ОСНОВЫ ИНФОРМАТИКИ И ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

ПРОБНОЕ УЧЕБНОЕ ПОСОБИЕ
для средних учебных заведений

В двух частях

•

ЧАСТЬ ПЕРВАЯ

Под редакцией
А. П. Ершова и В. М. Монахова

Рекомендовано
Главным управлением школ
Министерства просвещения СССР

МОСКВА
«ПРОСВЕЩЕНИЕ» 1985

ББК 73я72
0-75

А. П. ЕРШОВ, В. М. МОНАХОВ, С. А. БЕШЕНКОВ,
Я. Э. ГОЛЬЦ, А. А. КУЗНЕЦОВ, Э. И. КУЗНЕЦОВ,
М. П. ЛАПЧИК, Д. О. СМЕКАЛИН

Основы информатики и вычислительной техники: Проб.

0-75 учеб. пособие для сред. учеб. заведений. В 2-х ч. Ч. 1/

А. П. Ершов, В. М. Монахов, С. А. Бешенков и др.; Под ред.

А. П. Ершова, В. М. Монахова.— М.: Просвещение, 1985.—

96 с, ил.

В этом учебном пособии даны первоначальные сведения об ЭВМ, их устройстве и применении, о подготовке решения задач на ЭВМ с помощью алгоритмов. В конце каждого параграфа приводятся вопросы для повторения теории и упражнения. Пособие рассчитано для использования учащимися средних специальных учебных заведений, профессионально-технических училищ, средней общеобразовательной школы.

4306020400-516
103(03)-85 **инф. письмо**

ББК 73я72 + 32.973я72
001.8(075)+ 6Ф7.3(075)

ВВЕДЕНИЕ

РОЛЬ ЭВМ В СОВРЕМЕННОМ ОБЩЕСТВЕ

Мы с вами начинаем изучать основы информатики и вычислительной техники. Информатика исследует законы и методы переработки и накопления информации. Эта наука появилась недавно — во второй половине XX в. Ее развитие связано с появлением электронно-вычислительных машин (ЭВМ), мощных универсальных устройств для хранения и переработки информации. ЭВМ называют также компьютерами (от английского слова computer — вычислитель).

Потребность выразить и запомнить информацию привела к появлению речи, письменности и изобразительного искусства. В дальнейшем необходимость передачи и распространения информации вызвала к жизни книгопечатание, почтовую связь. Появление телеграфа, телефона, радио и телевидения позволило передавать огромные потоки текстовой информации и изображений со скоростью света. Однако целенаправленная обработка этой информации до самого последнего времени проводилась только человеком.

Использование ЭВМ позволяет теперь переложить часть этой обработки на автоматические устройства, способные достаточно долго работать без участия человека и со скоростью, в несколько миллионов раз превышающей скорость обработки информации человеком.

Универсальность ЭВМ, ее способность к целенаправленной переработке различных видов информации и объясняют происходящий сейчас стремительный процесс внедрения ЭВМ в самые разные сферы деятельности человека в современном обществе.

Внедрение ЭВМ приводит к коренной перестройке технологии производства во многих отраслях народного хозяйства, повышению производительности и улучшению условий труда людей. Разумеется, внедрение вычислительной техники невозможно без обучения людей, которые будут ее использовать (их часто называют пользователями ЭВМ). Это обучение начинается в школе. В начальной школе вы научились читать и писать — стали грамотными. В IX и X классах вы постигнете азы второй грамотности — компьютерной.

Область применений ЭВМ чрезвычайно широка.

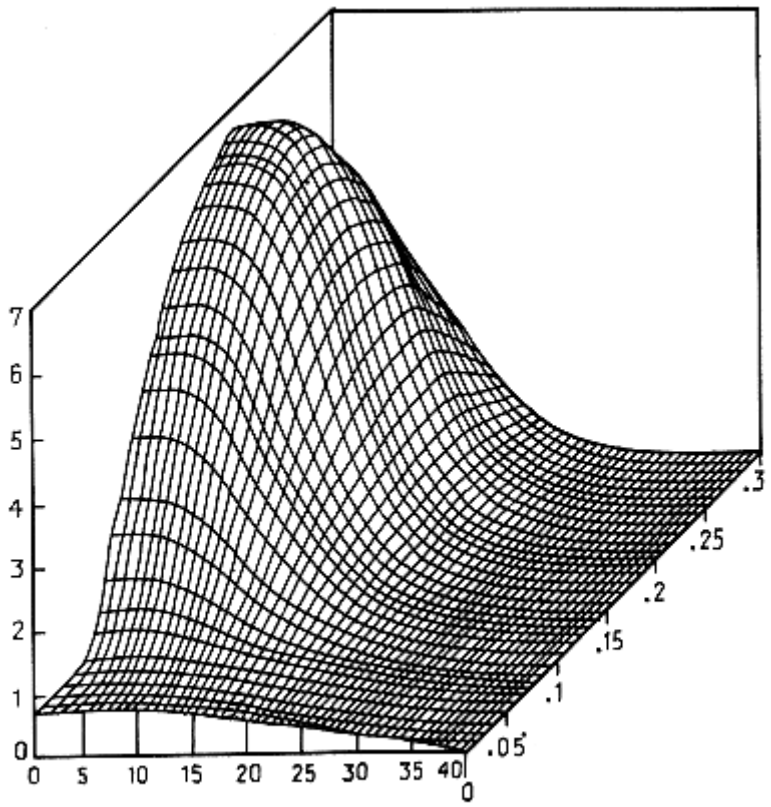

Рис. 1. Результаты математического моделирования плазмы в термоядерном реакторе

Важнейшим средством современного научного исследования является математическое моделирование физических явлений и исследование этих моделей с помощью ЭВМ. Современные компьютеры позволяют, например, проводить сложнейшие расчеты при создании установок термоядерного синтеза или сверхзвуковых самолетов (рис. 1).

Без экспериментов невозможен научно-технический прогресс. Эксперименты с термоядерными реакторами, аэродинамическими трубами, ускорителями очень сложны и дорогостоящи. ЭВМ позволяют заменить реальные эксперименты в тысячи раз более дешевыми машинными, вычислительными экспериментами. Кроме того, такие вычислительные эксперименты часто можно проводить во много раз быстрее, чем настоящие. Наконец, в некоторых областях науки, например в астрофизике, проведение реальных экспериментов и вовсе невозможно. Здесь на помощь исследователю приходит вычислительный эксперимент.

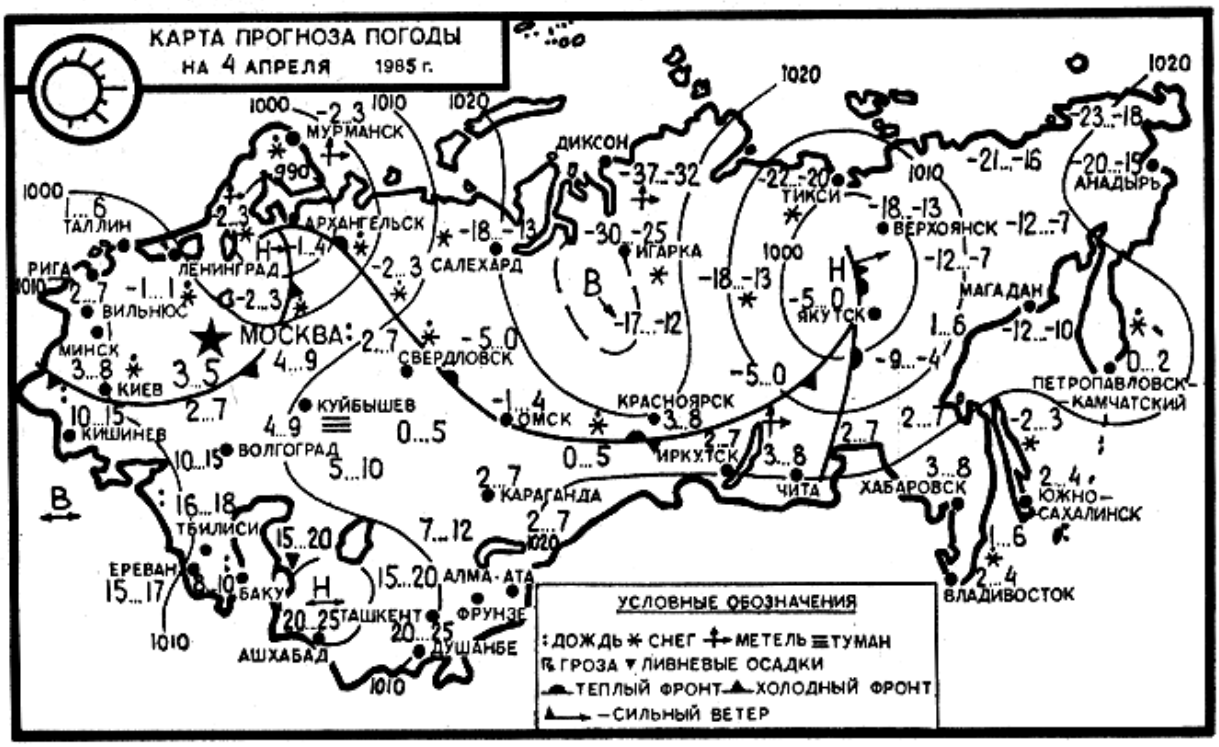

Рис. 2. Сообщает Гидрометцентр СССР

Одной из наиболее трудоемких вычислительных задач является задача прогноза погоды. Для ее решения необходимо собрать и проанализировать информацию, получаемую со спутников и метеостанций, выполнить огромный объем вычислений, необходимых для решения уравнений, возникающих при математическом моделировании процессов в атмосфере и океане, наконец, представить полученные результаты в удобной для человека форме. Все эти этапы немыслимы без применения ЭВМ (рис. 2).

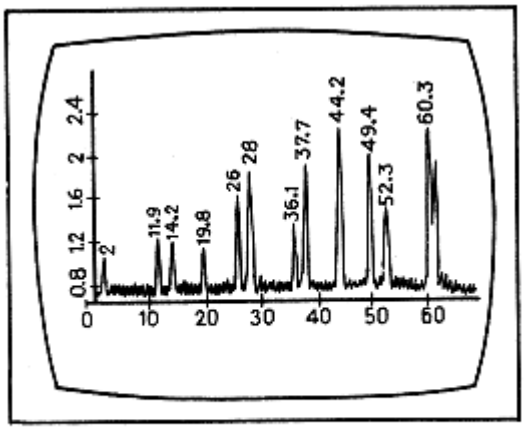

Рис. 3. Результаты математической обработки физического эксперимента на экране графического дисплея

Компьютеры нужны не только для проведения машинных экспериментов, но и для обработки результатов реальных экспериментов. Современный физический или биологический эксперимент часто дает столько информации, что обработать ее без ЭВМ практически невозможно (рис. 3).

Использование ЭВМ позволило в последние годы создать новый метод получения изображения внутренних частей непрозрачных тел. Этот метод называется томографией. Он позволяет получать изображение гораздо лучшего качества, чем рентгеноскопия. Томография основана на способности ЭВМ быстро выполнить сотни миллионов арифмети-

ческих действий над числами. Именно такое количество действий требуется в томографии для получения одного-единственного изображения.

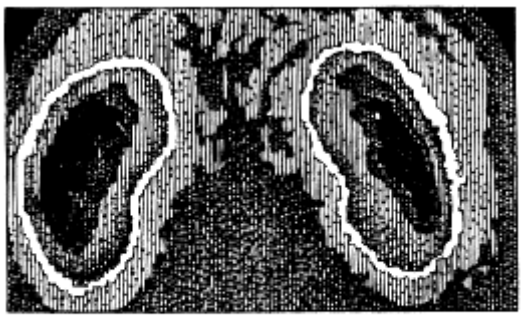

Рис. 4. Компьютерная томограмма, используемая при медицинской диагностике

Томография позволяет обнаружить дефекты, скрытые в глубине детали, или признаки заболевания, скрытые в тканях человеческого организма (рис. 4). ЭВМ способна и создавать изображения, в том числе движущиеся, «живущие». Мультфильмы, созданные компьютерами, уже используются для тренировки пилотов, водителей и т. д.

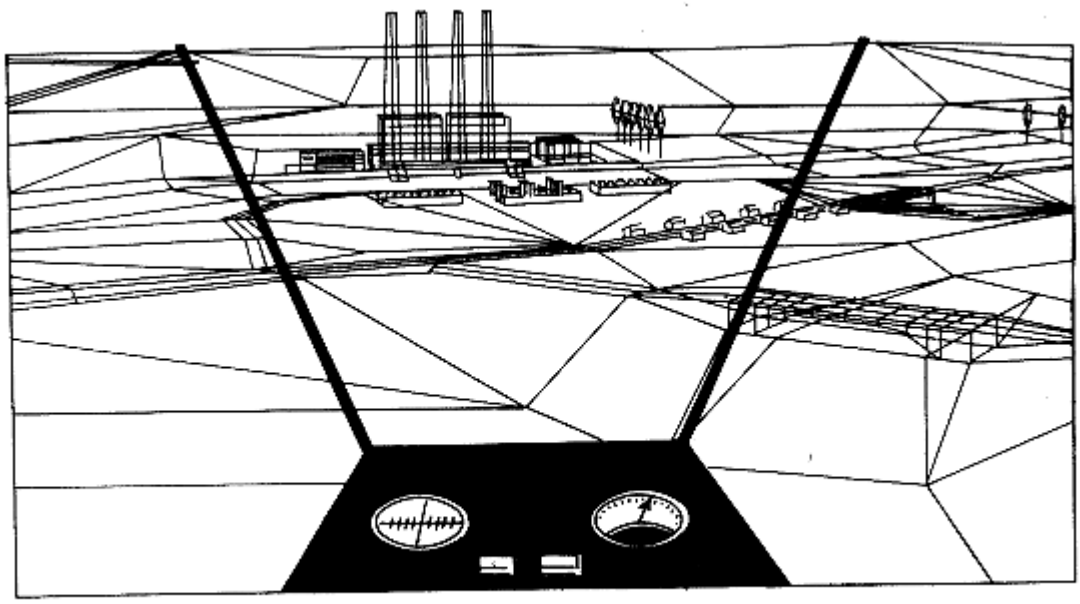

Рис. 5. В системе, обучающей пилота посадке самолета, используется изображение, созданное компьютером

Например, пилот может с помощью специального тренажера, управляемого ЭВМ, не покидая земли, отработать навыки управления новым самолетом или тренироваться в посадке на полосу нового аэропорта (рис.5).

Графические построения и расчеты с помощью ЭВМ различных машиностроительных, авиационных, автомобильных деталей и конструкций являются составной частью систем автоматизированного проектирования (сокращенно САПР). Такие системы во много раз повышают производительность труда конструктора и сокращают сроки разработки.

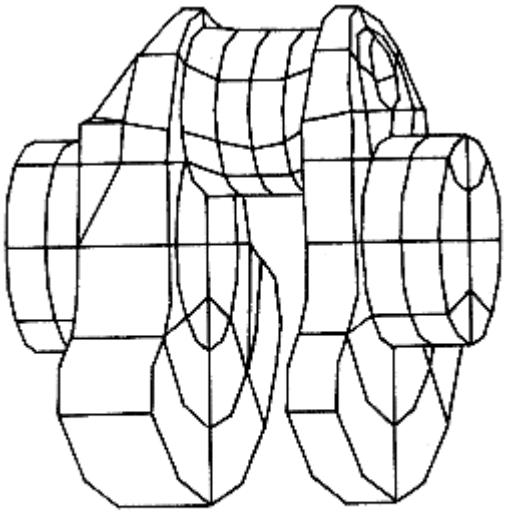

Рис. 6. Часть модели коленчатого вала автомобиля, построенная системой автоматизированного проектирования автомобилей

Сидя перед подключенным к ЭВМ экраном (подобным телевизионному), конструктор автомобиля, например, может скомандовать ЭВМ изобразить отдельные части и узлы автомобиля, что позволяет оценить внешний вид и компоновку (рис. 6). ЭВМ может также произвести моделирование работы этих узлов и показать их части, где наиболее вероятна поломка при эксплуатации.

Одной из важных областей применения компьютеров являются справочно-информационные системы. Возможно, вы видели на вокзале автоматическую справочную. Она имеет несколько десятков кнопок, около каждой из которых написан вопрос. Нажав на любую из них, можно прочесть ответ на соответствующий вопрос. Такая система может отвечать только на вопросы из заданного списка с помощью заранее подготовленных табличек с ответами. Бывают, однако, и более сложные задачи, которые система такого рода решить не в состоянии. Одна из таких задач, которую нельзя решить без компьютера, возникает при продаже авиабилетов. Эта задача может быть решена с помощью мощной ЭВМ, соединенной с большим числом специальных устройств — терминалов, за которыми работают кассиры. Информация о всех рейсах самолетов, об уже занятых и еще свободных местах на все эти рейсы хранится в памяти ЭВМ. Эта информация должна оперативно меняться при каждой покупке билета, при каждом назначении дополнительного рейса и т. д. С другой стороны, эта информация или некоторый результат ее переработки должны по требованию кассира быстро выдаваться на экран терминала. Наконец, когда пассажир выбрал нужный рейс или маршрут с пересадками, печатание соответствующего авиационного билета тоже должно проводиться автоматически (рис. 7). Со всеми этими задачами успешно справляется автоматизированная система продажи авиабилетов «Сирена-2», услугами которой вам еще доведется воспользоваться.

Рис. 7. Билет, выданный автоматизированной системой управления продажи авиабилетов «Сирена-2»

Информационные системы, построенные на базе ЭВМ, могут оказаться полезными не только при покупке авиабилетов, но и при решении многих других задач. Они могут помочь в разных исследованиях, например в химии. Если химик прошлого века мог помнить свойства интересующего его химического соединения или найти их в справочнике, то сегодня это немыслимо: исследовано громадное число различных химических веществ, описания которых заняли бы сотни томов. Найти описание конкретного вещества в этих томах еще можно, но найти описание вещества, обладающего заданными свойствами, или решить какую-нибудь другую задачу поиска нужной нам информации вручную уже невозможно. На помощь приходят компьютеры. С их помощью можно создать специальным образом организованное хранилище информации, так называемую «базу данных», содержащую сведения об известных к настоящему времени химических веществах. После организации «базы данных» компьютер может работать в качестве так называемой информационно-поисковой системы. Эта система способна практически мгновенно ответить химику на интересующие его вопросы: известно ли данное химическое вещество, какими свойствами оно обладает, какие другие химические вещества по свойствам наиболее близки к данному и т. д.

Информационно-поисковые системы находят широкое применение во многих областях науки. Например, они интенсивно используются в молекулярной биохимии при анализе структур генов, входящих в ДНК.

Выше мы уже говорили о системах автоматизированного проектирования, многократно повышающих производительность труда конструктора. Но после успешного окончания проектирования детали ее нужно еще изготовить. Процесс изготовления деталей на станках также можно автоматизировать с помощью ЭВМ. Станок, управляемый ЭВМ, называется станком с числовым программным управлением (ЧПУ). Для того чтобы обработать деталь на станке с ЧПУ, человек должен прежде всего написать программу обработки этой детали, т. е. описать последовательность элементарных операций, в результате выполнения которых деталь будет обработана. После того как эта программа будет написана и введена в память ЭВМ, станок с ЧПУ может обработать деталь автоматически, без участия человека. По одной и той же программе можно обработать серию одинаковых деталей. Для перехода к обработке другой детали нужно будет лишь сменить программу в памяти ЭВМ, управляющей станком.

В наше время значительная часть предназначенной для человека информации создается, передается и воспринимается в виде текстов. Тексты обычно пишутся или печатаются на бумаге, таковы книги, газеты, ваши классные и домашние задания, письма, деловая документация и т. д. Работы по составлению, написанию и оформлению текстов весьма трудоемки. Хорошо подготовленный и оформленный текст легко воспринимается челове-

ком, плохо подготовленный текст трудно понять. ЭВМ успешно используются для подготовки и корректирования (оформления) текстов. Стоит ввести текст в память ЭВМ, и. дальше исправление опечаток, деление на страницы, многократную перепечатку текста, составление указателей ЭВМ может взять на себя. Круг задач по обработке текстов, которые ЭВМ может помочь решить, весьма велик. Применение ЭВМ в этой области существенно повышает производительность труда.

Работа ЭВМ может происходить без непосредственного контакта с человеком. Специальные компьютеры — микропроцессоры могут встраиваться в самое различное оборудование и заменять человека при управлении этим оборудованием. Такой микропроцессор может, например, автоматически переключать скорости при движении автомобиля и выбирать наиболее экономичный режим работы двигателя.

Компьютер может быть соединен с другими компьютерами, тогда они образуют сеть ЭВМ. Такая сеть может охватывать компьютеры в одном здании, а может связывать компьютер с компьютерами других городов. Сети ЭВМ позволяют использовать информацию, накопленную в одном месте, сразу во многих местах, где она нужна.

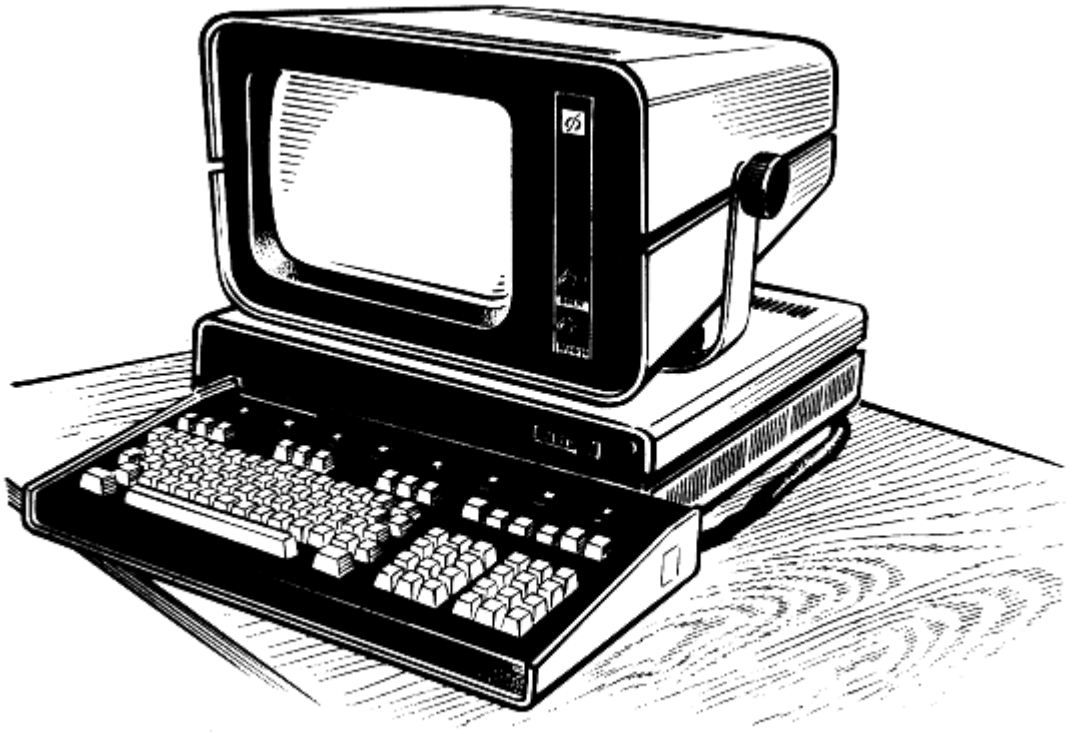

Рис. 8. Персональный компьютер

Прогресс в развитии ЭВМ сегодня привел к появлению новых ЭВМ — персональных компьютеров (рис. 8). Такие компьютеры помогут врачу поставить диагноз, школьнику выполнить уроки, учителю проверить домашние работы учащихся, архитектору спроектировать новые дома. Присоединенные к компьютерной сети персональные ЭВМ свяжут человека со справочными службами и библиотеками, позволяя ему быстро получить нужные сведения.

Поручая компьютерам механическую, рутинную работу, мы освобождаем человека для творческой деятельности. Для того чтобы ЭВМ могли решать нужные задачи, люди должны постоянно передавать компьютерам свои знания в виде точной информации, строгих правил, безошибочных алгоритмов и эффективных программ. Вот почему знание основ информатики и вычислительной техники, понимание их роли в жизни общества, деятельности людей становятся элементом человеческой культуры, составной частью общего образования, учебным предметом.

ПЕРВОНАЧАЛЬНЫЕ СВЕДЕНИЯ ОБ ЭВМ

Огромное разнообразие областей применения вычислительной техники потребовало создания ЭВМ разных типов — больших и малых, универсальных и специализированных.

