 1, при делении на 3 остаток 2, при делении на 4 остаток 3 и т. д.

127. На доске из $N\times N$ клеток расставить максимальное количество ладей, мешающих друг другу ($N=10$).

128. На доске из $N\times N$ клеток расставить максимальное количество слонов, не мешающих друг другу ($N\leqslant10$).

129. На доске из $N\times N$ клеток расставить максимальное количество ладей «нового типа», когда каждая ладья может ходить только по K клеток в любом допустимом направлении ($N\leqslant10$).

130. Составить программу построения обратной числовой спирали. Дано n^2 первых натуральных чисел. Построить квадратную таблицу, в центре которой расположено число 1, в клетках, соседних по спирали,— следующие цифры. Пример для $n=3$ приведен рядом (спираль образована движением против часовой стрелки).

5	4	3
6	1	2
7	8	9

131. Из русских слов ПАК, КОНСУЛ, ОТГУЛ, КОМАНДА, ПАМЯТЬ сформировать кроссворд. Каждое из приведенных слов

должно не менее двух раз пересечься с другим из этих же слов.

132. Из данных 10 слов составить чайнворд, каждое из слов встречается в чайнворде по одному разу.

133. Из слов АРГЕНТИНА, НЕГРА, МАНИТ составить слово ПАЛИНДРОМ. Данные слова следует объединить в одно с помощью операции конкатенации.

134. Составить программу формирования всех возможных кроссвордов из следующих четырех слов: по горизонтали — СЛОВО, КРОССВОРД; по вертикали — АЛГОЛ, АВОСТ.

135. Из слов КРОНА, КВАНТ, АНАПА, ТОЛПА, КОРАБЛЬ, КАРАВАН составить всевозможные кроссворды. Каждое слово должно не менее двух раз пересечься с другими данными словами.

136. Разработать программу, реализующую процесс игры между человеком и ЭВМ.

Условия игры: имеется N групп предметов, содержащих m_i предметов в каждой. Играют двое, ходят по очереди. Задача каждого из соперников — на каждом очередном ходе разбить одну любую группу на две меньшие. Игра продолжается до тех пор, пока во всех группах не останется по одному предмету. Победителем останется тот, кто сумеет сделать последний ход. В основу игры ЭВМ положить алгоритм, основанный на ответах на вопросы: каким по очереди вступать в игру? Как осуществлять очередной ход?

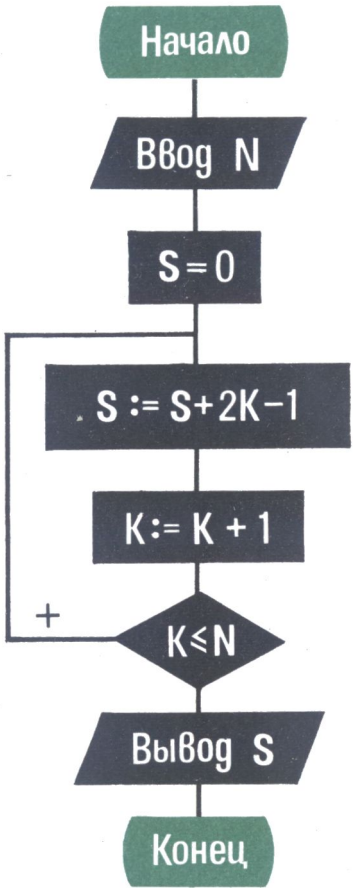
137. К началу игры имеется k предметов. Первая группа содержит m_1 предметов, вторая — m_2 предметов, третья — m_3 предметов, ..., k-я группа — m_k предметов. Играют двое. Ход любого состоит в раздроблении каждой группы, состоящей более чем из одного предмета, на две меньшие. Ходы выполняются до тех пор, пока во всех группах не останется по одному предмету. Победителем считается тот, кому удастся сделать последнее разбиение.

138. **Игра Гранди.** К началу игры имеется одна общая группа, содержащая N предметов. Играют двое, ходят поочередно. Выполнить ход — это значит разбить любую одну из $k \geqslant 1$ групп предметов на две неравные части. Игра продолжается до тех пор, пока все группы не будут состоять из одного-двух предметов. Победителем считается тот, кто выполнит последнее разбиение.

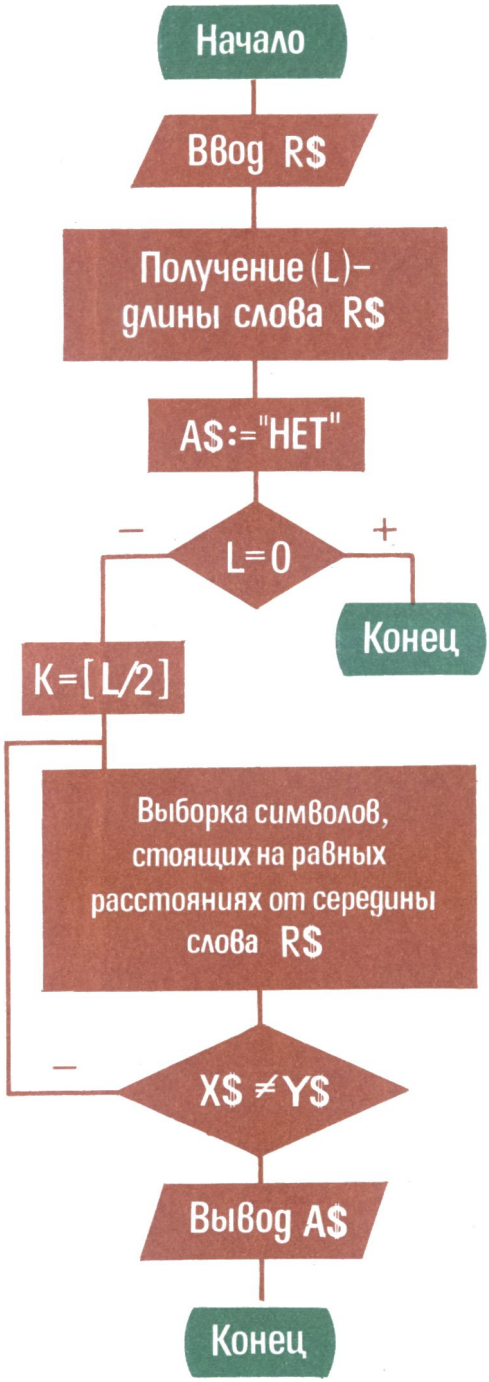
139. Разработать программу для беспроигрышного участия ЭВМ в игре, проводимой по условиям: к началу игры в коробке имеется некоторое (N) количество горошин. Два игрока (ЭВМ и человек) поочередно извлекают из коробки любое количество горошин, не превышающее половины имеющихся в ней. Проигрывает тот, кто берет последнюю горошину.

140. В предыдущей игре меняют условия определения победителей: выигрывает тот, кто берет последнюю горошину.

ОГЛАВЛЕНИЕ




```
 5  INPUT N
10  FOR K=1 TO N
15  S=S+(2K-1)
20  NEXT
25  PRINT S
30  END
```




```
20  INPUT R$
30  L=LEN(R$)
40  A$ ="НЕТ"
50  IF L=0 THEN END
60  K= INT (L/2)
70  FOR I=1 TO K
80 X$ =MID$(R$,I,1)
90 Y$=MID$(R$,L-I,1)
100 IFX$<>Y$ THEN 130
110  NEXT I
120  A$ ="ДА"
130  PRINT A$
140  GO TO 20
150  END
```

2 р. 10 к.