Любая ЭВМ — сложнейшая техническая система, состоящая из миллионов и даже сотен миллионов простейших устройств — элементов. При создании современных вычислительных машин используется технология, опирающаяся на последние достижения многих отраслей науки и техники — информатики, математики, физики, химии, а также материаловедения, микроэлектроники и др.

Несмотря на разницу в размерах, внешнем виде, выполняемых функциях, различные ЭВМ имеют общую структуру и принципы работы. Эти принципы достаточно просты. Для того чтобы понять их, нужно познакомиться с назначением памяти, процессора и устройств ввода и вывода информации — основных компонентов, из которых состоит любая ЭВМ (рис. 9).

Рис. 9. Схема ЭВМ

Форма представления информации в ЭВМ. Сначала познакомимся с тем, в какой форме представляется информация в ЭВМ. ЭВМ может «хранить» и обрабатывать информацию в виде комбинации электрических сигналов двух типов, которые принято обозначать цифрами 0 и 1.

Любая информация представляется в ЭВМ последовательностью этих двух цифр 0 и 1, такие последовательности называются двоичными кодами. В большинстве ЭВМ один символ (т. е. буква, цифра, знак препинания или специальный знак §,.%.. и т. д.) записывается кодом из 8 цифр (такой код называется восьмиразрядным). Например, буква *А* кодируется как 01000001, а буква *М* — как 01001101.

Используя двоичные коды, можно закодировать (записать с помощью кода) любую информацию. Скажем, слово *мама* кодируется последовательностью из 32 цифр:

01001101 01000001 01001101 01000001.

Поскольку многие важнейшие применения ЭВМ связаны с обработкой числовой информации, то и числа кодируются последовательностями двоичных цифр, например число 1985—последовательностью из 16 цифр:

0000 0111 1100 0001

Изображение, которое мы видим на экране телевизора, также может быть закодировано последовательностью нулей и единиц, состоящей из сотен тысяч и даже миллионов цифр. Любое преобразование информации внутри ЭВМ сводится к работе с двоичными кодами.

Для измерения количества информации используется единица измерения бит. Один бит — это количество информации, содержащееся в сообщении типа «да — нет» (0 или 1 в двоичном коде). Восемь бит образуют более крупную единицу информации — байт.

Память ЭВМ состоит из отдельных ячеек памяти. В большинстве ЭВМ каждая такая ячейка способна хранить один байт информации. Для измерения объема памяти ЭВМ байт

— слишком мелкая единица, практичнее использовать производные единицы — килобайт (1 килобайт = 2^{10} байт=1024 байт), мегабайт (1 мегабайт = 2^{10} килобайт) и гигабайт (1 гигабайт = 2^{10} мегабайт). Эти производные единицы обозначаются кбайт, Мбайт и Гбайт соответственно.

Программный принцип работы ЭВМ. Процессор — центральное устройство ЭВМ, обрабатывающее информацию. Процессор может выполнять фиксированный набор операций над информацией, хранящейся в памяти ЭВМ. Каков этот набор, определяется устройством процессора. Количество таких операций не очень велико. Среди них арифметические действия (сложение, умножение, вычитание, деление) над числами, содержащимися в памяти, перемещение информации из одной ячейки памяти в другую и др.

Работа ЭВМ состоит в выполнении процессором заданной последовательности операций. Это выполнение происходит под управлением программы. Программа состоит из отдельных команд, предписывающих процессору выполнить то или иное действие над информацией, хранящейся в памяти. В этом и состоит программный принцип работы ЭВМ.

В каждой команде указывается, где именно в памяти находится нужная информация, какую именно следует выполнить операцию, в какое место памяти нужно поместить результат операции.

Важнейшими особенностями работы ЭВМ являются следующие:

1. Решение задачи по заданной программе происходит автоматически, без вмешательства человека.

2. Программа записывается в память машины и может быть заменена на другую программу.

Итак, задача процессора состоит в выполнении операций, задача памяти — в хранении обрабатываемой процессором информации и программы работы ЭВМ.

Устройства ввода-вывода. Информация, обрабатываемая процессором и хранящаяся в памяти, в некоторый момент должна быть помещена или, как говорят, введена в память. Результаты работы ЭВМ должны быть переданы человеку (другой ЭВМ, управляемому станку и т. д.), другими Словами — выведены. Эти операции осуществляются устройствами ввода-вывода.

Основным устройством ввода, используемым человеком, является клавиатура (аналогичная клавиатуре пишущей машинки, кассового аппарата и калькулятора). На клавиатуре имеются буквы русского и латинского алфавитов, цифры, знаки препинания и специальные символы. В память ЭВМ они передаются закодированными с помощью электрических сигналов так, как это объяснялось выше.

Основным устройством вывода информации для непосредственного восприятия ее человеком является дисплей — телевизионный экран, на котором изображаются буквы, цифры и другие символы, имеющиеся на клавиатуре. Если этим возможности дисплея ограничиваются, то он называется алфавитно-цифровым (так как основными символами клавиатуры являются символы алфавита и цифры). Если на дисплее (используя соответствующие программы) можно, кроме того, получать различные геометрические изображения, то дисплей называется графическим.

Помимо получения изображений на экране дисплея, ЭВМ позволяет получать их на бумаге. Если требуется вывод алфавитно-цифровой информации, то это делается с помощью алфавитно-цифрового печатающего устройства (АЦПУ). Если требуется вывод графической информации (см. рис. 2, 6, 10), это делается с помощью графопостроителя или других устройств

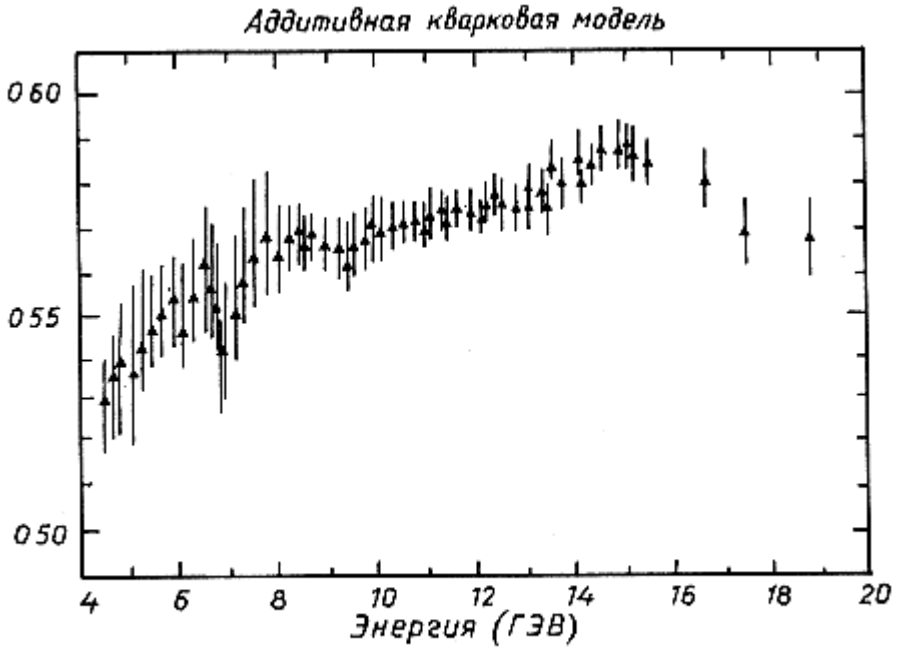

Рис. 10. Результаты математического моделирования физической системы изображены с помощью графопостроителя

Виды памяти ЭВМ. Выполнение каждой операции процессором требует определенного времени, для чтения информации из памяти и ее записи в память также требуется определенное время. Имеются различные виды памяти ЭВМ, основанные на различных принципах. При этом память с быстрым чтением и записью информации оказывается более дорогой, а память с медленным чтением и записью можно сделать более дешевой и большего объема.

В современных ЭВМ разные виды памяти применяются одновременно. Это связано со следующим. Чтобы обеспечить бесперебойную работу процессора, нужно, чтобы время чтения требуемой информации из памяти было не больше, чем время выполнения операции. Таким образом, в состав ЭВМ должна входить быстрая память. Такую память называют также оперативной (так как процессор обращается к ней постоянно в ходе выполнения своих операций) или внутренней (так как она непосредственно связана с процессором).

С другой стороны, если о некоторой информации заранее известно, что она долго не понадобится, ее можно поместить в медленную память и лишь при необходимости переписать в оперативную память. Медленную память называют внешней.

Физические основы работы ЭВМ. Как было сказано выше, обрабатываемая ЭВМ информация представляется в виде электрических сигналов. Обработку этих сигналов осуществляют электронные устройства, состоящие из тысяч и даже сотен тысяч согласованно работающих элементов. Такие устройства называют микросхемами (приставка микро- связана именно с тем, что очень большое число элементов размещается в небольшой схеме). Микросхема может представлять собой целый процессор (например, микропроцессор из числа тех, о которых говорилось выше). Если же все необходимые элементы процессора не удается поместить в одну микросхему, то несколько микросхем помещают на одну плату — пластмассовую пластинку с металлическими проводниками на ней, соединяющими микросхемы (рис. 11).

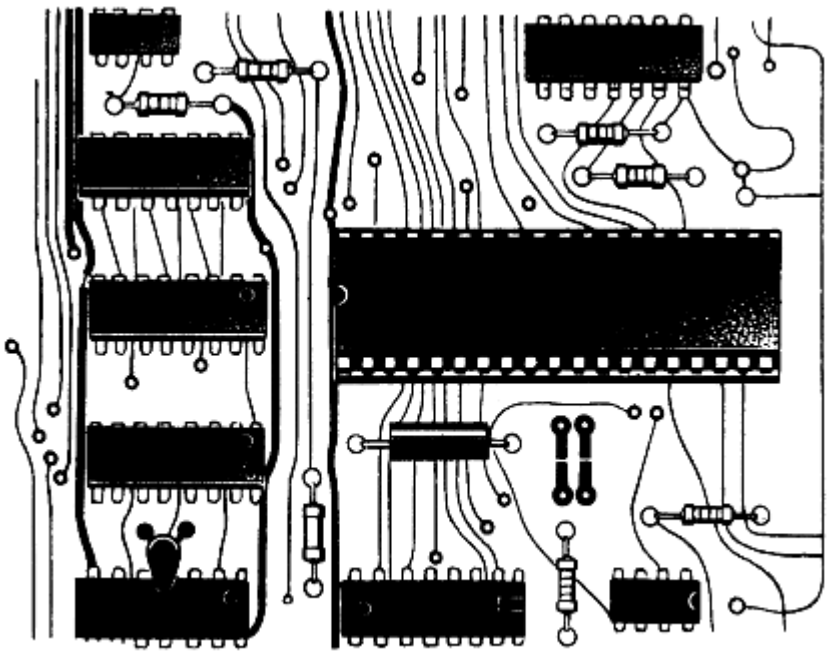

Рис. 11. Часть платы процессора ЭВМ

На такой же плате располагаются микросхемы, образующие внутреннюю память ЭВМ. Эти микросхемы напоминают микросхемы процессора и работают на тех же физических принципах (о них вы узнаете из курса физики).

Внешняя память ЭВМ устроена совсем иначе. Самым распространенным сейчас видом этой памяти является магнитная. В ней информация кодируется не электрическим сигналом, а намагниченностью частички вещества.

В качестве примера внешней памяти рассмотрим гибкие диски. Такой диск по размерам близок к гибкой грампластинке. На нем может быть записано до 0,5 Мбайт информации. Для записи и чтения информации гибкий диск помещают в устройство, называемое дисководом.

Основные характеристики современных ЭВМ. Вы знаете, что компьютеры возникли совсем недавно и развиваются очень быстро. Прогресс в этом развитии связан с достижениями информатики, усовершенствованием технологии производства и открытием новых физических принципов работы элементов компьютера, прежде всего элементов процессора. В соответствии с этим выделяют четыре поколения ЭВМ. В ЭВМ первого поколения (40—50-е гг.) использовались электронные лампы — те самые, которые сейчас употребляются в некоторых телевизорах. В ЭВМ второго поколения (50—60-е гг.) использовались транзисторы. Третье поколение ЭВМ (60—70-е гг.) уже содержало микросхемы, но небольшие, заменявшие десяток — сотню транзисторов. В теперешнем, четвертом поколении (с середины 70-х гг.) используются намного более сложные микросхемы, называемые, также большими интегральными схемами. Эволюцию ЭВМ можно проиллюстрировать следующей приблизительной схемой (рис. 12).

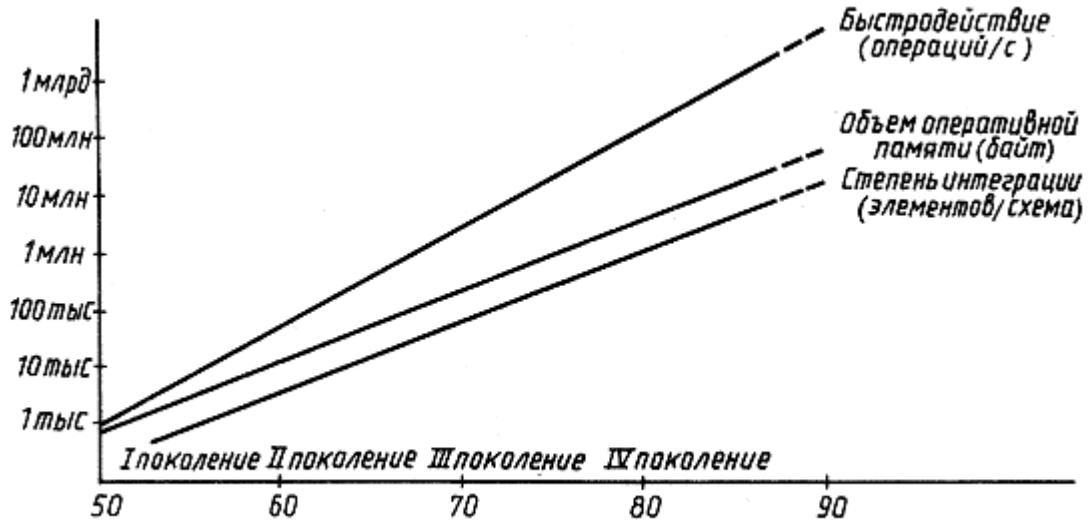

Рис. 12. Поколения ЭВМ

Конечно, поколения ЭВМ различаются не только тем, из каких электронных деталей они сделаны (элементной базой), но и тем, как организована их работа (архитектурой) и какие есть для них программы (программным обеспечением).

В заключение приведем основные технические характеристики современных ЭВМ: быстродействие от сотен тысяч до сотен миллионов операций в 1 с, внутренняя память от десятков кбайт до десятков Мбайт, внешняя память от сотен кбайт до сотен Гбайт.

Вопросы

1. Какие области применения ЭВМ вам известны?
2. Приведите несколько примеров задач, для решения которых, по вашему мнению, была бы полезна ЭВМ.

Упражнения

1. Заметьте, сколько времени вам понадобится, чтобы подсчитать число букв «а» в первых 10 строках страницы, которую вы читаете. Чтобы безошибочно подсчитать количество букв «а» в тексте учебника, компьютеру понадобилось бы около 10 с. Во сколько раз это быстрее того, что вы могли бы сделать вручную без ЭВМ?
2. Предположим, что компьютер выполняет 100 000 операций в 1 с и потребляет мощность 100 Вт. Сколько операций можно сделать за 1 р.? (Цена 1 кВт/ч электроэнергии равна 4 к.)
3. На диске объемом 100 Мбайт подготовлена к выдаче на экран дисплея информация: 24 строчки по 80 символов, эта информация заполняет экран целиком. Какую часть диска она занимает?
4. Придумайте эксперимент, позволяющий узнать, сколько символов в 1 с вы читаете, пишите, произносите. Проведите этот эксперимент.
5. Текст, который вы произносите, вводится в память ЭВМ объемом 64 кбайт. Через сколько минут память будет полностью заполнена?
6. Оцените, сколько школьных сочинений среднего размера можно уместить на гибком диске емкостью 500 кбайт.
7. Печатающее устройство печатает 100 символов в 1 с. Сколько времени будет печататься страница, которую вы сейчас читаете?
8. Пусть в некотором компьютере расстояние между процессором и памятью равно 30 см, а каждое выполнение операции требует передачи информации от процессора к памяти и обратно. Покажите, что такой компьютер не может работать с быстродействием 600 млн. операций в секунду.

Раздел I

АЛГОРИТМЫ. АЛГОРИТМИЧЕСКИЙ ЯЗЫК

§ 1. АЛГОРИТМ И ЕГО СВОЙСТВА

Любой человек ежедневно встречается с множеством задач от самых простых и хорошо известных до очень сложных. Для многих задач существуют определенные правила (инструкции, предписания), объясняющие исполнителю, как решать данную задачу. Эти правила человек может изучить заранее или сформулировать сам в процессе решения задачи. Чем точнее и понятнее будут описаны правила решения задач, тем быстрее человек овладеет ими и будет эффективнее их применять.

Решение многих задач человек может передавать техническим устройствам — автоматам, электронным вычислительным машинам, роботам. Применение таких технических устройств предъявляет очень строгие требования к точности описания правил и последовательности выполнения действий. Поэтому разрабатываются специальные языки для четкого и строгого описания различных правил. Это одна из задач информатики.

1. ПОНЯТИЕ АЛГОРИТМА

Под *алгоритмом* понимают понятное и точное предписание (указание) исполнителю совершить последовательность действий, направленных на достижение указанной цели или на решение поставленной задачи.

Слово алгоритм происходит от algorithmi — латинской формы написания имени великого математика IX в. аль-Хорезми, который сформулировал правила выполнения арифметических действий. Первоначально под алгоритмами и понимали только правила выполнения четырех арифметических действий над многозначными числами. В дальнейшем это понятие стали использовать вообще для обозначения последовательности действий, приводящих к решению поставленной задачи.

Рассмотрим примеры.

Пример 1.1. Вычислить значения y по формуле

$$y = (Ax + B)(Cx - D)$$ для любого значения x.

Чтобы решить эту задачу, достаточно выполнить последовательность следующих действий:

1) умножить A на x, результат обозначить R_1;
2) R_1 сложить с B, результат обозначить R_2;
3) умножить C на x, результат обозначить R_3;
4) из R_3 вычесть D, результат обозначить R_4;
5) умножить R_2 на R_4, считать результат значением y.

Это предписание является алгоритмом решения поставленной задачи. Для человека, выполняющего действия, уже необязательно знать исходную формулу для вычисления значения y. Ему нужно всего лишь строго следовать указанному предписанию, исполняя его пункт за пунктом.

В приведенном примере процесс решения задачи расчленяется на элементарные операции, арифметические действия, но это членение может быть продолжено и дальше. Например, п. 2 этого алгоритма — сложение чисел R_1 *и* B — можно развернуть в систему правил, описывающих процесс сложения двух чисел столбиком. Принцип расчленения

сложного процесса решения задачи на элементарные действия имеет важное значение для построения алгоритмов.

Пример 1.2. Найти наибольший общий делитель двух натуральных чисел *m* и *n*.

Составим алгоритм решения этой задачи, основанный на том свойстве, что если *m>n*, то наибольший общий делитель чисел *m, n* такой же, как и чисел *m − n, n.*

Алгоритм будет таким:

1) если числа равны, то взять любое из них в качестве ответа, в противном случае продолжить выполнение алгоритма;
2) определить большее из чисел;
3) заменить большее число разностью большего и меньшего чисел;
4) начать алгоритм сначала.

Как видно, этот алгоритм, известный под названием алгоритма Евклида, также состоит из отдельных пунктов, предписывающих исполнителю выполнить некоторое простое действие. Его особенностью является то, что все действия, указанные в алгоритме, могут повторяться многократно.

В принципе необходимость вернуться к началу алгоритма может привести к бесконечному повторению пунктов алгоритма. В данном случае этого не произойдет, потому что величина разности большего и меньшего чисел с каждым новым вычитанием уменьшается, и поэтому после некоторого числа повторений сравниваемые числа обязательно станут равными. Алгоритм применим к любым натуральным числам и всегда приводит к решению задачи.

С помощью алгоритмов можно решать не только вычислительные, но и многие другие задачи. Приведем пример алгоритма решения графической задачи.

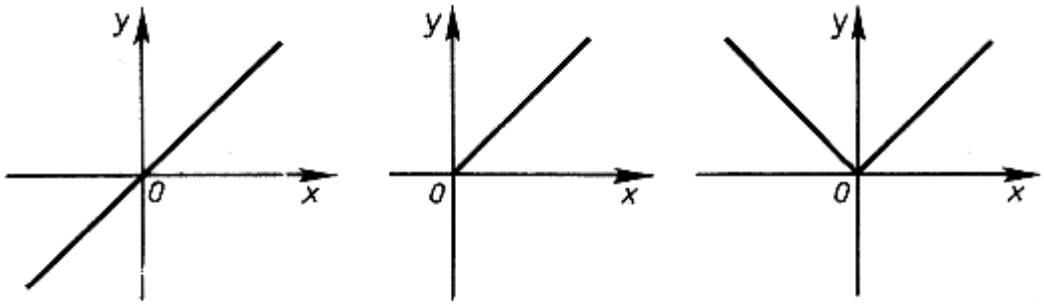

Рис. 13.

Пример 1.3. Построить график функции $y = a\,|x|$ при $a > 0$. Алгоритм построения имеет следующий вид (рис. 13):

1) начертить график функции $y = ax$;
2) стереть часть графика левее оси ординат;
3) имметрично отразить оставшуюся часть графика относительно оси ординат.

Пример 1.4. Игра Баше. (В игре участвуют двое.) Рассмотрим частный случай игры. Имеется 15 предметов. Соперники ходят по очереди, за каждый ход любой из играющих может взять 1, 2 или 3 предмета. Проигрывает тот, кто вынужден взять последний предмет.

Алгоритм выигрыша для первого игрока имеет следующий вид:

1) взять два предмета;
2) второй и последующие ходы делать так, чтобы количество предметов, взятых вместе с соперником за очередной ход, в сумме составляло 4.

Данный алгоритм приводит к выигрышу для 7, 11, 15, 19, ... предметов.

Человек, пользующийся данным алгоритмом, всегда будет выигрывать в этой игре. Ему совершенно необязательно знать, почему надо поступать именно так, а не иначе. Для успешной игры от него требуется только строго следовать алгоритму.

Примеры алгоритмов показывают, что запись алгоритма распадается на отдельные указания исполнителю выполнить некоторое законченное действие. Каждое тако'е указание называется *командой.* Команды алгоритма выполняются одна за другой. На каждом шаге исполнения алгоритма исполнителю точно известно, какая команда должна выполняться следующей.

Поочередное выполнение команд алгоритма за конечное число шагов приводит к решению задачи, к достижению цели.

Каждый алгоритм строится в расчете на некоторого исполнителя. Для того чтобы исполнитель мог решить задачу по заданному алгоритму, необходимо, чтобы он был в состоянии выполнить каждое действие, предписываемое командами алгоритма.

Совокупность команд, которые могут быть выполнены исполнителем, называется *системой команд исполнителя.*

Алгоритм должен быть понятным исполнителю, т. е. каждая его команда должна входить в систему команд исполнителя.

Таким образом, для правильного построения алгоритма необходимо знать систему команд исполнителя и быть уверенным, что исполнение алгоритма всегда завершится за конечное число шагов.

2. ФОРМАЛЬНОЕ ИСПОЛНЕНИЕ АЛГОРИТМА

Пример 2.1. Построить алгоритм нахождения середины отрезка при помощи циркуля и линейки. Алгоритм деления отрезка *AB* пополам:

1) поставить ножку циркуля в точку *A;*
2) установить раствор циркуля равным длине отрезка *AB;*
3) провести окружность;
4) поставить ножку циркуля в точку *B;*
5) провести окружность;
6) через точки пересечения окружностей провести прямую;
7) отметить точку пересечения этой прямой с отрезком *AB*. Каждая его команда указывает исполнителю одно конкретное законченное действие, и исполнитель должен выполнить его целиком. Исполнитель не может перейти к выполнению следующей команды, не закончив полностью выполнения предыдущей. Команды алгоритма надо выполнять последовательно одну за другой, в соответствии с указанным порядком их записи. Выполнение всех команд гарантирует правильное решение задачи. Данный алгоритм будет понятен исполнителю, умеющему работать с циркулем и знающему, что такое поставить ножку циркуля, провести окружность и т. д. Для ученика I класса этот алгоритм не понятен, а для ученика IX класса понятен.

Пример 2.2. Алгоритм «отгадывания» задуманного числа.

Пусть кто-либо задумает произвольное натуральное число. Ему предлагается произвести с этим числом следующие действия и затем сообщить результат:

1) умножить задуманное число на 5;
2) прибавить 8;
3) сумму умножить на 2.

Необходимо по результату «отгадать» задуманное число.

Решение данной задачи сводится к решению уравнения $(x \cdot 5 + 8) \cdot 2 = a,$ где x — неизвестное задуманное число, a — полученный результат.

«Отгадывание» x можно поручить исполнителю, совершенно незнакомому с содержанием задачи. Для этого достаточно сообщить ему следующий алгоритм:

1) вычесть из результата 16;
2) в полученной разности отбросить крайнюю правую цифру, полученное число и будет искомым.

Исполняя алгоритм, исполнитель может не вникать в смысл того, что он делает, и вместе с тем получать нужный результат.

В таком случае говорят, что исполнитель действует формально, т. е. отвлекается от содержания поставленной задачи и только строго выполняет некоторые правила, инструкции.

Это очень важная особенность алгоритмов. Наличие алгоритма формализовало процесс «отгадывания» задуманного числа, исключило рассуждения. Если обратиться к алгоритмам, которые мы рассматривали ранее, то можно увидеть, что и они позволяют исполнителю действовать формально. Таким образом, создание алгоритма дает возможность решать задачу формально, механически исполняя команды алгоритма в указанной последовательности.

Построение алгоритма для решения задачи из какой-либо области требует от человека глубоких знаний в этой области, бывает связано с тщательным анализом поставленной задачи, сложными, иногда очень громоздкими рассуждениями. На поиски алгоритма решения некоторых задач ученые затрачивают многие годы. Но когда алгоритм создан, решение задачи по готовому алгоритму уже не требует каких-либо рассуждений и сводится только к строгому выполнению команд алгоритма.

В этом случае исполнение алгоритма можно поручить не человеку, а машине. Действительно, простейшие операции, на которые при создании алгоритма расчленяется процесс решения задачи, может реализовать и машина, специально созданная для выполнения отдельных команд алгоритма и выполняющая их в последовательности, указанной в алгоритме. Это положение и лежит в основе работы автоматических устройств.

Вопросы

1. Приведите примеры известных вам алгоритмов.
2. Что понимается под командой алгоритма?
3. Что называется системой команд исполнителя? Поясните на примере.
4. Могут ли автоматические устройства быть исполнителями алгоритмов?

Упражнения

1. Сформулируйте и запишите алгоритм вычисления значения y по формуле:

а) $y = (2x + 3)(7x - 5)$; б) $y = \dfrac{2 - (x-3)^2}{(x-3)^2 + 4}$.

2. По приведенному алгоритму восстановите формулу для вычисления значения y:

а) 1) умножить x на x, обозначить результат R_1;
2) умножить R_1 на a, обозначить результат R_2;
3) сложить R_2 с b, обозначить результат R_3;
4) разделить R_3 на c, считать результат значением y;

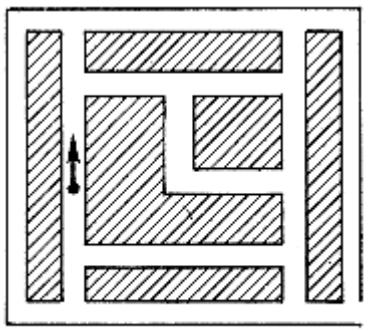

Рис. 14.

б) 1) сложить x с 1, обозначить результат A_1;
2) разделить 1 на A_1, обозначить результат A_2;
3) сложить A_2 с 1, обозначить результат A_3;

4) вычесть из A_2 единицу, обозначить результат A_4;
5) разделить A_4 на A_3, обозначить результат A_5;
5) вычесть из A_5 единицу, считать результат значением *y*.

3. Сформулируйте и запишите алгоритмы построения с помощью циркуля и линейки:
а) серединного перпендикуляра данного отрезка;
б) окружности, для которой данный отрезок — диаметр;
в) биссектрисы угла;
г) точки пересечения медиан данного треугольника;
д) перпендикуляра к прямой, проходящего через точку, принадлежащую прямой.
4. Двое играют в игру Баше с 20 предметами. Начинающий игрок действует по алгоритму 1.4. Может ли второй игрок у него выиграть?
5. Преобразуйте алгоритм так, чтобы его мог исполнить ученик II класса, не знающий действий возведения в квадрат и в куб:
1) возвести *x* в квадрат, результат обозначить a_1;
2) умножить a_1 на 2, результат обозначить a_2;
3) сложить a_2 и 15, результат обозначить a_3;
4) возвести a_3 в куб, результат обозначить a_4;
5) сложить a_4 и 25, результат обозначить a_5;
6) разделить a_3 на a_5, считать результат значением *y*,
6. Человек находится в лабиринте (рис. 14) и начинает двигаться в направлении, указанном стрелкой, согласно следующему предписанию: иди шаг за шагом, не отрывая руки от правой стены. Шагай, пока не выйдешь из лабиринта.
а) Достигает ли данное предписание цели вывести исполнителя из лабиринта?
б) Нарисуйте путь исполнителя этого предписания.
7. Сколько раз будет выполняться шаг 3 алгоритма Евклида для $m = 100$, $n = 18$?

§ 2. АЛГОРИТМИЧЕСКИЙ ЯЗЫК

Алгоритмический язык — это система обозначений и правил для единообразной и точной записи алгоритмов и их исполнения. Алгоритмический язык, с одной стороны, близок к обычному языку. Алгоритмы на этом языке могут записываться и читаться как обычный текст. С другой стороны, алгоритмический язык включает в себя и математическую символику: числа, обозначения величин и функций, знаки операций и скобки и др.

Правила алгоритмического языка лежат в основе языков программирования для ЭВМ. Изучение алгоритмического языка поможет вам в будущем освоить любой язык программирования для ЭВМ.

3. ОБЩИЕ ПРАВИЛА АЛГОРИТМИЧЕСКОГО ЯЗЫКА

Как и каждый язык, алгоритмический язык имеет свой словарь. Основу этого словаря составляют слова, употребляемые для записи команд, входящих в систему команд исполнителя того или иного алгоритма. Такие команды называются *простыми* командами.

Обычно простая команда выглядит как повелительное предложение русского языка в полной или сокращенной форме, включая, если необходимо, формулы и другие символические обозначения.

Кроме того, в алгоритмическом языке используется некоторое ограниченное число слов, смысл и способ употребления которых задан раз и навсегда. Эти слова называются *служебными словами.* При записи алгоритмов они выделяются и записываются, как правило, в сокращенной форме. Использование служебных слов делает запись алгоритма более наглядной, а форму представления различных алгоритмов — единообразной.

Алгоритм, записанный на алгоритмическом языке, должен иметь название. Название выбирается так, чтобы было ясно, решение какой задачи описывает данный алгоритм. Для выделения названия алгоритма перед ним записывается служебное слово **алг** (**алг**оритм).

За названием алгоритма (обычно с новой строки) записываются его команды. Для указания начала и конца алгоритма его команды заключаются в пару служебных слов **нач** (начало) и **кон** (**кон**ец). Команды записываются последовательно. При записи одной команды можно перейти на другую строчку. Если несколько команд записываются на одной строчке, то они разделяются точкой с запятой (;).

Последовательность нескольких команд алгоритма, выполняющихся одна за другой, называется *серией.* Серия может состоять и из одной команды.

Итак, общий вид алгоритма, записанного на алгоритмическом языке, таков:

алг название алгоритма
нач
команды алгоритма (серия)
кон

Пример 3.1. Записать на алгоритмическом языке алгоритм нахождения середины отрезка (с. 20), если в систему команд исполнителя входят привычные действия с циркулем и линейкой.

алг деление отрезка *AB* пополам
нач
поставить ножку циркуля в точку *A*
установить раствор циркуля равным длине отрезка *AB*
провести окружность поставить ножку циркуля в точку *B*
провести окружность
через точки пересечения окружностей провести прямую
отметить точку пересечения этой прямой с отрезком *AB*
кон

Пример 3.2. Записать на алгоритмическом языке инструкцию по использованию междугородного телефона-автомата:

Снимите трубку. Опустите монету 15 к. Услышав непрерывный сигнал, наберите код нужного вам города, а затем сразу нужный номер. Услышав ответ абонента, нажмите кнопку «Разговор».

алг использование междугородного телефона-автомата
нач
снять трубку
опустить монету 15 к.
дождаться появления непрерывного сигнала
набрать код нужного вам города
набрать номер нужного вам телефона
дождаться ответа абонента
нажать кнопку «Разговор»
говорить
кон

4. СОСТАВНЫЕ КОМАНДЫ

Некоторые типы алгоритмов: линейные, разветвляющиеся и циклические уже встречались вам в курсе алгебры.

В алгоритмическом языке линейными являются алгоритмы, состоящие из одной серии простых команд. Для записи разветвляющихся и циклических алгоритмов в алгоритмическом языке используются так называемые составные команды, аналогичные сложным предложениям русского языка.

В алгоритмическом языке употребляются две основные составные команды: команда ветвления и команда повторения (цикла). Каждая из этих двух команд отличается от простых тем, что в нее входит условие, в зависимости от которого выполняются или не выполняются команды из числа входящих в составную.

1. Команда ветвления записывается следующим образом:

если условие
    то серия 1
    иначе серия 2
все

В зависимости от условия выполняется только одна из двух серий команд, входящих в команду ветвления. Если условие соблюдено, то надо выполнить серию 1, а если нет — серию 2.

Команда ветвления используется и в сокращенной форме:

если условие
    то серия
все

В этом случае, если условие соблюдено, исполнитель выполняет серию команд, следующую в записи алгоритма за служебным словом **то,** а в противном случае, пропуская серию, переходит к выполнению команды, следующей за командой ветвления (после служебного слова **все**).

Команды из каждой серии выполняются подряд, каждая по своим правилам. Команда ветвления заканчивается, как только выполнится последняя команда из серии 1 или серии 2.

В качестве условия в команде ветвления может быть использовано любое понятное исполнителю утверждение, которое может соблюдаться или не соблюдаться. Утверждение может быть выражено словами или формулой.

Читается команда ветвления так же, как и записывается.

При изображении алгоритмов в виде схем условие можно понимать как вопрос, на который возможен ответ «да» или «нет».

На рисунке 15 изображены схемы для полной и сокращенной формы записи команды ветвления.

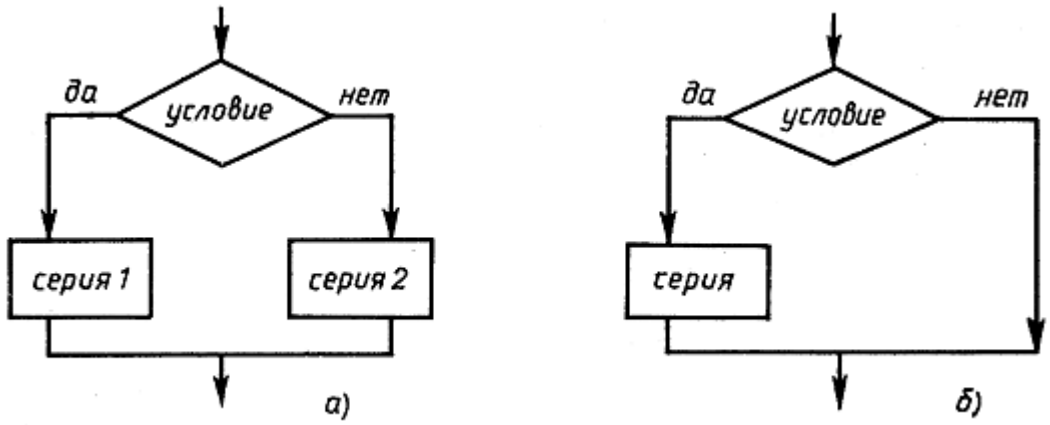

Рис. 15.

Проиллюстрируем на примерах употребление команды ветвления.

Пример 4.1. Записать алгоритм правописания приставок на «з» («с»).

алг правописание приставок на «з» («с»)
нач
    если корень слова начинается со звонкой согласной
        то на конце приставки написать «з»

иначе на конце приставки написать «с»

все

кон

Пример 4.2. Нужно включить электроприбор, имеющий переключатель напряжения, в сеть 220 В.

алг включение электроприбора в сеть 220 В
нач
если переключатель прибора установлен на 127 В
то установить переключатель прибора на 220 В
все
вставить вилку в розетку
кон

Пример 4.3. Определить кислотность раствора. Чтобы решить эту задачу, достаточно опустить в раствор лакмусовую бумажку и по ее цвету определить, является ли раствор кислотным, щелочным или нейтральным.

алг определение кислотности раствора
нач отлить в пробирку 1 мл раствора
опустить в пробирку лакмусовую бумажку
если бумажка красная
то ответ: раствор кислотный
иначе если бумажка синяя
то ответ: раствор щелочной
иначе ответ: раствор нейтральный
все
все
кон

Схема алгоритма приведена на рисунке 16.

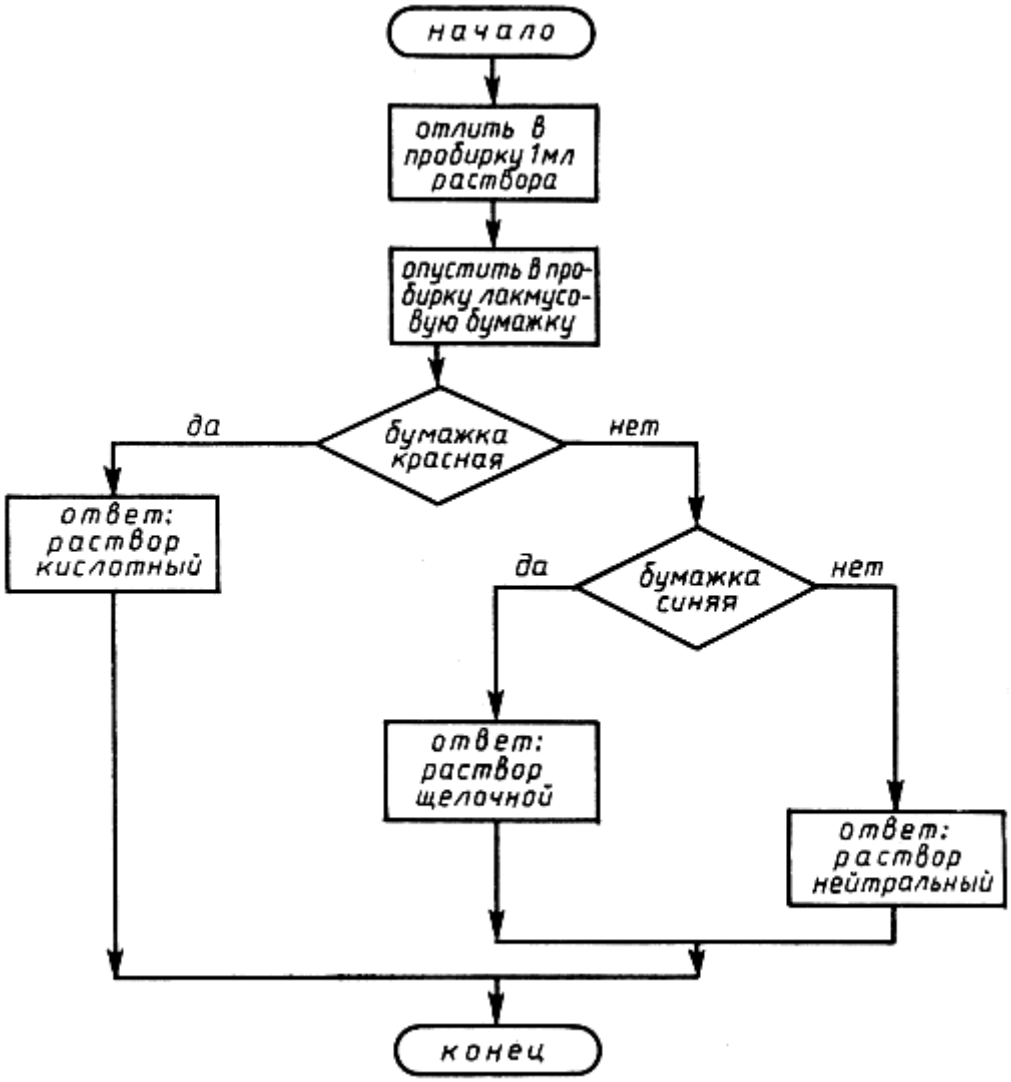

Рис. 16.

2. Команда повторения. В своей практической деятельности человек постоянно встречается с задачами, для решения которых требуется многократно повторять одни и те же действия. Именно для этого применяется составная команда повторения (цикл).

Команда повторения имеет особое значение для построения алгоритмов, исполняемых на ЭВМ, так как только ее использование позволяет с помощью сравнительно коротких алгоритмов записывать предписания о совершении очень длинной последовательности действий, для которых требуется высокая скорость ЭВМ.

Команда повторения записывается следующим образом:

пока условие
нц
серия
кц

Выполнение этой команды приводит к тому, что указанная в ней серия команд выполняется несколько раз подряд. Она выполняется столько раз, сколько нужно для того, чтобы указанное условие перестало соблюдаться.

Если условие не соблюдается с самого начала, то серия не выполняется ни разу. Условие цикла проверяется перед выполнением серии, но не в процессе ее выполнения.

Выполнение команды цикла можно пояснить графически в виде схемы (рис. 17),

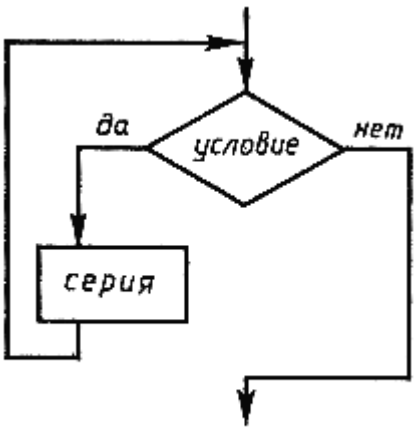

Рис. 17.

Поясним особенности выполнения этой команды на следующих примерах.

Пример 4.4. Пусть у исполнителя имеется пустое 7-литровое ведро, которое надо наполнить до краев теплой водой, выполняя следующие команды:

пока ведро не полно
нц
долить 1 л холодной воды
долить 1 л горячей воды
кц

Выполняя эту составную команду, исполнитель трижды дольет в ведро по 2 л, так как между служебными словами **нц** и **кц** стоят две команды: «долить 1 л». После этого, поскольку ведро еще не будет полным, он выполнит серию еще раз. При этом 1 л воды выльется из ведра.

Пример 4.5. Записать на алгоритмическом языке выигрышный алгоритм игры Баше, если имеется 15 предметов (см. пример 1.4).

алг игра Баше
нач
взять два предмета
пока осталось больше четырех предметов
нц
дать противнику сделать ход
запомнить число К взятых противником предметов
взять 4 — К предметов
кц
дать противнику сделать ход
кон

Пример 4.6. Используя уже известные команды ветвления и повторения, составить более сложный алгоритм — нахождения массы небольшого предмета с помощью чашечных весов.

алг определение массы предмета
нач
положить на левую чашку весов исследуемый предмет
положить на правую чашку весов гирьку 10 г
пока чашки не уравновешены
нц если перевесила левая чашка
то добавить на правую чашку еще одну гирьку
такой же массы, какую положили туда в предыдущий раз
иначе снять с правой чашки последнюю положенную на нее гирьку и заменить ее гирькой в 10 раз меньшей массы

все

кц

сосчитать суммарную массу гирек на правой чашке с гирьками

кон

Вопросы

1. Для чего нужен алгоритмический язык?
2. Какова роль служебных слов **алг, нач, кон, если, то, иначе, все, пока, нц, кц?** Поясните на примерах.
3. Какие бывают составные команды? Приведите примеры.

Упражнения

1. Запишите алгоритмы из упражнения 3, § 1 на алгоритмическом языке.
2. Измените алгоритм примера 4.4 так, чтобы вода не проливалась.
3. Придумайте и запишите выигрышный алгоритм для второго игрока игры Баше для 5 предметов.

§ 3. АЛГОРИТМЫ РАБОТЫ С ВЕЛИЧИНАМИ

Решение большинства практических задач связано с преобразованием информации.

Так, при решении квадратного уравнения вида $ax^2 + bx + c = 0$ на основании исходной информации (вида уравнения и значений *a, b* и *c)* мы получаем новую информацию — результат (имеет ли это уравнение корни, и если имеет, то получаем значения корней x_1 и x_2).

При нахождении середины отрезка *AB* мы на основании информации об отрезке *AB* получаем результат — точку, отмечающую его середину.

Под терминами «исходная информация», «преобразованная информация», «результат» понимаются вполне конкретные величины — числовые (коэффициенты о, *b* и *c),* графические (отрезок) и т. п.

5. ВЕЛИЧИНЫ

Величины делятся на переменные и постоянные. *Постоянной* называется величина, значение которой не меняется в процессе исполнения алгоритма, а остается одним и тем же, указанным в тексте алгоритма (например, 15; 2,4; 3,14 и т. д.). *Переменной* называется величина, значение которой меняется в процессе исполнения алгоритма.

При написании алгоритма для переменных величин вводятся обозначения аналогично обозначениям переменных в курсе алгебры. Такое обозначение переменной величины в алгоритмическом языке называется *именем* величины.

При исполнении алгоритма в каждый момент времени величина обычно имеет некоторое значение. Оно называется *текущим* значением. В процессе исполнения алгоритма величина может не получить конкретного значения. Такая величина называется неопределенной.

В школьном курсе математики и физики чаще всего встречаются числовые величины, значениями которых являются натуральные, целые, действительные числа. Однако в алгоритмах столь же часто встречаются и нечисловые величины: слова, таблицы, списки, тексты, графики, геометрические фигуры и т. д. В курсе математики и физики величины чаще всего обозначаются одной буквой латинского или греческого алфавита. Поскольку при применении ЭВМ и при записи алгоритмов и программ разнообразие величин очень велико, в алгоритмическом языке принято использовать в качестве имен величин произвольные буквы, буквосочетания и любые слова, поясняющие смысл и назначение величины в алгоритме.

Для того чтобы отличить имена величин от обычных слов в этом учебнике, имена величин будут выделены другим шрифтом — курсивом *(например,произведение, число*

шагов и т. д.). Например, *класс 9а* в школе является переменной величиной, значение которой — множество учеников, обучающихся в этом классе в текущем учебном году.

Величины, значениями которых являются слова или текст, называются *литерными.*

Для того чтобы выделить текст, который является значением литерной величины, значения литерных величин берутся в кавычки, например „нет решения".

Как мы видим, величины могут иметь различный тип. Они могут быть натуральными, целыми, действительными, литерными и т. д. в соответствии с тем, что может быть их назначением. Отметим, что в программировании вместо слова «действительный» в применении к числам используется слово «вещественный». Сокращенно типы переменных обозначаются словами **нат (**натуральный), **цел** (целый), **вещ** (вещественный), **лит** (литерный) и т. д.

6. ЗАГОЛОВОК АЛГОРИТМА

Запись всякого алгоритма начинается с заголовка. Вот пример заголовка алгоритма нахождения остатка от деления двух натуральных чисел:

алг остаток от деления **(нат** *делимое,* **нат** *делитель,* **нат** *остаток)*
арг *делимое, делитель*
рез *остаток*

Здесь **остаток от деления** — название алгоритма, *делимое, делитель* и *остаток* — имена величин. Перед каждым именем величины указан ее тип. Исходными данными для алгоритма нахождения остатка от деления двух натуральных чисел являются величины *делимое, делитель* типа **нат,** а в результате его выполнения величина *остаток* приобретает значение, равное остатку от деления значений величин *делимое* и *делитель.* Величины, являющиеся исходными данными для алгоритма, называются *аргументами.* Их список помещается после служебного слова **арг** (аргумент). Результаты алгоритма (в данном примере — величина *остаток)* перечисляются после служебного слова **рез** (результат).

Общий вид заголовка алгоритма таков:

алг название алгоритма (список величин с указанием типов)
арг имена аргументов
рез имена результатов

Имена аргументов и результатов алгоритма перечисляются через запятую.

Пример 6.1. Заголовок алгоритма Евклида:

алг нахождение наибольшего общего делителя (НОД) **(нат** *M,*
N, **нат** НОД)
арг M, N
рез *НОД*

7. ПРОМЕЖУТОЧНЫЕ ВЕЛИЧИНЫ. ПРИСВАИВАНИЕ ЗНАЧЕНИЙ

Пример 7.1. Рассмотрим запись алгоритма работы с величинами на примере алгоритма решения квадратного уравнения $ax^2 - bx + c = 0$.

алг решение квадратного уравнения **(вещ** a, b, c, x_1, x_2, **лит** y)
арг *a, b, c*
рез x_1, x_2,y
нач **вещ** D
$D = b^2 - 4 \cdot a \cdot c$
если $D < 0$
то y:= „нет решения"
иначе y:= „есть решение"

$$x_1 := \frac{-b+\sqrt{D}}{2a}; x_2 := \frac{-b-\sqrt{D}}{2a}$$

все

кон

Разберем подробно приведенный пример. Значение корней x_1 и x_2 надо найти при конкретных значениях *a, b* и с. Поэтому переменные *a, b и c* являются аргументами, а корни x_1 и x_2 — результатом алгоритма. Если корни уравнения не существуют, то алгоритм должен выдавать в качестве результата сообщение об отсутствии корней. Для этого введена литерная переменная *y,* значением которой является текст сообщения о том, имеет или нет уравнение корни.

Текст алгоритма начинается с заголовка.

Затем в тексте алгоритма следуют команды, указывающие, как решать поставленную задачу. Но перед ними после знакомого нам служебного слова **нач** появилось еще одно описание типа переменной: **вещ** *D*

Переменная *D* (дискриминант квадратного уравнения) не является ни аргументом, ни результатом алгоритма, а используется только при его выполнении для обозначения вычисляемого промежуточного значения. Такие переменные называются *промежуточными.* Их тип также должен быть указан исполнителю, и их описание приводится после служебного слова **нач.**

Далее в записи алгоритма следует команда:

$$D := b^2 - 4 \cdot a \cdot c.$$

Эта команда читается так: «Присвоить переменной *D* значение выражения b^2 — 4 ·*a* · *c».* Команда такого вида называется *присваиванием.*

Знак присваивания (: =) делит команду присваивания на левую и правую части. В левой части может стоять любая переменная величина алгоритма, а в правой части — любое числовое или нечисловое выражение.

Знак «: = » нельзя путать со знаком математического равенства «=». Знак «:==» указывает исполнителю действие (присвоить переменной вычисленное значение), в то время как знак «=» никаких действий не предполагает.

Далее в тексте алгоритма следует уже известная нам составная команда ветвления, содержащая две серии команд. Первая серия (после служебного слова **то**) состоит из одной команды присваивания литерной переменной *y:* = „нет решения".

Вторая серия (после служебного слова **иначе**) состоит из трех команд присваивания: первая присваивает значение литерной переменной *y,* две оставшиеся присваивают значения числовым переменным x_1, x_2.

8. ИСПОЛНЕНИЕ АЛГОРИТМА

Практическая реализация всех предусмотренных алгоритмом действий по получению результата для конкретных значений аргументов осуществляется в процессе исполнения алгоритма.

Рассмотрим, как происходит исполнение алгоритма решения квадратного уравнения (пример 7.1).

Прежде всего исполнитель составляет таблицу записи значений используемых в алгоритме переменных величин (табл. 1).

Таблица 1

Шаги алгоритма	Аргументы			Промежуточная величина	Результаты			Проверка условий
	a	*b*	*c*	*D*	x_1	x_2	*y*	

Такую таблицу мы будем называть *таблицей значений.* При исполнении алгоритма компьютером значения величин хранятся в его памяти. При исполнении алторитма человеком таблица значений выполняет роль дополнительной памяти для исполнителя.

В данном случае исполнитель получает значения аргументов *a, b* и *c.* Пусть, например, $a = 2$, $b = 1$, $c = -6$. Эти данные исполнитель заносит в таблицу (табл. 2).

Затем он приступает непосредственно к исполнению алгоритма. Алгоритм расчленяется на отдельные команды (для удобства их последовательной регистрации в таблице имеется специальная графа «Шаги алгоритма»).

На первом шаге выполняется команда присваивания

$$D := b^2 - 4 \cdot a \cdot c,$$

т. е. исполнитель подставляет в формулу текущие значения входящих в нее переменных, производит нужные вычисления и присваивает переменной *D* значение 49.

Это значение вносится в таблицу (табл. 2).

Таблица 2

Шаги алгоритма	Аргументы			Промежуточная величина	Результаты			Проверка условий
	a	*b*	*c*	*D*	x_1	x_2	*y*	
	2	1	-6					
1				49				

На этом первый шаг алгоритма заканчивается, и исполнитель переходит к выполнению команды ветвления. Выполнение этой составной команды состоит из нескольких шагов.

Вторым шагом алгоритма является проверка условия $D < 0$ этой команды. Текущим значением переменной *D,* как следует из таблицы, является число 49, поэтому условие $D < 0$ в данном случае не соблюдается. Результат проверки условия для контроля заносится в соответствующую строку графы «Проверка условий» таблицы, а именно записывается условие и за ним слово «да», если условие соблюдено, и «нет» в противном случае (табл. 3).

Условие не соблюдено, нужно переходить к выполнению серии команд, следующей за служебным словом **иначе** команды ветвления, за этим словом стоит серия из трех команд присваивания.

Третий шаг алгоритма — выполнение команды присваивания *y:=* „есть решение". В результате выполнения этого шага исполнитель присваивает переменной *y* литерное значение “есть решение" и это значение вносится в таблицу (табл. 3).

Таблица 3

Шаги алгоритма	Аргументы			Промежуточная величина	Результаты			Проверка условий
	a	*b*	*c*	*D*	x_1	x_2	*y*	
	2	1	-6					
1 2 3				49			есть решение	49 < 0 (нет)

Четвертый шаг алгоритма — выполнение команды присваивания $x_1 := \dfrac{-b + \sqrt{D}}{2a}$. Исполнитель присваивает переменной x_1, значение $\dfrac{-b + \sqrt{D}}{2a} = \dfrac{-1 + \sqrt{49}}{4} = 1{,}5$ (табл. 4).

Таблица 4

Шаги алгоритма	Аргументы			Промежуточная величина	Результаты			Проверка условий
	a	*b*	*c*	*D*	x_1	x_2	*y*	
	2	1	-6					
1				49				49 < 0 (нет)
2								
3							есть решение	
4					1,5			

Пятый шаг алгоритма — выполнение команды присваивания $x_2 := \frac{-b-\sqrt{D}}{2a}$ Исполнитель присваивает переменной x_2 значение $x_2 := \frac{-b-\sqrt{D}}{2a} = \frac{-1-\sqrt{49}}{4} = -2$ (табл. 5).

После пятого шага закончилось выполнение составной команды ветвления и исполнение всего алгоритма. В результате переменные x_1, x_2 и *y* получили значения: $x_2 = -2$, x_1, = 1,5, *y* = „есть решение" (см. табл. 5).

Таблица 5

Шаги алгоритма	Аргументы			Промежуточная величина	Результаты			Проверка условий
	a	*b*	*c*	*D*	x_1	x_2	*y*	
	2	1	-6					
1				49				49 < 0 (нет)
2								
3							есть решение	
4					1,5			
5						-2		

Если бы аргументы алгоритма были заданы так, что значение выражения $b^2 - 4ac$ оказалось бы меньше нуля (например, $a = 2$, $b = -3$, c = 7), то в этом случае исполнение закончилось бы за три шага и переменная *y* получила бы значение „нет решения", а переменные x_1 и x_2 никаких значений не получили бы (табл. 6).

Таблица 6

Шаги алгоритма	Аргументы			Промежуточная величина	Результаты			Проверка условий
	a	*b*	*c*	*D*	x_1	x_2	*y*	
	2	-3	7					
1				-47				-47 < 0 (да)
2								
3							нет решения	

Рассмотрим еще один пример выполнения алгоритма.

П р и м е р 8.1. Алгоритм Евклида — нахождения наибольшего общего делителя двух натуральных чисел *M* и *N*.

алг нахождение наибольшего общего делителя **(нат** *M, N,* **нат**

НОД)
арг *M, N*
рез *НОД*
нач **нат** *x, y*
x:= *M; y*:= *N*
пока $x \neq y$
нц
если $x > y$
то *x*: = $x - y$
иначе *y*:= $y - x$
все
кц
НОД: = *x*
кон

В таблице 7 приведено исполнение алгоритма Евклида для $M = 35$, $N = 21$.

Таблица 7

Шаги алгоритма	Аргументы		Промежуточная величина		Результат	Проверка условий
	M	*N*	*x*	*y*	*НОД*	
	35	21				
1			35			
2				21		
3						35 ≠ 21(да)
4						35 > 21(да)
5			14			
6						14 ≠ 21(да)
7						14 > 21(нет)
8				7		
9						14 ≠ 7 (да)
10						14 > 7 (да)
11			7			
12						7 ≠ 7 (нет)
13					7	

9. ОТНОШЕНИЯ МЕЖДУ ВЕЛИЧИНАМИ В КАЧЕСТВЕ УСЛОВИЙ

Так же как в математике, в алгоритмическом языке используются следующие знаки отношения между величинами:

для числовых величин

< меньше ≥ не меньше = равно

> больше ≤ не больше ≠ не равно

для литерных величин

= равно ≠ не равно

В алгоритмах работы с числовыми и литерными величинами именно такие отношения между величинами и используются в качестве условий.

Пример 9.1. Построить алгоритм решения следующей задачи: «Ракета запускается с точки на экваторе Земли со скоростью *y* (в км/с) в направлении движения Земли по орбите вокруг Солнца. Каков будет результат этого запуска ракеты в зависимости от скорости *y*?»
алг запуск ракеты **(вещ** *y,* **лит** ***A*****)**

```
    арг y
    рез A
нач
    если   y < 7,8
           то A := „ракета упадет на Землю"
           иначе если y < 11,2
                  то A:= „ракета станет спутником Земли"
                  иначе если y < 16,4
                         то A := „ракета станет спутником Солнца"
                         иначе A := „ракета покинет Солнечную систему"
                  все
           все
    все
кон
```

В этом алгоритме имеются переменные величины y и A, постоянные числовые величины 7,8; 11,2; 16,4 и постоянные литерные величины „ракета упадет на Землю" и др. В составных командах в качестве условия используются сравнения переменной величины y и постоянных величин 7,8; 11,2; 16,4. Все условия состояли только из одного отношения между величинами. Такие условия называются *простыми.*

Пример 9.2. Построить алгоритм вычисления выражения $y = \frac{1}{x} + \frac{1}{x-1}$. При этом учесть, что при $x = 0$ и при $x = 1$ выражение не имеет смысла.

```
алг  пример (вещ x, y, лит P)
     арг x
     рез y, P
нач
     если x = 0 или x = 1
          то P := „значение y не определено"
          иначе P := „значение y определено"; y := 1/x + 1/(x-1)
     все
кон
```

В этом алгоритме команда ветвления содержит условие, которое состоит из двух отношений, соединенных словом **или:**

$$x = 0 \text{ **или** } x = 1.$$

Условия такого вида называются *составными.* При записи составных условий на алгоритмическом языке пользуются служебными словами **и**, **или, не.** Использование служебных слов аналогично их использованию в обычном языке. Пусть a **и** b — условия. Условие a **и** b соблюдается, если соблюдаются вместе и a и b. Условие a **или** b соблюдается, если соблюдается хотя бы одно из условий a, b, неважно какое. Условие **не** a соблюдается, если не соблюдается a, и наоборот.

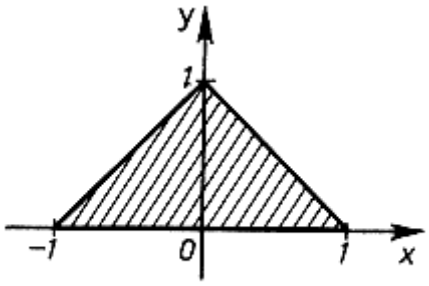

Рис. 18.

Пример 9.3. Составить алгоритм решения следующей задачи: «Точка *A* задана координатами *x* и *y*. Определить, принадлежит ли точка *A* фигуре на плоскости (рис. 18)».

Этой фигуре будут принадлежать точки, координаты которых удовлетворяют условиям:

$$\begin{cases} y \ge 0, \\ |x| + |y| \le 1. \end{cases}$$

Соответствующий алгоритм имеет вид:

алг фигура (**вещ** *x, y,* **лит** *z*)
арг *x, y*
рез *z*
нач
если $y \ge 0$ **и** $|x| + |y| \le 1$
то *z* := „точка принадлежит фигуре"
иначе *z* : — „точка не принадлежит фигуре"
все
кон

Используем этот алгоритм для проверки, принадлежит ли фигуре точка *A* с координатами $x = -0{,}4$ и $y = 0{,}95$.

Проверяем условие: $y \ge 0$ и $|x| + |y| \le 1$. Подставим значения аргументов: $0{,}95 \ge 0$ и $|-0{,}4| + 0{,}95 \le 1$.

Отношение $0{,}95 \ge 0$ является правильным, т. е. первая часть составного условия соблюдается. Однако $|-0{,}4| + 0{,}95 = 0{,}4 + 0{,}95 = 1{,}35 > 1$. Вторая часть условия и, следовательно, все условия не соблюдаются. Результату *z* присваивается значение „точка не принадлежит фигуре".

10. ТАБЛИЧНЫЕ ВЕЛИЧИНЫ

При решении задач человек очень часто пользуется таблицами: при записи исходных данных, получении справочной информации и т. п. Таблицы бывают разными, но наиболее часто встречаются линейные и прямоугольные таблицы.

Значения, образующие линейную таблицу, располагаются при записи на бумаге в строчку или в столбец. Каждому значению, или элементу таблицы, соответствует его порядковый номер, и наоборот: стоит задать порядковый номер, и сразу ясно, о каком элементе таблицы идет речь.

Например, на метеостанции каждый час измеряется температура воздуха и значения измерений за сутки записываются в таблицу (табл. 8).

Таблица 8

Время измерения, ч	0	1	2	3	. . .	22	23
Температура, °С	17	16	15,5	14	. . .	18	17,5

Эта линейная таблица содержит 24 элемента, занумерованные от 0 до 23. Например, второй элемент таблицы имеет значение 15,5, а нулевой элемент — значение 17.

Запишем в таблицу средние температуры дня, измеренные за неделю (табл. 9).

Таблица 9

Дата измерения	22	23	24	25	26	27	28
Средняя температура, °С	15	15,5	17	20	18	17	17,5

Очевидно, что при хранении линейной таблицы порядковые номера хранить нет необходимости: зная начало нумерации, можно путем отсчета найти любой элемент. Кроме того, полезно знать и самый большой порядковый номер, так как это позволяет определить заранее размер таблицы. Таким образом, чтобы указать, что некоторая величина является линейной таблицей, нужно задать тип элементов таблицы, ее имя, начальный и конечный порядковые номера ее элементов.

В алгоритмах работы с табличными величинами это указание записывается следующим образом: служебное слово, указывающее тип (**цел**, **вещ**, **лит** и т. п.), затем служебное слово **таб** (**таб**лица), имя таблицы, за которым стоят в квадратных скобках начальный и конечный порядковые номера ее элементов, разделенные двоеточием. Например:

вещ таб *время* [0:23]

вещ таб *средняя температура* [22:28]

Аналогичным образом происходит описание и прямоугольных таблиц.

Запишем таблицу умножения в виде прямоугольной (табл. 10).

Таблица 10

		множимое					
		1	2	3	. . .	8	9
множитель	1	1	2	3	. . .	8	9
	2	2	4	6	. . .	16	18
	3	3	6	9	. . .	24	27
	⋮	⋮	⋮	⋮	⋮	⋮	⋮
	8	8	16	24	. . .	64	72
	9	9	18	27	. . .	72	81

Для прямоугольной таблицы должны быть указаны границы номеров как по вертикали, так и по горизонтали. Для данного примера это описание имеет вид:

цел таб *произведение* [1:9, 1:9]

границы множителя — границы множимого

Работа с таблицей сводится к работе с ее элементами. Для того чтобы указать, какой элемент таблицы в данный момент используется, достаточно указать его порядковый номер. Этот порядковый номер приписывается к имени таблицы в виде индекса.

Из курса математики хорошо известны обозначения a_0, a_1, b_5, a_i, x_{m+n} и т. п. Однако для того чтобы исполнителю было легче отличить переменные a_0, a_1 и т. д. от нулевого, первого и т. д. элемента таблицы *a,* принято порядковый номер элемента таблицы заключать в квадратные скобки и помещать вслед за именем таблицы на том же уровне строки, например:

a [*i*], *произведение* [2,7] и т. д.

Рассмотрим примеры использования таблиц в алгоритмах.

Пример 10.1. Построить алгоритм, который находит сумму S вещественных чисел, образующих таблицу *a* из 1000 элементов, занумерованных от 1 до 1000. Для отсчета количества просуммированных чисел используется промежуточная целая переменная *I,* которая пересчитывает элементы таблицы.

```
алг  сумма (вещ таб a [1:1000], вещ S)
     арг a
     рез S
нач  цел i
     i := 1
     S := 0
     пока i ≤ 1000
     нц
          S := S + a [i]
          t := t + 1
     кц
кон
```

Использование переменной *i* в качестве индекса позволяет записать прибавление по очереди всех элементов таблицы в виде одной повторяемой команды присваивания $S := S + a[i]$.

Пример 10.2. Построить более сложный алгоритм — алгоритм заполнения таблицы умножения (табл. 10).

```
алг  таблица умножения (цел таб произведение [1:9, 1:9])
     рез произведение
нач цел i, j
     i := 1
     пока i ≤ 9
     нц
          j := 1
          пока j ≤ 9
          нц
               произведение [i, j] := i · j
               j := j + 1
          кц
          i := i + 1
     кц
кон
```

Рассмотрим подробнее этот алгоритм. В нем есть две команды повторения, причем одна из команд повторения входит в состав другой команды.

Сначала исполнитель присваивает *i* значение 1 и переходит к выполнению первой внешней команды повторения. Так как условие цикла при этом значении соблюдается, он должен выполнить все команды, входящие в состав этого цикла.

Он присваивает *j* значение 1 и переходит к выполнению второй внутренней команды повторения. Команда повторения считается выполненной, когда перестает соблюдаться входящее в ее состав условие, т. е. когда *j* станет больше 9. Это произойдет после того, как входящая в ее состав серия команд будет выполнена 9 раз. В результате окажется заполненной первая строка таблицы *произведение.*

Поясним это.

Элементы первой строки таблицы *произведение* имеют индексы:

произведение [1,1], *произведение* [1,2], ..., *произведение* [1,9].

Команда присваивания, входящая в указанную серию, имеет вид:

произведение [*i, j*] := *i* ·*j*,

при этом *i* равно 1, а *j* меняется от 1 до 9, т. е. действительно первая строка таблицы *произведение* в результате выполнения второй (внутренней) команды повторения окажется заполненной.

После этого исполнитель продолжит выполнение первой (внешней) команды повторения. Он увеличит значение *i* на единицу так, что *i* станет равным 2. Проверит условие

внешней команды повторения ($i \leq 9$), снова присвоит j значение 1. Затем снова надо выполнять вторую (внутреннюю) команду повторения. Как мы знаем, в результате выполнения этой команды заполнится строка с номером i таблицы *произведение.*

Продолжая этот процесс, исполнитель заполнит всю таблицу *произведение.*

Вопросы

1. Какие величины встречаются в рассмотренных алгоритмах?

2. Что такое имя величины? Приведите примеры имен.

3. Что такое значение величины? Приведите примеры изменения значений величин в процессе исполнения алгоритма.

4. Приведите примеры постоянных и переменных величин.

5. Какие величины называются: а) аргументами; б) результатами алгоритма? Приведите примеры.

6. Какие типы величин используются в алгоритмическом языке? Приведите примеры.

7. Какие величины называются промежуточными? Приведите примеры.

8. Команды какого вида называются командами присваивания? Поясните на примерах их назначение.

9. Как выполняется команда присваивания?

10. Какие знаки отношений между величинами используются в алгоритмическом языке?

11. Чем различаются знаки отношений числовых и литерных величин?

12. Для чего в алгоритмическом языке служат знаки отношений между величинами?

13. Какие условия называются: а) простыми; б) составными? Приведите примеры.

14. Какие величины называются табличными? Приведите примеры.

15. Какая таблица называется: а) линейной; б) прямоугольной? Приведите примеры.

Упражнения[1]

1**.** Укажите тип следующих величин: 4; 4,0; «четыре»; «4,4»; «аргумент»; 0,0; «ноль»; 0.

2. Составьте и запишите алгоритм решения задачи: «В трех сосудах содержится вода. В первом сосуде содержится V_1, л воды температуры t_1 во втором — V_2 л температуры t_2, в третьем — V_3 л температуры t_3. Воду слили в один сосуд. Найдите объем и температуру t воды в этом сосуде (изменением объема воды при изменении температуры пренебречь)».

Исполните алгоритм для V_1, = 0,2 л, V_2 = 1 л, V_3 = 0,7 л; t_1,= 1°С, t_2 = 3°С, t_3 = 20°С.

3. Составьте и запишите алгоритм решения задачи. Принадлежит ли точка *A (x, y)* фигуре на плоскости (рис. 19)?

x	0,2	-0,1	-0,15	0,25	0,3	0,9	0	1	-0,5
y	0,3	-0,1	-0,85	-0,55	0,4	0,15	0	1	0,5

[1]Во всех упражнениях на составление алгоритмов исполнителю доступны все арифметические действия.

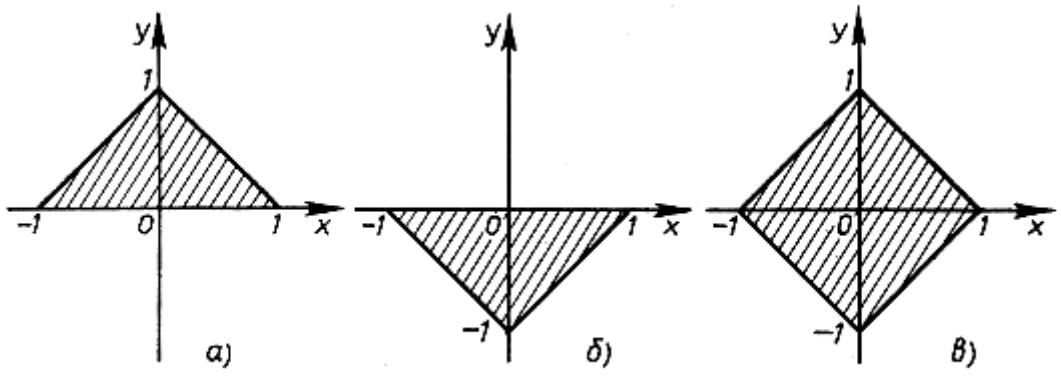

Рис. 19.

Исполните алгоритм для следующих координат точек:

4. Заполните таблицу промежуточных значений для алгоритмов:

а) нахождения наибольшего общего делителя чисел 161, 253;

б) решения уравнения $2x^2 + 3x - 5 = 0$.

5. Запишите с использованием промежуточных величин алгоритмы

а) вычисления выражения $y = \frac{a^2}{3} + \frac{a^2+4}{6} + \frac{\sqrt{a^2+4}}{4} + \frac{(\sqrt{a^2+4})^3}{4}$ при a = 11,7;

б) решения уравнения $ax^3 + bx = 0$.

6. Составьте и запишите следующие алгоритмы:

Даны площадь круга S_1, и площадь квадрата S_2. Определите, поместится ли: 1) круг в квадрате; 2) квадрат в круге. Проверьте алгоритм для нескольких значений S_1, и S_2.

7. Запишите алгоритм вычисления функции $y = f(x)$, заданной графиком (рис. 20), используя знаки отношения, приведенные на с. 37.

8. Запишите алгоритм вычисления факториала натурального числа n по формуле $n! = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot (n - 1) \cdot n$.

9. В одной команде разрешается использовать только одну операцию. Запишите алгоритмы вычисления выражений:

а) $y = \frac{2x^2 + \sqrt{x^3+1}}{2}$; б) $y = 2 \sin x^2 + 4 \cos x$;

в) $y = 4 \log (x^2 + 2\sqrt{\frac{1}{x} + x^3})$.

10. Объясните условие *кошка* = „кошка". Что является именем, что — значением величины?

11 Из скольких элементов состоят таблицы, описанные следующим образом:

а) **вещ таб** x [1:10];

б) **цел таб** x [4:5];

в) **вещ таб** x [100:100];

г) **вещ таб** x [1:N];

д) **цел таб** x [− 1:1];

е) **вещ таб** x [M:N]

ж) **цел таб** x [3:4,5:10];

з) **вещ таб** x [1:N, 5:10]?

12. Как задать вектор на плоскости, таблицей?

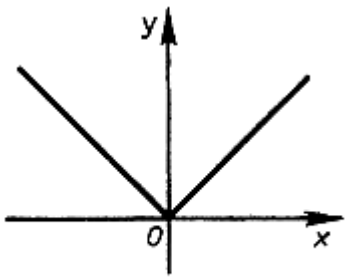

Рис. 20.

13. Запишите правило сложения векторов (рис.21) как действие с таблицами, их задающими.

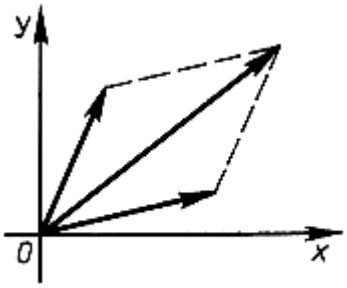

Рис. 21.

§ 4. ВСПОМОГАТЕЛЬНЫЕ АЛГОРИТМЫ

11. ПОНЯТИЕ ВСПОМОГАТЕЛЬНОГО АЛГОРИТМА

При построении новых алгоритмов могут использоваться алгоритмы, составленные раньше. Алгоритмы, целиком используемые в составе других алгоритмов, будем называть *вспомогательными* (или подчиненными) алгоритмами. Не исключено, что алгоритм, содержащий ссылку на вспомогательный, сам в определенной ситуации может оказаться в роли вспомогательного алгоритма. Использование ранее составленных алгоритмов при составлении новых находит широкое применение в практике алгоритмизации.

Все приведенные в учебнике алгоритмы, представляющие интерес для последующего использования, объединены в особый фонд — *библиотеку алгоритмов,* которая помещена в приложении.

Правила включения вспомогательных алгоритмов в основные в процессе их использования рассмотрим на примерах.

Пример 11.1. Алгоритм поиска большего из двух чисел α и β (обозначим его БИД) может быть записан следующим образом:

```
алг  БИД (вещ α, β, γ)
     арг α, β
     рез γ
нач  если α ≥ β
          то γ := α
          иначе γ := β
     все
кон
```

Рассмотрим более подробно использование вспомогательного алгоритма при составлении основного алгоритма. Записывая основной алгоритм, мы в какой-то момент обнаруживаем, что нам надо из двух величин *хну* выбрать большую и результат присвоить некоторой величине *z*. Чтобы воспользоваться вспомогательным алгоритмом, надо: 1) присвоить значение величины *x* аргументу α, значение величины *y* аргументу β; 2) исполнить" алгоритм БИД; 3) значение результата γ присвоить переменной *z*. Такое обращение к вспомогательному алгоритму в алгоритмическом языке можно записать в виде одной простой команды вызова вспомогательного алгоритма, который в нашем случае будет иметь вид БИД *(x, y, z)*.

После выполнения команды вызова выполняется следующая за ней в основном алгоритме команда.

Пример 11.2. Алгоритм поиска большего из трех чисел (обозначим его БИТ) можно составить в форме двукратного обращения к алгоритму БИД:

алг БИТ **(вещ** *a, b, c, y*)
арг *a, b, c*
рез *y*
нач **вещ** *z*
БИД (*a*, *b, z*)
БИД (*z, c, y*)
кон

При записи шагов исполнения основного алгоритма, содержащего в себе вызов вспомогательного алгоритма, в таблице значений записываются только величины основного алгоритма. При записи шага, состоящего в вызове вспомогательного алгоритма, записываются новые значения тех величин, которым присваиваются результаты данного исполнения вспомогательного алгоритма. Если есть необходимость проследить исполнение вспомогательного алгоритма, то для каждого обращения к нему составляется своя таблица значений.

Исполним алгоритм БИТ для аргументов 3, 9, 5. Составим таблицу значений (табл. 11).

Таблица 11

Шаги алгоритма	Аргументы			Промежуточная величина	Результат	Проверка условий
	a	*b*	*c*	*z*	*y*	
	3	9	5			
1				9		
2					9	

Из этой таблицы видно, что каждое исполнение вспомогательного алгоритма — это один шаг основного алгоритма. Проследим теперь, как исполняется алгоритм БИД для аргументов 3, 9. Составим таблицу значений (табл. 12).

Таблица 12

Шаги алгоритма	Аргументы		Результат	Проверка условий
	α	*β*	*γ*	
	3	9		
1				$3 \geq 9$ (нет)
2			9	

Аналогично можно было бы проследить исполнение алгоритма БИД для аргументов 9,5.

Пример 11.3. Составить алгоритм поиска наибольшего элемента в линейной таблице из *n* чисел *x* [1:*n*], используя алгоритм БИД как вспомогательный.

Наибольшее число в линейной таблице можно отыскать путем многократного применения алгоритма БИД к очередному элементу таблицы и результату предыдущего применения алгоритма БИД.

Запись алгоритма поиска наибольшего элемента линейной таблицы (НЭЛТ) на алгоритмическом языке:

алг НЭЛТ (**цел** *n,* **вещ таб** *x* [1:*n*], **вещ** *y*)
арг *n, x*
рез *y*
нач цел *i*
i : = 2; *y* := *x*[1]
пока *i* ≤ *n*
нц
БИД (*y, x* [*i*], *y*)*; i* := *i* + 1
кц
кон

12. ПОСЛЕДОВАТЕЛЬНОЕ ПОСТРОЕНИЕ АЛГОРИТМА

Метод вычленения вспомогательного алгоритма может быть успешно использован в процессе поиска и разработки нового алгоритма тогда, когда вспомогательный алгоритм еще неизвестен.

Процесс последовательного построения алгоритма может выглядеть следующим образом. Алгоритм сначала формулируется в самых «крупных» командах, при этом в записи алгоритма могут использоваться команды, выходящие за рамки возможностей исполнителя. Затем на каждом последующем этапе отдельные детали алгоритма уточняются, при этом недоступные исполнителю команды записываются как вызовы вспомогательных алгоритмов. После этого так же строятся вспомогательные алгоритмы. Процесс продолжается до тех пор, пока все алгоритмы не будут состоять из команд, понятных исполнителю. Такой способ построения алгоритма называется *методом последовательного уточнения.*

Рассмотрим этот метод на примере построения алгоритма вычисления степени $y = a^x$, где x — целое число, $a \neq 0$. В курсе алгебры степень с целым показателем определяется так:

$$a^x = \begin{cases} 1, \text{если } x = 0, \\ a^x, \text{если } x > 0, \\ \dfrac{1}{a^{-x}}, \text{если } x < 0. \end{cases}$$

Учитывая, что $\dfrac{1}{a^{-x}} = \left(\dfrac{1}{a}\right)^{-x}$, запишем искомый алгоритм:

алг степень с целым показателем **(вещ** *a*, **цел** *x,* **вещ** *y*)
арг *a, x*
рез *y*
нач
если *x* = 0
то *y*: = 1
иначе
если *x* :> 0
то вычислить $y = a^x$, x — целое положительное число
иначе вычислить $y = \left(\dfrac{1}{a}\right)^{-x}$, $-x$ —целое положительное число
все

все

кон

Замечаем, что в тексте алгоритма в двух местах предписывается выполнять похожие действия, а именно вычислять степень с одним натуральным показателем, но с различными основаниями. В этих местах может быть ссылка на один и тот же вспомогательный алгоритм вычисления степени с натуральным показателем, и тогда алгоритм приобретает следующий вид:

алг степень с целым показателем **(вещ** *a,* **цел** *x,* **вещ** *y*)

арг *a, x*

рез *y*

нач

если $x = 0$

то y: = 1

иначе

если $x > 0$

то степень (*a, x, y*)

иначе степень $\left(\frac{1}{a}, -x, y\right)$

все

все

кон

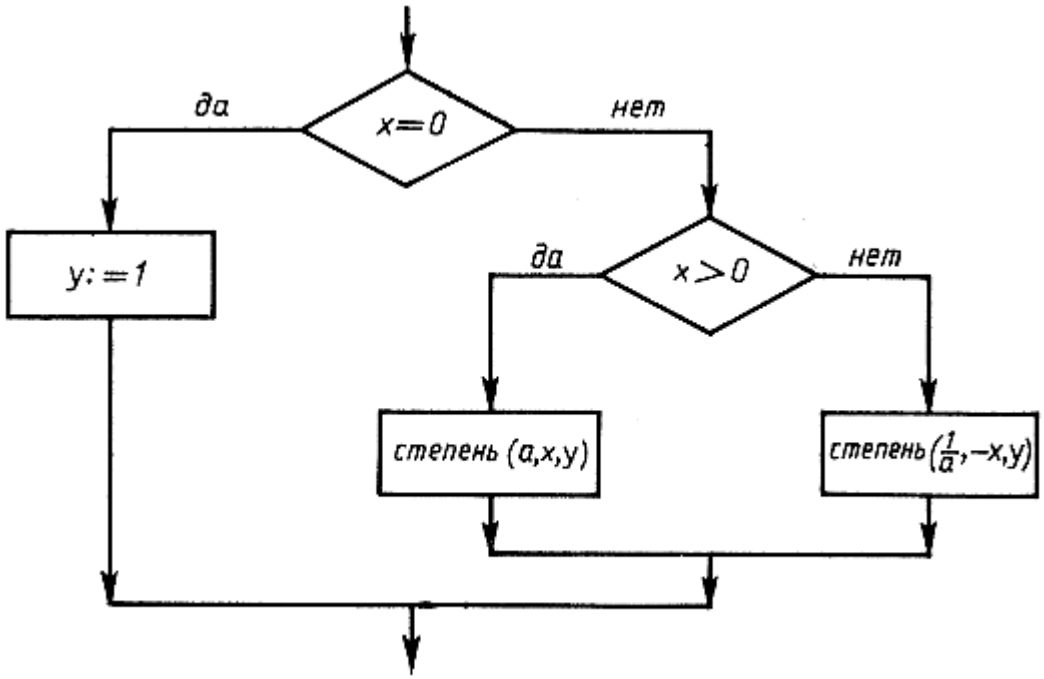

Рис. 22.

Схема этого алгоритма приведена на рисунке 22.

Теперь остается построить вспомогательный алгоритм вычисления степени с натуральным показателем $y = \alpha^n$. При вычислении α^n по формуле $\alpha^n = \underbrace{\alpha \cdot \alpha \cdot \ldots \cdot \alpha}_{n \text{ раз}}$ нужно проделать $(n - 1)$ умножение.

При составлении алгоритма удобно пользоваться формулой

$$\alpha^n = 1 \cdot \underbrace{\alpha \cdot \alpha \cdot \ldots \cdot \alpha}_{n \text{ раз}},$$

в которой число умножений равно показателю степени.

алг степень (**вещ** α, **нат** *n,* **вещ** *y*)

```
      арг α, n
      рез y
нач   цел i
      y := 1; i := 1
      пока i ≤ n
      нц
          y := y   α
      кц
кон
```

Рассмотрим процесс исполнения общего алгоритма вычисления степени с целым показателем $y = a^x$ для двух аргументов 5, —2 (табл. 13).

Таблица 13

Шаги алгоритма	Аргументы		Результат	Проверка условий
	a	*x*	*y*	
	5	-2		
1				-2 = 0 (нет)
2				-2 > 0 (нет)
3			0,04	

Проследим теперь исполнение вспомогательного алгоритма степень для аргументов 0,2, 2 (табл. 14).

Таблица 14

Шаги алгоритма	Аргументы		Промежуточная величина	Результат	Проверка условий
	α	*n*	*i*	*y*	
	0,2	2			
1				1	
2			1		
3					1 ≤ 2 (да)
4				0,2	
5			2		
6					2 ≤ 2 (да)
7				0,04	
8			3		
9					3 ≤ 2 (нет)

Вопросы

1. Для чего нужны вспомогательные алгоритмы?
2. Как записывается команда вызова вспомогательного алгоритма?
3. Каков порядок исполнения основного алгоритма при использовании вспомогательных алгоритмов?
4. В чем заключается метод последовательного уточнения при построении алгоритмов?
5. Может ли алгоритм, содержащий ссылку на вспомогательный, оказаться в роли вспомогательного?

Упражнения

1. Составьте алгоритм вычисления по формуле

$$y = \frac{|x+5|}{|3x^2 - x + 2| + 9},$$

используя вспомогательный алгоритм вычисления модуля числа МОД *(a, m)*.

2. В четырехугольнике *ABCD AB = x, BC = y, CD = z, AD = t, AC = d.* Составьте алгоритм вычисления площади четырехугольника, используя вспомогательный алгоритм ГЕРОН (a, *b, c, S)* вычисления площади треугольника по формуле Герона:

$$S = \sqrt{p(p-a)(p-b)(p-c)}, \text{ где } p = \frac{(a+b+c)}{2}.$$

3. Постройте алгоритм решения биквадратного уравнения $ax^4 + bx^2 + c = 0$, используя как вспомогательный алгоритм решение квадратного уравнения (с. 32).

4. Постройте алгоритм нахождения наименьшего общего кратного (НОК) двух натуральных чисел, используя алгоритм нахождения наибольшего общего делителя (с. 36) как вспомогательный. Исполните алгоритм для аргументов 14 и 26, 21 и 84.

У к а з а н и е . Для любых натуральных чисел *a, b, c* справедливо тождество $a \cdot b$ = НОД (a, *b*) · НОК (a, *b*).

5. Постройте алгоритм нахождения наибольшего общего делителя трех натуральных чисел, используя вспомогательный алгоритм нахождения наибольшего общего делителя двух чисел. Исполните построенный алгоритм для аргументов 24, 18, 12.

Раздел II
ПОСТРОЕНИЕ АЛГОРИТМОВ ДЛЯ РЕШЕНИЯ ЗАДАЧ

Как уже отмечалось, одна из основных целей курса информатики — научиться решать задачи с использованием ЭВМ. Решение задачи в этом случае требует предварительной работы и производится в несколько этапов. Часть этих этапов выполняется только человеком, а другая часть — человеком и ЭВМ.

Цель данного раздела — познакомиться с этапами решения задачи, предшествующими непосредственной работе на ЭВМ. Эти этапы заканчиваются построением алгоритма для решения задачи.

Следующий этап, связанный с исполнением алгоритма на ЭВМ, будет рассмотрен в X классе.

§ 5. ЭТАПЫ РЕШЕНИЯ ЗАДАЧИ С ИСПОЛЬЗОВАНИЕМ ЭВМ

Формулировка условия любой математической задачи начинается с описания исходных данных и предпосылок, которые излагаются на языке строго определенных математических понятий. Затем формулируется цель решения, т. е. указывается, что именно необходимо определить в результате решения задачи. Точную формулировку условия задачи называют также математической постановкой задачи, и решение любой задачи начинается именно с ее постановки. В результате постановки задачи выделяются исходные данные или аргументы и те величины, значение которых нужно определить, т. е. результаты. Постановка задачи является первым этапом ее решения.

При решении практических задач приходится иметь дело с реальными объектами — явлениями природы, физическими и производственными процессами, планами выпуска продукции и т. п. Для того чтобы поставить такую задачу, необходимо сначала описать объект исследования в математических терминах, т. е. построить его математическую модель, позволяющую свести исследование реального объекта к решению математической задачи. Степень соответствия модели реальному объекту проверяется практикой, экспериментом. Практика дает возможность оценить построенную модель и уточнить ее в случае необходимости.

Рассмотрим следующий пример. Пусть необходимо определить, каким образом будет двигаться тело, брошенное под углом к горизонту, и найти дальность бросания. Для решения задачи необходимо сделать некоторые допущения, при которых будет строиться математическая модель. В качестве таких допущений можно принять следующие предложения:

1. Пренебречь кривизной Земли, считая ее поверхность плоскостью.
2. Пренебречь движением Земли.
3. Считать ускорение свободного падения g постоянным.
4. Пренебречь сопротивлением воздуха.

Из курса физики известно, что при таких предположениях движение тела, брошенного под углом α к горизонту с начальной скоростью v_0, описывается системой уравнений:

$$\begin{cases} x = v_0 \cdot \cos\alpha \cdot t, \\ y = v_0 \cdot \sin\alpha \cdot t - g\dfrac{t^2}{2}. \end{cases} \qquad (1)$$

При этом предполагается, что эти уравнения описывают движение тела в неподвижной системе координат, начало которой совмещено с точкой бросания, ось x направлена вдоль поверхности Земли в сторону бросания, а ось y — вертикально вверх.

Эти уравнения являются математической моделью, описывающей движение тела. Исходя из этой модели, мы получим ответ на интересующий нас вопрос о дальности бросания.

Выразив из первого уравнения t и подставив его во второе уравнение, получим уравнение траектории:

$$y = x \operatorname{tg} \alpha - x^2 \frac{g}{2v_0^2 \cos^2 \alpha}$$

Эта траектория приведена на рисунке 23.

Рис. 23.

Решая квадратное уравнение $x \operatorname{tg} \alpha - x^2 \frac{g}{2v_0^2 \cos^2 \alpha} = 0$, можно определить точки пересечения этой траектории с осью x : $x_1 = 0$ — точка бросания и $x_2 = \frac{v_0^2}{g} \sin 2\alpha$ — точка падения.

Создание математической модели исследуемого явления позволяет поставить задачу математически. В нашем примере задача формулируется так: «Пусть движение тела, брошенного под углом а к горизонту с начальной скоростью v_0, описывается системой уравнений (1), и пусть заданы значения v_0 и α. Требуется определить дальность (l) полета тела».

Исходными предложениями задачи являются уравнения (1), а исходными данными (аргументами) — значения v_0 и a. Результатом решения является значение l.

После постановки задачи начинается поиск метода ее решения. При использовании ЭВМ для решения задачи строится алгоритм. Таким образом, следующим этапом решения задачи является построение алгоритма. В рассмотренном примере алгоритм строится достаточно просто:

1) выразить t из первого уравнения системы (1) через x
2) подставить полученное выражение во второе уравнение системы (1)
3) положить y равным 0
4) решить полученное квадратное уравнение
5) взять в качестве ответа l ненулевой корень этого уравнения.

Такой алгоритм может быть записан на алгоритмическом языке или в виде схемы.

Теперь необходимо записать алгоритм в виде, доступном исполнению на ЭВМ. Так как ЭВМ «понимает» специальный язык, называемый языком программирования, то алгоритм должен быть записан на таком языке. Этот этап решения задачи называется записью алгоритма на языке программирования. Этот и последующие этапы решения задачи будут рассматриваться позднее, в X классе.

Следующий этап — исполнение алгоритма с помощью ЭВМ. Этот этап завершается получением результата решения.

Наконец, завершающий этап решения задачи — анализ полученных результатов. Этот анализ проводится с целью определения, насколько точно полученные результаты соответствуют реальности. Анализ результатов помогает уточнить модель, если это необ-

ходимо. Кроме того, при решении задачи могут быть получены результаты, которые противоречат смыслу задачи. Например, если математическая модель, описывающая количество изделий, выпускаемых за день некоторым предприятием, представляет собой квадратное уравнение, один из корней которого отрицателен, то такой результат противоречит смыслу задачи и должен быть отброшен.

Во многих случаях анализ результатов задачи тоже проводится на ЭВМ.

Итак, решение задачи с помощью ЭВМ разбивается на следующие этапы:

— постановка задачи, включающая построение математической модели и выделение аргументов и результатов;

— построение алгоритма;

— запись алгоритма на языке программирования;

— реализация алгоритма с помощью ЭВМ;

— анализ полученных результатов.

В следующих параграфах будут рассмотрены отдельные этапы решения задачи, причем основное внимание будет обращено на построение алгоритма для решения задач.

Упражнения

1. Постройте математическую модель движения тела, брошенного вертикально вверх.

2. Постройте математическую модель движения тела массой m по наклонной плоскости с углом α. Трением пренебречь.

3. Постройте математическую модель движения тела массой m по окружности радиуса R.

§ 6. АЛГОРИТМЫ ДЛЯ РАБОТЫ С ТАБЛИЧНЫМИ ВЕЛИЧИНАМИ

Многие задачи, которые решаются с помощью ЭВМ, связаны с обработкой больших объемов информации. Для удобства обработки информация часто сводится в линейные или прямоугольные таблицы. Тогда обработка состоит из поиска в таблице нужного элемента, записи в таблицу новых элементов, изменения порядка элементов в таблице и т. п. Преимущества ЭВМ перед другими исполнителями алгоритма и состоят в том, что ЭВМ может решать задачи по обработке таблиц, содержащих десятки и сотни тысяч элементов, достаточно быстро и тем самым освобождать человека от утомительной и непродуктивной (рутинной) работы.

Рассмотрим задачу поиска заданного элемента в линейной таблице. Будем предполагать, что элементы таблицы являются числами.

Итак, предположим, что имеется линейная таблица, имеющая имя *„ключ",* элементы которой — вещественные числа:

вещ таб *ключ* [1 *:n*].

Задача состоит в том, чтобы определить, имеется ли в этой таблице элемент, значение которого совпадает с заданным числом *L.* Если такой элемент существует, то необходимо в качестве ответа получить порядковый номер этого элемента. В противном случае необходимо выдать номер, равный нулю. Будем просматривать подряд все элементы таблицы, начиная с первого, и сравнивать их значения с *L.* Такой просмотр надо продолжать до тех пор, пока не будет найден элемент таблицы, равный *L*, или до тех пор, пока не будет просмотрена вся таблица (если такого элемента не существует).

Опишем алгоритм поиска в таблице. Он будет использовать величины *i* (число просмотренных элементов) и *k* (номер искомого элемента).

Вначале *i* будет равно 0, так как мы еще не просмотрели ни одного элемента таблицы. Затем *i* будет возрастать: мы будем просматривать все новые и новые элементы таблицы, пытаясь найти среди них элемент, равный *L.* Если такой элемент обнаружится, то *k*

станет равным его номеру, а до тех пор k будет равно 0. Если числа L в таблице нет, то k так и останется равным нулю. Таким образом, значение переменной k будет сигнализировать об успехе ($k > 0$) или неуспехе ($k = 0$) поиска.

```
алг  поиск (цел n, вещ таб ключ [1:n], вещ L, цел k)
     арг ключ, n, L
     рез k
нач
     цел i
     i := 0; k := 0
     пока i ≠ n и k = 0
     нц
          i := i + 1
          если   ключ [ i] = L
                 то k: = i
          все
     кц
кон
```

Пояснение. Посмотрим, как будет выполняться наш алгоритм. Основная его часть — это команда повторения. Но еще до нее величинам i, k присваивается значение 0. Это соответствует смыслу этих переменных, ведь ни одного элемента еще не просмотрено и число L в таблице не найдено. Что происходит в команде повторения? Входящая в нее серия выполняется многократно и заканчивается, когда мы доходим до конца таблицы *($i = n$)* или находим нужный элемент *(k* стало положительным).

Пусть мы уже i раз выполнили серию, причем $i \neq n$, и среди элементов *(ключ* [1], ..., *ключ* [i]) нет L (а значит, $k = 0$). Тогда мы начинаем выполнение серии еще раз. В первой команде серии i увеличивается на 1, т. е. становится равным номеру элемента, который мы должны сейчас просмотреть. Если этот элемент равен L, т. е. мы нашли нужный элемент, то k становится равным номеру этого элемента. Выполнение серии закончено, и больше она выполняться не будет. Если этот элемент отличен от L, то значение $k = 0$ сохраняется, серия закончена и мы снова возвращаемся к проверке условия цикла.

Поиск нужного элемента может быть ускорен, если элементы таблицы упорядочены по своей величине, например в порядке возрастания своих значений или в алфавитном порядке. Так, если фамилии в таблице расположены в алфавитном порядке, то при поиске нужной фамилии можно отыскать в таблице нужную первую букву и просматривать только те фамилии, которые начинаются с этой буквы.

Упорядочение таблицы приходится выполнять так часто, что, по оценкам специалистов, больше четверти времени работы всех ЭВМ приходится на выполнение этого процесса, который называют также сортировкой. По этой причине для решения задачи сортировки с помощью ЭВМ было придумано много различных алгоритмов. Цель создания таких алгоритмов — обеспечить достаточно быстрое решение задачи сортировки для разного типа таблиц.

Рассмотрим один из наиболее простых алгоритмов сортировки, идея которого состоит в следующем. Будем сначала просматривать всю таблицу подряд с первого до последнего элемента и найдем элемент, имеющий наименьшее значение. Поменяем местами найденный элемент и первый элемент таблицы. В результате на первом месте окажется элемент, имеющий наименьшее значение. Теперь будем просматривать все элементы таблицы, начиная со второго, и снова найдем среди них элемент, имеющий наименьшее значение. Поменяем местами этот элемент со вторым элементом таблицы, в результате чего на втором месте будет стоять элемент, имеющий наименьшее значение среди оставшихся. Продолжим процесс. В последний раз останется просмотреть только два последних элемента таблицы, выбрать среди них элемент с наименьшим значением и поставить его на

предпоследнее место в таблице. После этого таблица будет упорядочена в порядке возрастания значений составляющих ее элементов.

Описанную задачу сортировки можно разбить на две задачи: задачу поиска в таблице элемента с наименьшим значением и задачу упорядочения таблицы путем перестановки найденного элемента с очередным (первым, вторым и т. д.) элементом таблицы. Рассмотрим каждую из этих задач.

Пусть задана линейная таблица *A,* элементы которой нумеруются от *K* до *N (K < N).*

Построим алгоритм поиска наименьшего элемента линейной таблицы.

В этом алгоритме элементы таблицы будут просматриваться по порядку, от первого до последнего. Для этого будет использоваться переменная *i* — номер первого непросмотренного элемента таблицы. Кроме *i,* будут использоваться еще две переменные:

МИН — «минимум из уже просмотренных элементов»;

L — «номер элемента, минимального среди уже просмотренных».

Работа алгоритма начинается с серии команд, в которой мы присваиваем переменной *МИН* значение начального элемента таблицы. Этот элемент уже просмотрен, и мы берем *K* в качестве значения для *l*; просмотрен один элемент *A* [*K*], и он минимален среди просмотренных. Берем K + 1 в качестве значения для *i* — номера первого непросмотренного элемента.

Выполнение серии, входящей в команду повторения, происходит следующим образом. Если *МИН* оказался больше, чем очередной элемент *A* [*i*], нужно сменить *МИН*, и мы это делаем, присваивая ему значение *A* [*i*]; соответственно с этим и значение *l* меняется, теперь оно равно *i*. Так как элемент с номером *i* уже просмотрен, серия завершается увеличением номера *i* на 1:

$$i := i + 1.$$

Работа алгоритма завершается, когда вся таблица просмотрена и номер ее минимального элемента найден. Этот номер и является результатом.

Алгоритм имеет следующий вид:

```
алг  МИНЭЛЕМЕНТ (цел K, N, вещ таб A[K : N], цел l)
     арг A, K, N
     рез l
нач  цел i, вещ МИН
     МИН := A[K]; l := K; i:=K + 1
     пока i < N
     нц
         если МИН := A[i]
             то МИН := A[i]; l := i
         все
         i := i + 1
     кц
кон
```

Построим теперь алгоритм упорядочения линейной таблицы, используя алгоритм МИНЭЛЕМЕНТ в качестве вспомогательного. Задача упорядочения таблицы формулируется следующим образом. Пусть задана линейная таблица *C,* элементы которой нумеруются от *n* до *M* (*n* <.*M*):

вещ таб *C* [*n : M*]

Необходимо переставить элементы этой таблицы так, чтобы они шли в порядке возрастания.

Идея алгоритма упорядочения состоит в следующем. Применив алгоритм МИНЭЛЕМЕНТ к таблице *C* [*n* : *M*], мы определим номер *l* элемента этой таблицы, имеющего наименьшее значение. После этого поменяем значения элементов C [*n*] и *C* [*I*]. Тогда на *n*-м месте таблицы *C* (напомним, что таблица *C* начинается с элемента с номером

n) окажется самый маленький по величине элемент. На следующем шаге применяем алгоритм МИНЭЛЕМЕНТ к таблице *C* [*n* + 1 : *M*] и снова определяем номер *l* минимального элемента этой таблицы. Поменяем местами элементы *C* [*n* + 1] и *C* [*l*], тогда на *n* + 1-м месте окажется самый маленький из оставшихся элементов. Далее будем применять алгоритм МИНЭЛЕМЕНТ к таблицам *C* [*n* + 2 : *M]*, *C* [*n* + 3 *: M*], ...,*C* [*M* − 1, *M*] и менять местами элементы *C [n* + 2] и *C* [*l*], *C* [*n* + 3] и C [*l*] и, наконец, *C [M* − 1] и *C* [*l*]. В результате таблица *C* окажется упорядоченной.

На алгоритмическом языке алгоритм упорядочения будет выглядеть следующим образом:

алг упорядочение (**цел** *n, M,* **вещ таб** *C*[*n : M*])
арг *C*, *n*, *M*
рез *C*
нач **цел** *i*, *l*, **вещ** *R*
i := *n*
пока *i* < *M*
нц
МИНЭЛЕМЕНТ (*i*, *M*, *C*, *l*)
R := *C* [*i*]
C [*i*] := *C* [*l*]
C [*l*] := *R*
i := *i* + 1
кц
кон

Заметим, что обмен местами двух элементов таблицы С осуществляется с помощью трех команд:

$$R := C[i]$$
$$C[i] := C[l]$$
$$C[l] := R$$

Это объясняется тем, что как только выполнится команда *C* [*i*] := *C* [*l*], то «старое» значение *C* [*i*] будет «забыто», поэтому предварительно нужно запомнить это значение, присвоив его вспомогательной переменной *R*. Тогда третья команда обеспечит присваивание этого значения переменной *C* [*l*].

В заключение рассмотрим задачу о вычислении значений элементов в некоторой последовательности, которые сводятся в линейную таблицу, удобную для дальнейшего использования.

В качестве примера рассмотрим получение так называемых чисел Фибоначчи, названных по имени итальянского математика средневековья (Леонардо Пизанского). Эти числа образуют бесконечную последовательность, причем каждое число в ней получается по следующему правилу: первые два числа (обозначим их F_1, и F_2) равны 1, т. е. $F_1 = F_2 = 1$, а каждое следующее число, начиная с третьего, равно сумме двух предыдущих, т. е. $F_3 = F_1 + F_2$, $F_4 = F_3 + F_2$ и т. д., вообще $F_i = F_{i-1} + F_{i-2}$ для любого $i \geq 3$.

Построим алгоритм получения *N* чисел Фибоначчи, которые будем рассматривать как элементы линейной таблицы *F* [1 *:.N*]. Этот алгоритм на алгоритмическом языке имеет вид:

алг Фибоначчи **(цел** *N,* **цел таб** *F*[1 : *N*])
арг *N*
рез *F*
нач **цел** *i*
F[1] := 1
F[2] : = 1
i :=3
пока *i* ≤ *N*

нц

$F[i] := F[i - 1] + F[i - 2]$

$i := i + 1$

кц

кон

Упражнения

1. Постройте алгоритм определения номера элемента с максимальным значением из линейной таблицы $B[1:n]$ в предположении, что значениями элементов таблицы B являются целые числа.

2. Используя алгоритм упражнения 1 в качестве вспомогательного, постройте алгоритм упорядочения элементов линейной таблицы по убыванию их значений.

3. Постройте алгоритм для подсчета числа положительных элементов в линейной таблице A.

4. Задана линейная упорядоченная таблица, состоящая из целых чисел. Постройте алгоритм, с помощью которого значения всех элементов таблицы, меньшие 100, заменялись бы числом 100.

5. Постройте алгоритм, определяющий, сколько раз число 10 встречается среди элементов линейной таблицы A [1:1000], состоящей из целых чисел.

6. Постройте алгоритм для определения среднего арифметического n чисел, сведенных в линейную таблицу A [1:n]. (Напомним, что среднее арифметическое n чисел вычисляется по формуле $c = \frac{a_1 + a_2 + ... + a_n}{n}$.)

7. Постройте алгоритм для определения суммы всех положительных чисел, входящих в линейную таблицу $x[1:n]$.

8. Постройте алгоритм:

а) переписи значений элементов одной таблицы в другую;

б) заполнения линейной таблицы числом 0;

в) заполнения i-й строки прямоугольной таблицы числом 0.

§ 7. ПОСТРОЕНИЕ АЛГОРИТМОВ ДЛЯ РЕШЕНИЯ ЗАДАЧ ИЗ КУРСА МАТЕМАТИКИ

Понятие алгоритма оказывается очень важным при решении многих математических задач. При этом объекты, действия над которыми описывает алгоритм, могут быть самыми разными. Построив алгоритм решения задачи и записав его на алгоритмическом языке, можно многократно применять его, уже не вникая в условия задачи.

Пример 1. Алгоритм вычисления значения многочлена.

Пусть дан многочлен $x^4 + 2x^3 + 3x^2 - 5x - 15$. Требуется вычислить значение многочлена при $x = 2$.

Существует общепринятый способ вычисления значений многочлена: найти числа x^2, x^3 и x^4 при $x := 2$, затем вычислить $2x^3$, $3x^2$ и $5x$ и, наконец, путем сложения и вычитания получить значение многочлена. При вычислении многочлена потребовалось десять действий, так как $x^2 = x \cdot x$, $x^3 = x \cdot x^2$, $x^4 = x \cdot x^3$.

Вычисление значения многочлена может быть упрощено. Для этого перепишем заданный многочлен в виде

$$x^4 + 2x^3 + 3x^2 - 5x - 15 = (((x + 2)x + 3)x - 5)x - 15$$

и будем выполнять операции в порядке, определяемом скобками. Тогда потребуется семь арифметических действий. Этот способ вычисления значения многочлена называется схемой Горнера.

Объясним, как, применять его к произвольному многочлену.

Пусть задан многочлен

$$a_0x^3 + a_1x^2 + a_2x + a_3.$$

Преобразуем его к виду

$$((a_0x + a_x)\,x + a_2)\,x + a_3$$

и будем вычислять последовательно такие величины:

$a_0 \cdot x$	прибавляем a_1,
$a_0x + a_1$	умножаем на x
$(a_0x + a_1)x$	прибавляем a_2
$(a_0x + a_1)x + a_2$	умножаем на x
$((a_0x + a_1)x + a_2)x$	прибавляем a_3
$((a_0x + a_1)x + a_2)x + a_3$	

Последняя величина и будет искомым значением многочлена. Так можно вычислить значение произвольного многочлена

$$a_0x^n + a_1x^{n-1} + \ldots + a_{n-1}\,x + a_n,$$

преобразовав его к виду

$$(((a_0x + a_1)x + a_2)x + \ldots + a_{n-1})x + a_n.$$

Алгоритм вычисления значения многочлена n-й степени выглядит следующим образом:

алг схема Горнера **(цел** n, **вещ** x, **вещ таб** $a[0{:}n]$, **вещ** y**)**
арг n, a, x
рез y
нач **цел** i
$i := 0$; $y := a[0]$
пока $i \neq n$
нц
$i := i+1$
$y := y \cdot x + a[i]$
кц
кон

П р и м е р 2. Алгоритм приближенного вычисления площадей. Пусть график функции $f(x)$ имеет вид, изображенный на рисунке 24.

Пусть требуется приближенно вычислить площадь фигуры, ограниченной графиком функции $f(x)$ и прямыми $x = a$, $x = b$, $y = 0$. Эта фигура называется криволинейной трапецией и отмечена на рисунке штриховкой.

Идея алгоритма вычисления площади криволинейной трапеции состоит в следующем. Разобьем отрезок $[a, b]$ на n равных отрезков точками $a = x_0 < x_1 < x_2 < \ldots < x_n = b$ и на каждом из полученных отрезков построим прямоугольник, одной стороной которого будет отрезок $[x_i, x_{i+1}]$, а другой — отрезок, длина которого равна $f(x_i)$. Случай при $n = 4$ показан на рисунке 25.

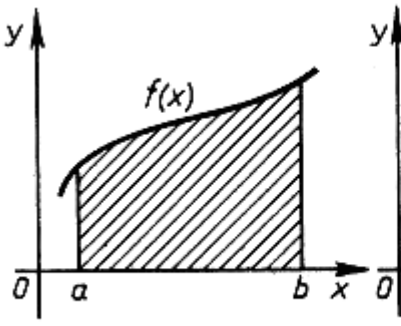

Рис. 24.

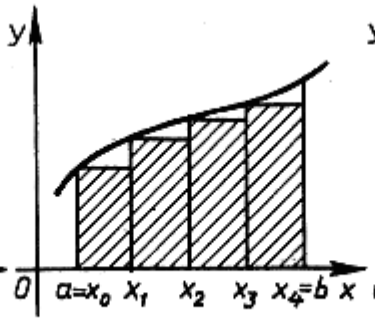

Рис. 25.

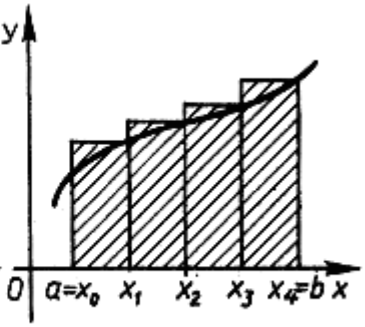

Рис. 26.

Площадь криволинейной трапеции можно приближенно считать равной сумме площадей заштрихованных прямоугольников. Можно получить другую картину (см. рис. 26),

если в качестве второй стороны прямоугольников выбирать отрезки, длины которых равны $f(x_{i+1})$, $i = 0, 1, 2, ..., n - 1$.

Ясно, что если увеличить число отрезков $[x_i, x_i+1]$, т. е. отрезок $[a, b]$ разбить на большее число равных отрезков, то сумма их площадей все с большей точностью будет совпадать с площадью криволинейной трапеции. Значит, точность вычисления площади криволинейной трапеции определяется величиной n.

Площадь каждого прямоугольника можно вычислить так. Одна из сторон прямоугольника, построенного на отрезке $[x_i..x_{i+1}]$, равна $h = \frac{b-a}{n}$, а вторая — $f(x_i)$ либо $f(x_{i+1})$. Поэтому в первом случае площадь «левого» прямоугольника $\frac{b-a}{n} \cdot f(x_i)$, а во втором случае площадь «правого» прямоугольника $\frac{b-a}{n} \cdot f(x_{i+1})$.

Площадь криволинейной трапеции в первом случае приближенно равна сумме «левых» прямоугольников:

$$\frac{b-a}{n} \cdot f(x_0) + \frac{b-a}{n} f(x_1) + ... + f(x_{n-1}) = \frac{b-a}{n}(f(x_0) + f(x_1) + ... + f(x_{n-1})),$$

а во втором случае — сумме «правых» прямоугольников:

$$\frac{b-a}{n}(f(x_1) + f(x_2) + ... + f(x_n)),$$

причем, чем больше n, тем больше точность вычисления площади криволинейной трапеции.

Обозначив величину площади S, запишем алгоритм приближенного вычисления площади криволинейной трапеции для случая «левых» прямоугольников.

алг площадь (**вещ**, *a b, S,* **цел** *n)*
арг *a, b, n*
рез *S*
нач **вещ** *h, x,* **нат** *i*
$h := \frac{b-a}{n}$; $S := 0$; $x := a$; $i := 1$
пока $i \le n$
нц
$S := S + f(x) \cdot h$
$x := x + h$
$i := i + 1$
кц
кон

Для решения квадратного уравнения $ax^2 + bx + c = 0$ используются известные формулы вычисления корней: $x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$. Соответствующие алгоритмы были рассмотрены выше. Это алгоритмы, основанные на так называемых точных методах решения. Не всегда, однако, для решения уравнения можно применить точный метод. Например, для уравнения $2x - \cos x := 0$ уже не существует равносильных преобразований, приводящих к выражению для переменной x. На практике часто встречаются такие уравнения.

Поэтому широко используются приближенные методы решения уравнений, позволяющие тем не менее получать ответ с любой желаемой степенью точности. Особенно широкое применение эти методы получили в связи с применением вычислительных машин.

Рассмотрим этот вопрос сначала в общем виде. Пусть имеется уравнение с одной переменной $f(x) — 0$, где $f(x)$ — некоторая непрерывная функция. С геометрической точки

зрения корень уравнения $f(x) = 0$ — это точка пересечения графика функции $f(x)$ с осью x. Корней может быть не один, а несколько, тогда на графике будет соответствующее число точек пересечения (рис. 27).

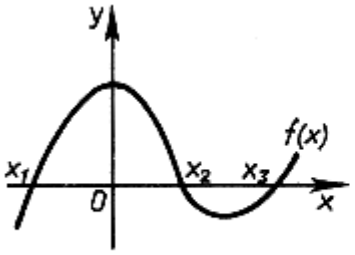

Рис. 27.

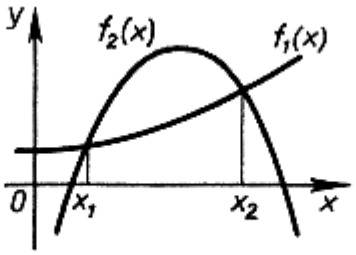

Рис. 28.

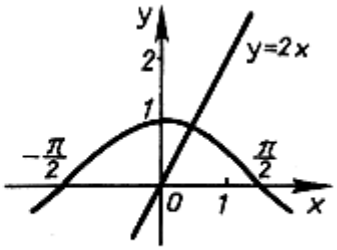

Рис. 29.

Очевидно, что графический метод может в отдельных случаях использоваться как метод грубого решения уравнения. Эта задача значительно облегчается, если удается преобразовать уравнение $f(x) = 0$ к равносильному виду $f_1(x) = f_2(x)$ таким образом, чтобы графики функции $f_1(x)$ и $f_2(x)$ могли быть легко построены. В этом случае корень уравнения — это абсцисса точек пересечения графиков функций $f_1(x)$ и $f_2(x)$ (рис. 28). Графическим путем можно достаточно точно определить отрезки, в каждом из которых содержится один корень уравнения. Это так называемый этап отделения корня.

П р и м е р 3. Отделить корень уравнения $2x - \cos x = 0$.

Преобразуем уравнение к виду $2x - \cos x$ и построим графики функций $f_1(x) = 2x$ и $f_2(x) = \cos x$; (рис. 29). Из рисунка видно, что уравнение $2x - \cos x = 0$ имеет единственный корень, принадлежащий отрезку [0; 1].

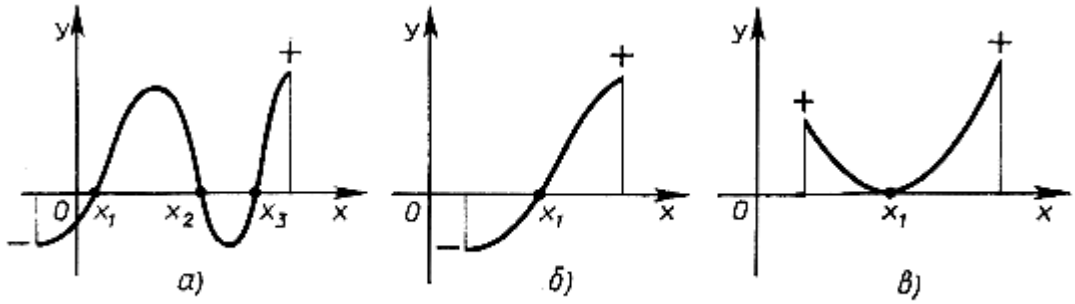

Рис. 30.

Предположим, что непрерывная функция $f(x)$ имеет на концах некоторого отрезка значения разных знаков. В этом случае она обязательно имеет на этом отрезке хотя бы один корень (рис. 30, *а, б)*. Как видно из рисунка 30, *а,* функция может иметь несколько корней. Если же она возрастает на этом отрезке (или убывает на нем), то корень единственный (рис. 30, б). Приведем один из методов отыскания корня — метод половинного деления. Этот метод применим в том случае, когда функция принимает значения разных знаков на концах некоторого отрезка. Он позволяет найти один из корней функции на этом отрезке. Если их несколько (как на рис. 30, *а)* или если значения функции на концах отрезка одного знака (рис. 30, *в),* то метод половинного деления не позволяет найти все корни функции на данном отрезке. Тем не менее он часто оказывается полезным.

Пусть функция $f(x)$ непрерывна на отрезке $[a; b]$ и принимает на его концах значения разных знаков. Составим алгоритм вычисления корня уравнения $f(x) = 0$ с заданной точностью.

Будем последовательно сужать отрезок $[a; b]$, пользуясь следующим методом. Разделим отрезок $[a; b]$ пополам точкой $c = \dfrac{(a+b)}{2}$ и вычислим значение $f(c)$. После этого от-

бросим ту часть отрезка [a; *b*], на которой функция *f*(*x*) знака не меняет (на рис. 31 такой частью является отрезок [*a*; *c*]). Процесс будем продолжать до тех пор, пока длина отрезка, содержащего корень, не станет меньше 2ε.

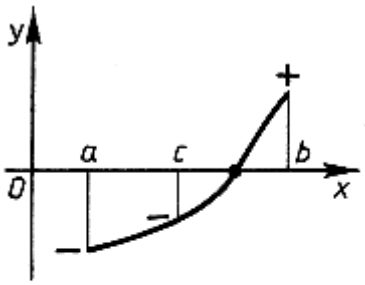

Рис. 31.

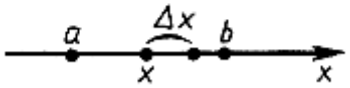

Рис. 32.

Действительно, если в этом случае в качестве корня Δ*x* взять середину отрезка $\frac{a+b}{2}$ (см. рис. 32), то допущенная при этом погрешность *x* не будет превышать ε (на рис. 32 точное значение корня обозначено *x*).

Запись алгоритма уточнения корня уравнения *f*(*x*) на отрезке [*a*; *b*] с заданной точностью ε на алгоритмическом языке приведена в библиотеке алгоритмов (приложение).

Пользуясь этим алгоритмом, можно искать корни уравнений с помощью калькулятора.

Упражнения

1. Постройте алгоритм «прямого» вычисления значения многочлена $2x^3 - 3x + 5$ при $x = 6$, не пользуясь схемой Горнера.

2. Вычислите значение многочлена $3x^4 + 2x^2 + x + 1$ при $x = 0{,}1$ «прямым» алгоритмом и по схеме Горнера. Сравните полученные результаты. Оцените точность вычисления каждым методом.

3. Используя схему Горнера, составьте таблицу значения многочлена:

а) $2x^4 - 1{,}4x^3 - 3{,}5x^2 + 4$ на отрезке [1; 2] с шагом 0,2 (имеет ли данная функция точки экстремума на отрезке [1; 2] ?);

б) $4x^3 - 0{,}7x^2 + x - 2{,}18$ на отрезке [0; 4] с шагом 0,5;

в) $x^3 — 2{,}1x^5 + 0{,}2x^2 - 4$ на отрезке [−2; 2] с шагом 0,4.

4. Решите квадратное уравнение:

а) $3x^2 - 1{,}8x - 2{,}4 = 0$; б) $6{,}2x^2 + 12{,}1x + 3{,}81 = 0$

и найдите приближенные значения корней с точностью до 0,001.

5. Стороны прямоугольника *a* и *b* удовлетворяют соотношению $\frac{a+b}{b} = \frac{b}{a}$, которое называется «золотым сечением». В этом случае отношение сторон прямоугольника $q = \frac{b}{a}$ удовлетворяет квадратному уравнению $q^2 - q - 1 = 0$ (докажите). Найдите приближенно с точностью до 0,01 положительный корень этого уравнения.

6. Через сколько секунд упадет на Землю камень, брошенный с высоты 2 м вверх с начальной скоростью 10 м/с? Напомним, что высота в этом случае меняется по закону:

$$h(t) = 2 + 10t - \frac{g}{2}t^2, \text{ где } g = 9{,}8 \text{ м/с}^2.$$

7. Вычислите приближенно площадь одной «арки» синусоиды, применяя:

а) метод «правых» прямоугольников;

б) метод «левых» прямоугольников.

8. Вычислите площадь криволинейной трапеции, ограниченной графиком функции $y = \sqrt{x+1}$ и прямыми $y = 0$, $x = 0$, $x = 3$, методом «левых» и «правых» прямоугольников. Сравните результаты.

9. Уточните корни уравнений, пользуясь калькулятором:

а) $\sin x - 0{,}2x = 0$ (наименьший положительный корень);

б) $\cos x = x^2$;

в) $x - 10 \sin x = 0$ (наименьший положительный корень);

г) $\lg x = (x - 2)^2$.

10. Постройте алгоритм решения биквадратного уравнения, используя алгоритм решения квадратного уравнения в качестве вспомогательного.

11. Постройте алгоритм решения уравнения $\sqrt[4]{l+x} + \sqrt[4]{l-x} = m$, используя вспомогательные алгоритмы из биб лиотеки. При построении алгоритма используйте метод последовательного уточнения.

12. Постройте алгоритм решения системы уравнений, используя вспомогательный алгоритм из библиотеки:

$$\begin{cases} x^2 \cdot y + x \cdot y^2 = 30, \\ \dfrac{1}{x} + \dfrac{1}{y} = \dfrac{5}{6}. \end{cases}$$

§ 8. ПОСТРОЕНИЕ АЛГОРИТМОВ ДЛЯ РЕШЕНИЯ ЗАДАЧ ИЗ КУРСА ФИЗИКИ

В данном параграфе рассматривается несколько физических задач. В каждой из них речь идет о каком-либо физическом явлении, для которого имеется математическая модель. По исходным данным задачи, производя определяемые моделью вычисления, можно найти требуемый ответ. Вычисления, как это обычно и бывает в практических задачах, имеют довольно большой объем. Правила их выполнения формулируются в виде алгоритма на алгоритмическом языке. На основе этого алгоритма может быть разработана программа для компьютера. Вы, производя вычисления по алгоритму, можете использовать калькулятор.

Модели, которые мы сейчас будем рассматривать, можно назвать вычислительными. Они не дают явной формулы, по которой можно сразу найти требуемые значения искомых величин. Вместо этого есть алгоритм, позволяющий приближенно вычислять эти искомые значения. При этом вычислении промежутки времени, интервалы длины, участки поверхности разбиваются на мелкие доли, фрагменты. Вычисления производятся отдельно для каждого участка. Объем вычислений бывает очень большим. Вычислительные модели стали особенно важны после появления ЭВМ, которые могут реально осуществить огромный объем вычислений, требуемый для таких моделей. В примере 1 речь идет о важной вычислительной задаче, возникающей в физических исследованиях,— определении значений параметров по результатам измерений.

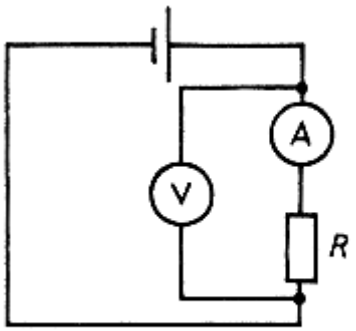

Рис. 33.

Пример 1. Пусть задана электрическая цепь (рис. 33). Измеряя напряжение U вольтметром, а силу тока I амперметром, можно найти сопротивление R. Результаты будут более точными и надежными, если повторим измерения несколько раз, а еще лучше — проведем измерения при нескольких различных значениях U (меняя источник тока в цепи).

Предположим, что при таком эксперименте для неизвестного резистора получились следующие результаты (табл. 15).

Таблица 15

U, B	1	2	3	4	5	6	7
I, A	0,010	0,018	0,031	0,042	0,050	0,061	0,072

Нужно найти сопротивление этого резистора. Нанесем эти точки на график (рис. 34).

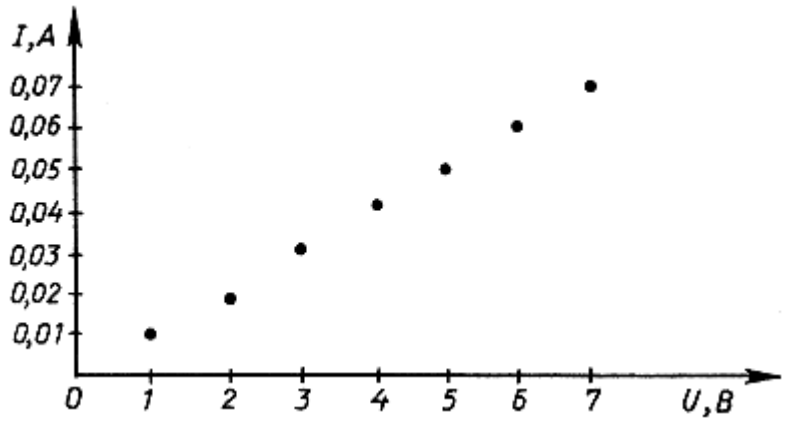

Рис. 34.

По закону Ома сила тока, проходящего через резистор, зависит от приложенного к нему напряжения:

$$I = \frac{1}{R} \cdot U.$$

Таким образом, графиком этой зависимости является прямая, проходящая через начало координат, у которой $k = \frac{1}{R}$. Но на одной прямой эти точки не лежат (рис. 34). Это связано с тем, что показания амперметра и вольтметра не совсем точные.

Постараемся провести прямую таким образом, чтобы точки графика были по возможности близки к этой прямой. По методу наименьших квадратов[1] угловой коэффициент искомой прямой вычисляется по формуле:

$$K = \frac{U_1 I_1 + U_2 I_2 + \ldots + U_7 I_7}{U_1^2 + U_2^2 + \ldots + U_7^2},$$

где U_1, I_1 —значения для первой экспериментальной точки, U_2, I_2 — для второй точки и т. д. Если все величины удалось измерить точно (все точки легли на прямую $I = \frac{1}{R} U$), то в этом случае

[1] Метод наименьших квадратов используется при построении математической модели, описывающей результаты эксперимента в физике, биологии, социологии и других областях. Его обоснование не входит в школьную программу и проводится в курсе высшей математики.

получаем $K = \frac{\frac{1}{R}(U_1^2 + ... + U_7^2}{(U_1^2 + ... + U_7^2} = \frac{1}{R}$, т.е. формула дает ожидаемый результат.

При таком количестве экспериментальных точек значение *K* можно вычислить вручную или с помощью калькулятора. Если же экспериментальных точек много или вычисления должны производиться многократно для разных наборов экспериментальных точек, разумно составить программу для ЭВМ. Алгоритм, реализующий данный метод вычисления, имеет вид:

алг метод наименьших квадратов (**нат** *N,* **вещ таб** *U*[*l:N*], *I*[1*:N*], **вещ** *K*)
арг *N, U, I*
рез *K*
нач
цел *i*
вещ *числ, знам*
i := 0
числ:= 0
знам:= 0
пока $i \neq N$
нц
$i := i + 1$
числ := *числ* + *U*[*i*] × *I*[*i*]
знам := *знам* + *U*[*i*] ×*U*[*i*]
кц
K := *числ/знам*
кон

Пояснение. Между выполнениями серии команд, входящих в цикл, в переменных *числ* и *знам* хранится сумма первых / членов в числителе и знаменателе.

Для нашего набора экспериментальных точек в результате вычисления по этой программе или после вычисления на калькуляторе получается следующий результат:

$$k = \frac{1}{R} = 0{,}01019 \text{ или R = 98 Ом.}$$

(При проверке этих результатов на калькуляторе удобно выразить значение силы тока не в амперах, а в миллиамперах. При этом в формулу подсчета углового коэффициента искомой прямой нужно ввести множитель 0,001.)

Построим прямую (рис. 35), соответствующую полученному коэффициенту.

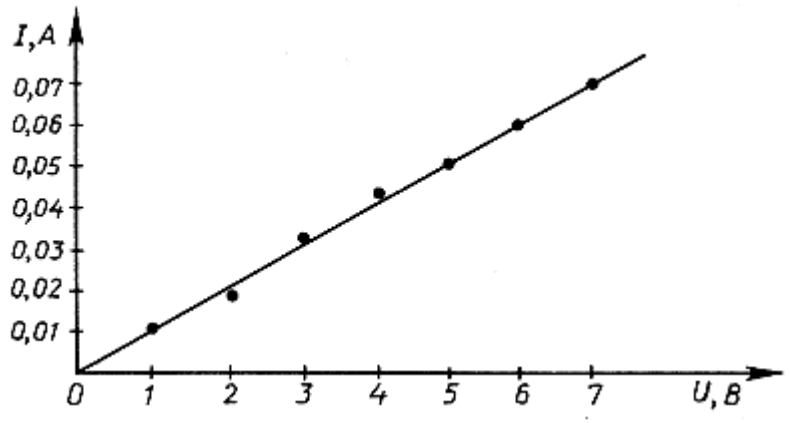

Рис. 35.

Видно, что отклонение экспериментальных точек от найденной прямой действительно мало.

Пример 2. Шарик массой m = 0,1 кг подвесили к пружине от школьного динамометра, оттянули вниз от положения равновесия на 1 см и отпустили. Он начинает двигаться вверх и через некоторое время проходит через положение равновесия. Найти это время, если жесткость пружины, определенная опытным путем, оказалась равной 40 Н/м.

Начнем с построения математической модели. Введем систему координат. Начало координат O поместим в положение равновесия, ось x направим вертикально вниз. Сила, с которой отклоненная на расстояние x от положения равновесия пружина действует на тело, равна $F = -kx$, где k — жесткость пружины, знак «минус» поставлен, так как сила, с которой пружина действует на тело, направлена противоположно отклонению тела. Ускорение a тела равно $\frac{F}{m}$, где m — масса тела. Пусть отрезок времени Δt мал, тогда за время Δt ускорение и скорость изменятся мало. Если ускорение равно a, то за время Δt скорость изменяется на $\Delta v = a \cdot \Delta t$. Если скорость равна v, то за время Δt координата тела изменяется на $\Delta x = v \cdot \Delta t$. Будем вычислять положение, ускорение и скорость тела, разбив ось времени на участки по Δt = 0,01 с.

Начнем с первого интервала: от t = 0 до t = 0,01. Координата в момент 0 равна 1 см = 10^{-2} м по условию задачи. Ускорение находится по формуле $a = -\frac{k}{m}x$ и равно −4 м/с^2. В начальный момент скорость по условию равнялась 0. В середине интервала скорость изменилась на $a \cdot \frac{\Delta t}{2}$ (прошло $\frac{\Delta t}{2}$ с) и стала равной $0 - 4 \cdot \frac{10^{-2}}{2} = — 2 \cdot 10^{-2}$. За первый интервал координата изменилась на $v \cdot \Delta t$ и к началу второго интервала стала равной $0{,}01 - 2 \cdot 10^{-2} \cdot 0{,}01 = 0{,}98 \cdot 10^{-2}$. В этот момент ускорение становится равным $a = \frac{-k}{m}\ x = -3{,}92$. Найдем скорость тела в середине второго интервала. От середины первого интервала до середины второго прошло Δt = 0,01, скорость изменилась на $a \cdot \Delta t$ и стала равной $-2 \cdot 10^{-2} - 3{,}92 \cdot 10^{-2} = -5{,}92 \cdot 10^{-2}$. Зная ее, мы можем определить координату тела к концу второго интервала, его ускорение, скорость в середине третьего интервала и т. п. Результаты этих вычислений сведены в таблицу 16.

Таблица 16

Номер интервала	Время начала интервала, с	Координата в момент начала интервала, 10^{-2} м	Ускорение в момент начала интервала, м/с^2	Скорость в середине интервала, 10^{-2} м/с
1	0,00	1,00	-4,00	-2,00
2	0,01	0,98	-3,92	-5,92
3	0,02	0,92	-3,68	-9,60
4	0,03	0,82	-3,20	-12,88
5	0,04	0,69	-2,76	-15,64
6	0,05	0,53	-2,12	-17,76
7	0,06	0,35	-1,40	-19,16
8	0,07	0,16	-0,64	-19,80
9	0,08	-0,04	0,16	-19,64
10	0,09	-0,24	0,96	-18,68

Анализируя результаты вычислений, приведенных в таблице, мы видим: модуль ускорения тела по мере его приближения к положению равновесия уменьшается, а модуль скорости увеличивается вплоть до точки, которая соответствует 8-му интервалу. В этот момент координата меняет знак. Это значит, что тело проходит положение равновесия в промежутке между 0,07 и 0,08 секунды от начала движения и по инерции движется дальше, при этом его движение замедляется (модуль скорости уменьшается).

Запишем теперь все сказанное выше в виде алгоритма, вычисляющего координату x точки, в которой находится тело массой m по истечению N промежутков времени, каждый из которых равен At (k — жесткость пружины, v_0 — начальная скорость).

алг колебание (**вещ** $m, k, \Delta t, N, v_0, x_0, x$)
арг $m, k, \Delta t, N, v_0, x_0$
рез x
нач **цел** i; **вещ** a, v
$i := 1; x := x_0; a := -k \cdot x/m; v := v_0 + a \cdot \Delta t/2$
пока $i \neq N + 1$
нц
$i := i + 1; x := x + v \cdot \Delta t; a := -k \cdot x/m; v := v + a \cdot \Delta t$
кц
кон

Пояснение. Между выполнением повторяющейся серии x равно координате тела в начале i-го интервала времени, a — ускорение в тот же момент, v — скорость в середине интервала.

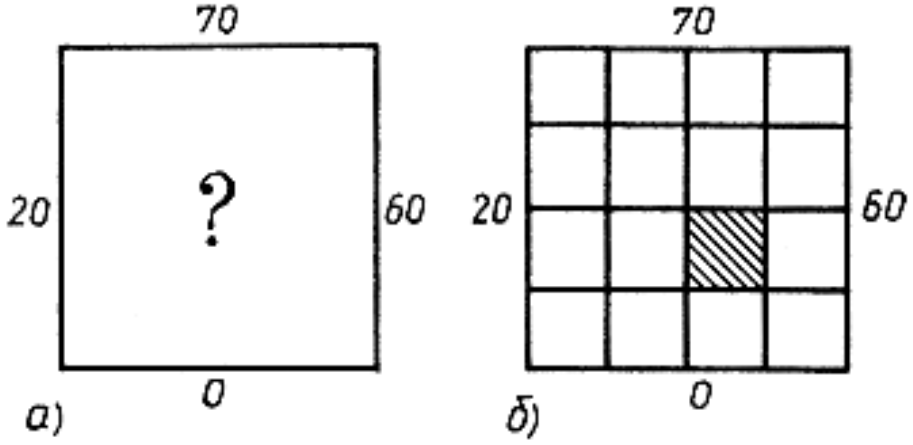

Рис. 36.

Пример 3. На краях однородной квадратной пластины (рис. 36, *а)* поддерживается температура, указанная на рисунке. Какова будет температура во внутренних точках пластины?

Эта температура будет меняться от точки к точке: у нижнего края она будет близка к 0° С, у верхнего — к 70° С, у левого — к 20° С, у правого — к 60° С. Внутри пластины она будет принимать какие-то промежуточные значения. Чтобы найти их, нам придется сделать допущение, что теплообмен с пластиной происходит по ее краям, а в середине ее не подогревают и не охлаждают.

Такие задачи могут решаться с помощью ЭВМ. Поясним метод их решения на примере этой задачи.

Прежде всего разобьем нашу пластину на много маленьких квадратиков. Будем считать, что внутри каждого квадратика температура постоянна. Это предположение будет справедливо лишь приближенно, но при достаточно малом размере квадратика ошибка будет мала (рис. 36, б).

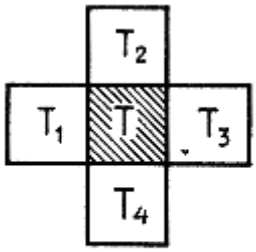

Рис. 37.

Затем мы составим систему уравнений, исходя из следующего принципа: температура каждого квадратика равна среднему арифметическому температуры четырех соседних квадратиков. Этот принцип можно обосновать следующим образом. Если квадратик температурой T граничит с квадратиком температурой T_1, то через границу идет поток тепла, пропорциональный разности температур с некоторым коэффициентом пропорциональности c. Таким образом, через границу квадратика, изображенного на рисунке 37, проходит суммарный поток тепла:

$$c(T_1 - T) + c(T_2 - T) + c(T_3 + T) + c(T_4 - T).$$

Поскольку мы считаем квадратик теплоизолированным, то эта сумма должна равняться 0. Сокращая на с и перенося T в другую часть, получаем:

$$T_1 + T_2 + T_3 + T_4 = 4T.$$

Мы рассмотрели квадратик, не примыкающий к краю. Если квадратик примыкает к краю, то у него не четыре соседа, а только три (или два, если он в углу). В этом случае роль соседа со стороны края выполняет окружающая среда, температура которой нам известна. Разобьем нашу пластину на 9 квадратиков. Через $t[1,1]$, $t[1,2]$, ..., $t[3,3]$ обозначим температуру соответствующих квадратиков.

70

	t[1,1]	t[1,2]	t[1,3]	
20	t[2,1]	t[2,2]	t[2,3]	60
	t[3,1]	t[3,2]	t[3,3]	

0

Мы получаем такие уравнения:

$$t[1,1] = \frac{20 + 70 + t[1,2] + t[2,1]}{4};$$

$$t[1,2] = \frac{t[1,1] + 70 + t[1,3] + t[2,2]}{4};$$

$$t[1,3] = \frac{70 + 60 + t[1,2] + t[2,3]}{4};$$

$$t[2,1] = \frac{20 + t[1,1] + t[2,2] + t[3,1]}{4};$$

$$t[2,2] = \frac{t[2,1] + t[1,2] + t[2,3] + t[3,2]}{4};$$

$$t[2,3] = \frac{60 + t[2,2] + t[1,3] + t[3,3]}{4};$$

$$t[3,1] = \frac{20 + 0 + t[2,1] + t[3,2]}{4};$$

$$t[3,2] = \frac{0 + t[3,1] + t[2,2] + t[3,3]}{4};$$

$$t[3,3] = \frac{0 + 60 + t[3,2] + t[2,3}{4}.$$

Эта система имеет 9 уравнений и 9 неизвестных. Если бы мы разбили каждую сторону квадрата не на три, а на N частей, то неизвестных и уравнений было бы N^2. На практике значения N — это сотни и даже тысячи. Для решения таких трудоемких задач применяются супер-ЭВМ, делающие миллиарды операций в секунду. Мы, однако, проиллюстрируем их решение на примере всего с девятью квадратиками. В этом примере вычисления можно провести с помощью калькулятора или даже вручную.

Один из методов решения нашей системы уравнений таков.

В качестве первого приближения мы считаем, что все температуры t равны 35. Разумеется, при этом уравнения не будут выполнены: их левые части будут равны 35, а правые — нет. Запишем те и другие в таблицу на соответствующие места.

Левые

	70			
	30	30	30	
20	30	30	30	60
	30	30	30	
	0			

Правые

	70			
	40	43,75	50	
20	31,25	35	41,25	60
	22,5	26,25	32,5	
	0			

Правые части мы вычислили в соответствии с нашими уравнениями; например,40 есть $\frac{70 + 20 + 35 + 35}{4}$; 43,75 есть $\frac{35 + 70 + 35 + 35}{4}$ и т. д. Числа в правой таблице и будут следующими приближениями к искомой температуре.

Они снова не будут решениями нашего уравнения. Если мы подставим их в уравнение, то, округляя до сотых, получим такие левые и правые части:

Левые

	70			
	40	43,75	50	
20	31,25	35	41,25	60
	22,5	26,25	32,5	
	0			

Правые

	70			
	41,25	48,75	53,75	
20	29,38	35,63	44,38	60
	19,38	22,5	31,88	
	0			

Все расчеты происходят по тому же алгоритму: число в клетке правой таблицы равно среднему арифметическому чисел в левой таблице, расположенных на соседних местах. При этом для клеток с краю используются известные нам температуры (20, 70, 60 и 0):

$41{,}25 = \dfrac{20 + 70 + 43{,}75 + 31{,}25}{4}$; $48{,}75 = \dfrac{40 + 70 + 50 + 35}{4}$ и т.д.

Числа в правой таблице и составят следующее приближение. Будем повторять это много раз: переписывая на каждом шаге числа из правой таблицы в левую и затем находя новые значения чисел в правой таблице описанным способом. Мы приведем еще несколько шагов. При этом не будем дважды повторять одну и ту же таблицу, будем выписывать только правые части.

	70			
	42,03	50,16	55,78	
20	29,07	36,25	45,32	60
	17,97	21,72	31,72	
	0			

	70			
	42,31	50,02	56,37	
20	29,06	36,57	45,94	60
	17,70	21,49	31,76	
	0			

	70			
	42,52	51,31	56,74	
20	29,15	36,88	46,18	60
	17,64	21,51	31,86	
	0			

Видно, что с каждым разом значения чисел в каждой следующей таблице становятся все более близкими к значениям соответствующих чисел в предыдущей таблице. Это значит, что разница между левой и правой частями уравнений становится все меньше и меньше. Таким образом, мы находим решение наших уравнений все точнее и точнее.

Мы можем остановиться на нашем последнем приближении, поскольку все равно надеяться на точное решение исходной физической задачи не приходится: уж слишком мало мы взяли квадратиков. При большем числе квадратиков решать систему «вручную» трудно, поэтому приходится поручать исполнение алгоритма компьютеру.

Запишем на алгоритмическом языке заголовок использованного нами алгоритма. Его аргументами будут температуры t_1, t_2, t_3, t_4, заданные на четырех сторонах квадрата, а также число N отрезков, на которые мы разбивали стороны квадрата. Результатом этого алгоритма будет таблица размером $N \times N$, описывающая распределение температуры.

алг теплопроводность (**вещ** t_1, t_2, t_3, t_4, **нат** N, **вещ таб** *температура* [1:N, 1 :N])
арг t_1, t_2, t_3, t_4, N
рез *температура*

Мы не приводим текст этого алгоритма полностью, поскольку ход его выполнения был подробно описан. Задачи, аналогичные рассмотренным в данном параграфе, и методы их решения типичны для современных применений ЭВМ. Аналогичными методами решаются упоминавшиеся во введении задачи прогноза погоды и моделирования термоядерной реакции.

Упражнения

1. Выпишите формулу для нахождения углового коэффициента прямой по методу наименьших квадратов при условии, что ток измерен в миллиамперах, напряжение — в вольтах, а сопротивление — в омах (пример 1). Произведите подсчет по формуле.

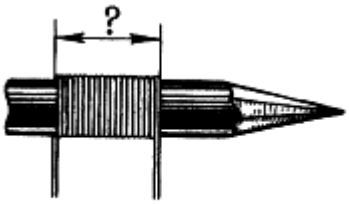

Рис. 38.

2. Измерьте толщину нити. Для этого возьмите карандаш и намотайте на него 10, 20, 30, ..., 100 витков нити (укладывая ее виток к витку). Измерьте линейкой длину участка карандаша, покрытого нитью (рис. 38). С помощью разобранного метода (пример 1) найдите толщину нити, используя результаты всех 10 измерений.

3. Найдите толщину листа бумаги, складывая его в несколько раз и измеряя толщину получившейся стопки.

4. В задаче о движении шарика на пружинке постройте график зависимости положения шарика x от времени на промежутке в 0,1 с.

5. В задаче о движении шарика на пружинке проведите расчет движения шарика, взяв в качестве Δt не 0,01 с, а 0,02 с. Как изменится результат?

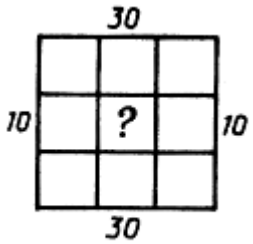

Рис. 39.

6. Найдите распределение температуры во внутренних точках квадратной однородной пластины, считая, что пластина разбита на 9 квадратиков, а распределение температуры на границе пластины дано на рисунке 39.

Выполните четыре шага вычислений.

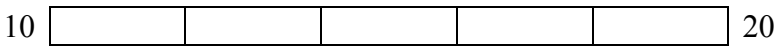

Рис. 40

7. Найдите распределение температуры во внутренних точках тонкого однородного стержня, считая, что стержень разбит на 5 отрезков и что на концах стержня температура равна 10 и 20° (рис. 40). Начните с температуры 15° на всех отрезках. Выполните 5 шагов вычислений. Так же как в случае пластины, исходите из предположения, что температура каждого отрезка равна среднему арифметическому температур соседних отрезков.

§ 9. АЛГОРИТМЫ РАБОТЫ С ГРАФИЧЕСКОЙ ИНФОРМАЦИЕЙ

На практике встречаются задачи, результатом решения которых являются чертежи, графики, диаграммы, рисунки и другая графическая информация.

Если задача решается с помощью компьютера, то графическая информация выводится либо на графический дисплей, на экране которого как в телевизоре эта информация высвечивается, либо на графопостроитель, который вычерчивает графическую информацию на листе бумаги. Выбрав координаты на экране дисплея или на листе бумаги, мы будем рассматривать их как часть плоскости и в дальнейшем говорить о рисовании на плоскости.

Для того чтобы вывести графическую информацию, необходимо указать, какие ее элементы (точки, отрезки, окружности и т. д.) надо вывести и в какой последовательности.

Поэтому, так же как и в случае других задач, решение графических задач требует составления алгоритмов. Эти алгоритмы состоят из специальных графических команд, каждая из которых указывает, какое элементарное графическое действие надо совершить (нарисовать точку, отрезок, окружность, многоугольник и т. д.).

Для рисования с помощью компьютера разработаны удобные наборы графических команд. Наш набор команд очень маленький и удобен для рисования простейших геометрических фигур. Он будет использовать прямоугольную систему координат на плоскости.

Для лучшего понимания правил выполнения графических команд удобно представлять себе исполнителя двигающимся по плоскости и рисующим на ней. Сначала исполнитель находится в точке плоскости с координатами (0,0) и смотрит вдоль оси *y*.

При выполнении графических команд может изменяться как точка, где находится исполнитель, так и направление, куда он смотрит.

1. Команды вычерчивания: вперед (*a*), назад (*a*)

Для того чтобы нарисовать отрезок, необходимо задать его начальную точку, направление и длину.

Исполнитель по команде вперед (*a*) вычерчивает отрезок длиной *a* с начальной точкой и с направлением таким же, как у исполнителя перед началом выполнения этой команды.. После выполнения команды исполнитель попадает в концевую точку нарисованного отрезка, а его направление остается неизменным.

Команда назад (*a*) отличается от команды вперед (*a*) только тем, что отрезок вычерчивается в направлении, противоположном направлению исполнителя. Направление исполнителя при этом не изменяется, а сам он попадает в конечную точку нарисованного отрезка.

Если выполнить команды

назад (1)
вперед (3)

то получим отрезок, изображенный на рисунке 41.

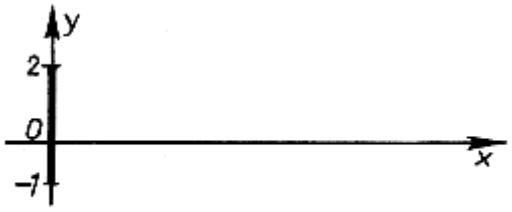

Рис. 41.

2. Команды поворота: направо (*b*), налево (*b*)

Исполнитель по команде направо (*b*) поворачивается на *b* градусов вправо.
Исполнитель по команде налево (*b*) поворачивается на *b* градусов влево.

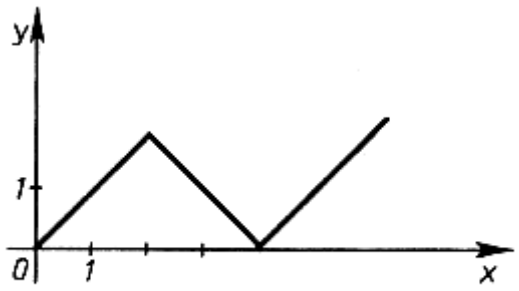

Рис. 42.

Пример 1 (рис. 42).

```
алг ломаная
нач
    направо (45)
    вперед (3)
    направо (90)
    вперед (3)
    налево (90)
    вперед (3)
кон
```

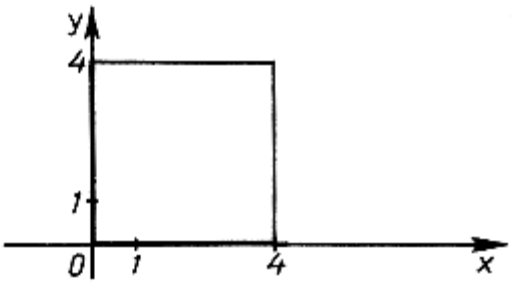

Рис. 43.

Пример 2 (рис. 43).

```
алг квадрат
нач
    вперед (4)
    направо (90)
    вперед (4)
    направо (90)
    вперед (4)
    направо (90)
    вперед (4)
кон
```

В примере 2 команды вперед (4) и направо (90) многократно повторяются. Запишем алгоритм вычерчивания квадрата со стороной длины *N*.

```
алг квадрат (вещ N)
    арг N
нач цел I
    I := 1
```

пока $I \le 4$
нц
вперед (N)
направо (90)
$I := I + 1$
кц
кон

3. Команды для вычерчивания изображения, состоящего из отдельных элементов: рисуй, не рисуй

В некоторых случаях возникает необходимость перемещаться по плоскости без вычерчивания. Для этого вводится команда не рисуй. .После этой команды исполнитель перестает рисовать отрезки, задаваемые командами вперед и назад, которые используются теперь только для передвижения. Для отмены действия команды не рисуй вводится команда рисуй. После команды рисуй исполнитель при выполнении команд вперед и назад начинает рисовать отрезки. Поясним сказанное на примере.

Пример 3. Написать слово МИР.

алг написание слова МИР
нач
рисуй
вперед (4); налево (30)
назад (2); направо (60)
вперед (2); налево (30)
назад (4)

Нарисована буква М (рис. 44).

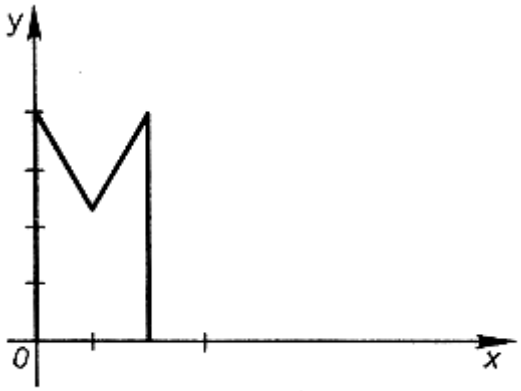

Рис. 44.

Теперь необходимо оставить промежуток между буквами М и И.

не рисуй
направо (90); вперед (1)
налево (90); рисуй

Продолжаем, рисуем букву И (рис. 45).

вперед (4); не рисуй
назад (4); рисуй
направо (30); вперед (4,8)
налево (30); назад (4)

Делаем промежуток между буквами И и Р и рисуем букву Р (рис. 46).

не рисуй

направо (90); вперед (1)
налево (90)
рисуй; вперед (4)
направо (90) вперед (2)
направо (90) вперед (2)
направо (90) вперед (2)
кон

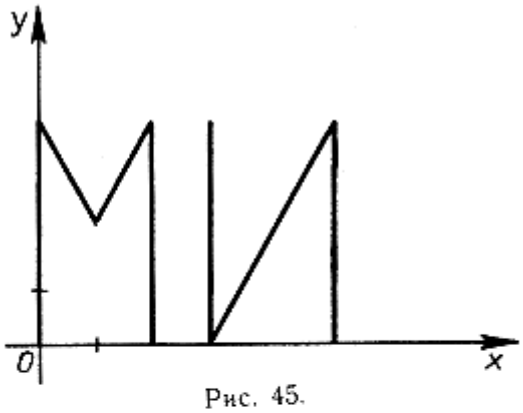

Рис. 45.

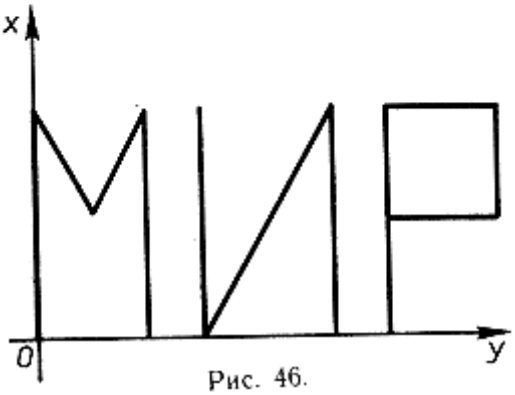

Рис. 46.

Используя в качестве исполнителя графических команд компьютер, можно оформлять результаты решения задач по математике, физике, химии и т. д. в виде графиков, таблиц, диаграмм, гистограмм, моделировать на экране дисплея явления природы, играть и создавать свои компьютерные игры.

Упражнения

1. Составьте алгоритмы рисования следующих изображений:
а) слова МИР, используя только команды вперед, вправо, рисуй, не рисуй;
б) своих инициалов;
в) проекции куба.
2. Исполните алгоритм при $N = 4$, $N = 20$.

алг многоугольник (**нат** *N*)
арг *N*
нач цел *I*
I := 1
пока $I \le N$
нц
вперед *(I*/*N*)
направо (360/*N*)
I := *I* + 1

кц

кон

Что будет происходить с изображением при увеличении *N?* Чему равен периметр нарисованного многоугольника?

2. Составьте и запишите алгоритмы рисования фигур, изображенных на рисунках 47—51.

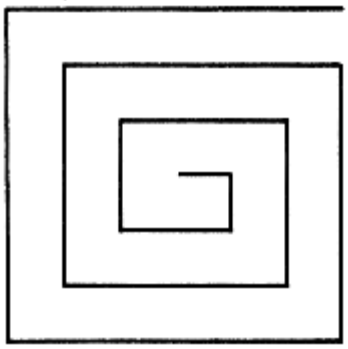

Рис. 47.

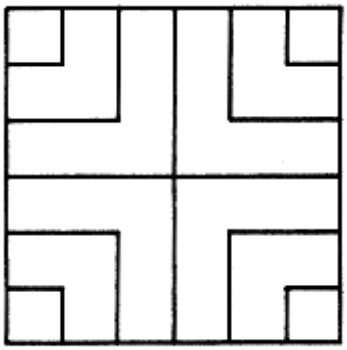

Рис. 48.

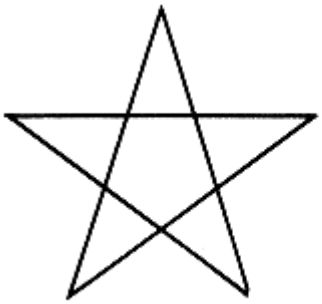

Рис. 49.

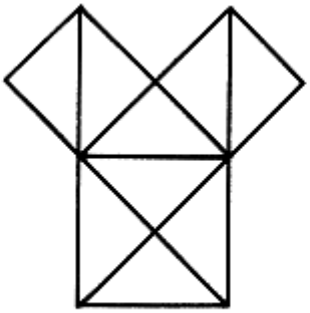

Рис. 50.

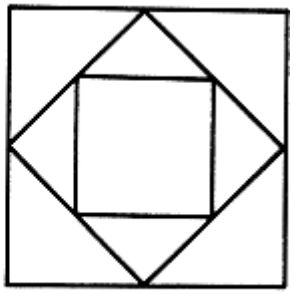

Рис. 51.

ПРИЛОЖЕНИЕ

1. РАБОТА С КАЛЬКУЛЯТОРОМ

Всякий алгоритм строится в расчете на какого-либо конкретного исполнителя. Однако исполнение каждого алгоритма включает в себя ряд действий, обязательных для любого исполнителя. Эти действия состоят в задании значений аргументов, выполнении очередной команды, осуществлении перехода от одной команды к другой, получении результата исполнения алгоритма.

Рассмотрим таких исполнителей алгоритма, как, например, человек с бумагой и карандашом и человек с калькулятором. Каждый из этих исполнителей обладает своими особенностями. Человек, вооруженный бумагой и карандашом, может в принципе исполнить вычислительный алгоритм, однако при этом все команды алгоритма выполняются вручную.

Калькулятор позволяет в ряде случаев выполнить отдельные вычислительные команды алгоритма. В этом смысле калькулятор является вычислительным средством исполнителя-человека.

1. Выполнение вычислительных команд алгоритма

Одним из распространенных типов инженерного калькулятора, или, как его иначе называют, калькулятора для научно-технических расчетов, является калькулятор «Электроника Б3-36». Он представляет собой вычислительное устройство небольших размеров с набором из 25 клавиш и световым экраном (индикатором). Общий вид этого калькулятора изображен на рисунке 52.

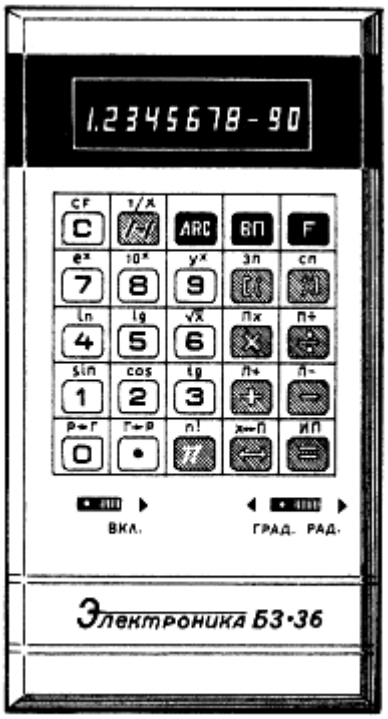

Рис. 52.

Прежде чем переходить к исполнению алгоритма, нужно научиться выполнять при помощи калькулятора часто встречающиеся вычисления. С этой целью ниже приводятся математические задачи, которые помогают освоить технику вычислений.

А. Задачи на непосредственное вычисление

1. Какое из чисел больше:

а) $\sqrt{10}+1$ или $\sqrt{17}$;

б) $\mathrm{tg}70°$ или $10\sqrt{19}$;

в) $60!$ или 30^{60};

г) π^{26} или 200π;

д) $100\sin 40°$ или 8^3;

е) $10^{\sin 29°}$ или $\sqrt{10}$;

ж) $\sin(\cos 1)$ или $\cos(\sin 1)$;

з) 41^{53} или 53^{41};

и) $2^{\sin 3}$ или $3^{\sin 2}$;

к) $\sqrt{3}+2\sqrt{2}$ или $\sqrt{\sqrt{5}+\sqrt{6}}$;

л) 2^{100} или 100^{20};

м) $\mathrm{tg}\left(\frac{\pi}{2}-\frac{1}{100000}\right)$ или 2^{16}?

2. К какому значению стремятся следующие выражения при увеличении n:

а) $\underbrace{\sqrt{4\cdot\sqrt{4\cdot\sqrt{4\cdot\ldots\cdot\sqrt{4\cdot\sqrt{5}}}}}}_{n\text{ раз}}$;

б) $\frac{100n}{n^2+1}$;

в) $\underbrace{\sqrt{2+\sqrt{2+\sqrt{2+\ldots+\sqrt{2+\sqrt{2}}}}}}_{n\text{ раз}}$;

г) $\left(1+\frac{1}{2}+\frac{1}{4}+\ldots+\frac{1}{2^n}\right)$;

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

е) $\left(1+\frac{1}{1!}+\frac{1}{2!}+\ldots+\frac{1}{n!}\right)$;

ж) $\left(1+\frac{1}{2^2}+\frac{1}{3^2}+\ldots+\frac{1}{n^2}\right)$;

з) $\left(1+\frac{1}{2}+\frac{1}{3}+\ldots+\frac{1}{n}-1\mathrm{n}\,n\right)$;

и) $\left(1+\frac{1}{2^8}+\frac{1}{3^8}+\ldots+\frac{1}{n^8}\right)$?

Б. Задачи с буквенными обозначениями

С помощью калькулятора нельзя доказывать теоремы. Например, нельзя доказать, что $\sin^2 x + \cos^2 x = 1$. Для этого пришлось бы проверить тождество для всех значений x, которых существует бесконечно много. Однако можно доказать неправильность какого-нибудь тождества. Для этого достаточно найти всего лишь один пример, который его опровергает. Например, тождество $(x + 1)^2 = x^2 + x + 1$ не выполняется, поскольку, подставив значение $x = 1$, получаем $4 = 3$.

3. Какие из следующих тождеств можно опровергнуть:

а) $\sin^2 x + \cos^2 x + 2\sin x \cos x = \frac{1}{2}$;

б) $\mathrm{tg}\left(x+\frac{\pi}{2}\right) = 2\mathrm{ctg}\,x$;

в) $\mathrm{tg}\left(x+\frac{5+\pi}{2}\right) = 4\mathrm{tg}\,x$?

В. Задачи на построение графиков

4. Составьте таблицу значений функции $f(x)$ на отрезке $[a, b]$ с шагом h и в соответствии с таблицей постройте график функции $f(x)$ на отрезке $[a, b]$:

а) $f(x) = 2x^3 - 7x^2 + 12, [a, b] = [-5,5], \quad h = 0{,}5;$

б) $f(x) = 4\sqrt{x} - \sin 3x$, $[a, b] = [0, 2]$, $h = 0{,}2$;

в) $f(x) = x^2 - 4\sqrt{x} + 2$, $[a, b] = [0, 4]$, $h = 0{,}2$;

г) $f(x) = x^3 - 3x^2 - 2x$, $[a, b] = [0, 4\ ]$, $h = 0{,}5$.

Г. Аналитические задачи

Эти задачи, кроме непосредственного вычисления на калькуляторе, требуют анализа результатов. Каждая из задач представляет собой небольшое исследование. Выводы, которые можно из него получить, могут оказаться полезными в дальнейшем.

5. Над числами $a_1 = \frac{8}{7}$, $a_2 = 1{,}2$, $a_3 = \sqrt{5} - 1$, $a_4 = \frac{\sqrt{10}}{3}$, $a_5 = 1$ произведите следующие действия:

а) расположите в порядке возрастания;

б) вычислите среднее арифметическое этих чисел ($a_{\text{ср}}$) с точностью до 0,01;

в) для каждого из данных чисел найдите его отклонение от среднего арифметического $\Delta_i = |a_i - a_{\text{ср}}|$;

г) вычислите $\sum_{i=1}^{5} \Delta_i$;

д) вычислите среднее геометрическое $\sqrt[5]{\Pi a_i}$ и сравните его с $\sum_{i=1}^{5} \Delta_i$ и с $a_{\text{ср}}$.

6. Последовательность задана формулой ее n-го члена: $a_n = \frac{1}{n^2 + 4}$.

а) Вычислите первые восемь членов последовательности.

б) Найдите a_{40} и a_{200}.

в) Чему равно a_{967} с точностью до 10^{-3}?

г) Укажите какой-нибудь член последовательности, отличающийся от 0 менее чем на 10^{-5}.

7. Последовательность задана формулой ее n-го члена:

$$x_n = \frac{3_n + 4}{n + 1}.$$

а) Найдите первые восемь членов последовательности.

б) Вычислите x_{100} и x_{1000}.

в) Для первых восьми членов последовательности вычислите $|x_n - 3|$.

8. Для последовательности (u_n) известно:

$$u_1 = 1,\ u_{n+1} = \frac{u_n + \frac{3}{u_n}}{2} (n \geq 2).$$

а) Вычислите первые восемь членов этой последовательности.

б) Найдите $u_{n+1} - u_n$ для $n = 1, 2, ..., 8$.

в) Вычислите $|u_n - 1{,}7|$ для $n = 1, 2, ..., 8$.

г) Верно ли, что предел последовательности равен 1,7 при увеличивающихся значениях n?

9. Последовательность задана формулой ее n-го члена:

$$b_n = q^n, \text{ где } q = 2;\ 1;\ 2;\ 0{,}4;\ -0{,}7;\ \frac{\sqrt{2}}{3}.$$

а) Вычислите первые шесть, двадцатый и сотый члены последовательности для каждого значения q.

б) Найдите значения q, для которых $q^n \to 0$.

в) Для каждого из указанных значений q подберите номер, начиная с которого $q^n \approx 0$, с точностью до 10^{-3}.

10. Известны формулы n-х членов последовательностей (a_n) и (b_n): $a_n = \frac{1}{n^3}$, $b_n = \frac{1}{2^n}$.

а) Выбрав произвольно 10 значений n, вычислите a_n, b_n, $a_n + b_n$, $a_n - b_n$, $b_n - a_n$, $\frac{a_n}{b_n}$, $\frac{b_n}{a_n}$.

б) Верно ли, что $a_n \to 0$, $b_n \to 0$, $(a_n - b_n) \to 0$, $\frac{a_n}{b_n} \to 0$?

11. Для последовательности (F_n) известно, что $F_1 = F_2 = 1$, $F_n = F_{n-1} + F_{n-2} (n \geq 3)$ (последовательность Фибоначчи).

а) Выпишите 10 первых членов последовательности.

б) Проверьте для F_2, F_3, F_{10} формулу

$$F_n = \frac{(\frac{1+\sqrt{5}}{2})^n - (\frac{1-\sqrt{5}}{2})^n}{\sqrt{5}}.$$

в) Составьте новую последовательность $q_n = \frac{F_{n+1}}{F_n}$.

г) Стремится ли q_n к какому-нибудь числу при увеличении n?

2. Вычислительные задачи из курсов физики и химии

Инженерный калькулятор может быть использован при решении различных задач, которые встречаются в курсах физики и химии.

Рассмотрим, например, следующие задачи:

Задача 1. Чему равна масса воздуха, впускаемого в цилиндр дизельного двигателя объемом 1,58 л при температуре 67° С и давлении $8 \cdot 10^4$ Па ($R \approx 8{,}31$ Дж/(моль · K))?

Решение. Введем обозначение: $V = 1{,}58$ л, $t = 67°$ С, $T = 340$ К, $P = 8 \cdot 10^4$ Па, $M = 0{,}029$ кг/моль.

Запишем уравнение: $PV = \frac{m}{M}RT$, откуда $m = \frac{PVM}{RT}$:

$$m = \frac{8 \cdot 10^4 \cdot 0{,}029 \cdot 1{,}58 \cdot 10^{-3}}{8{,}31 \cdot 340}.$$

Для вычисления значения этого выражения запишем все числа в следующем виде:

$$m = \frac{0{,}8 \cdot 10^5 \cdot 0{,}29 \cdot 10^{-1} \cdot 0{,}158 \cdot 10^{-2}}{0{,}831 \cdot 10 \cdot 0{,}34 \cdot 10^3} =$$

$$= \frac{0{,}8 \cdot 0{,}29 \cdot 0{,}158}{0{,}831 \cdot 0{,}34} \cdot \frac{10^5 \cdot 10^{-1} \cdot 10^{-2}}{10 \cdot 10^3} = \frac{0{,}8 \cdot 0{,}29 \cdot 0{,}158}{0{,}831 \cdot 0{,}34} \cdot 10^{-2}.$$

Вычислим на калькуляторе выражение $\frac{0{,}8 \cdot 0{,}29 \cdot 0{,}158}{0{,}831 \cdot 0{,}34}$ по программе 0,8 [×] 0,29 [×] 0,158 [÷] 0,831 [÷] 0,34.

Получим 0,1297373.

Поскольку данное выражение состоит только из произведений и частного, то согласно правилу округления его надо округлить до такого числа значащих цифр, какое имеет наименее точное исходное данное.

В данном случае 0,8 имеет одну значащую цифру. Поэтому ответ округляется до одной значащей цифры: $m \approx 1 \cdot 10^{-3}$ кг.

Задача 2. Рассчитать массу железа, образующегося при восстановлении оксида железа (III) массой 475 кг, содержащего 7% примесей, оксидом углерода (II).

Решение. Составим уравнение реакции:

$$\underset{m' = 160кг}{Fe_2O_3} + 3CO = \underset{m'' = 56кг}{2Fe} + 3CO_2$$

$$(1моль) \qquad (2моль)$$

Найдем массу оксида железа (III) без примесей по программе 475 [П] [ЗП] [÷] [100] [×] 7 [П].

Получим 442 кг.

Определим массу полученного железа. Из 160 кг Fe_2O_3 получается $2 \cdot 56$ кг Fe**,** из 442 кг: $x = \frac{442 \cdot 112}{160}$.

Вычислим по программе: 442 [×] 112 [÷] 160 [=].

После округления получим ответ 309 кг.

Используя калькулятор, решите следующие задачи:

1. Определите среднюю квадратическую скорость движения молекул водорода при температуре 330К. Масса молекул водорода составляет $3{,}35 \cdot 10^{-27}$ кг.

2. Паровой молот массой 9,5 т падает с высоты 2,3 м на стальную болванку массой 230 кг. Сколько раз он должен упасть, чтобы температура болванки поднялась на 45°С? На нагревание болванки идет 50% количества теплоты, полученной при ударе.

3. Найдите отношение массы протона, равной $1{,}6726 \cdot 10^{-31}$ кг, к массе электрона, равной $9{,}1095 \cdot 10^{-31}$ кг.

4. Три резистора, сопротивления которых соответственно равны R_1 = 12 Ом, R_2 = 17 Ом, R_3 = 2,9 Ом, соединены параллельно. Определите общее сопротивление цепи.

5. Смешали раствор гидроксида калия массой 185 г с массовой долей (%) растворенного вещества 27% и раствор соляной кислоты массой 65,0 г с массовой долей (%) растворенного вещества 36,5%. Найдите массу образовавшегося хлорида калия.

6. Из азота массой 59 кг был синтезирован аммиак массой 47 кг. Каков выход аммиака в процентах от теоретически возможного?

7. Вычислите массовую долю (%) азота в кальциевой селитре.

8. К раствору, содержащему 17,6 хлорида меди (II), прилили раствор, содержащий 23,2 г гидроксида натрия. Вычислите массу образовавшегося гидроксида меди (II) (с точностью до 0,1 г).

9. Сколько литров воздуха необходимо для полного сгорания 27,0 л этана?

10. Запишите молекулярную формулу вещества по данным состава: С — 52%, Н — 13%, О — 35%, если относительная плотность паров этого газа по воздуху 1,59.

3. Использование калькуляторов для выполнения алгоритма

Пример **1.** Алгоритм Евклида.

Запишем алгоритм в виде, удобном для исполнителя-человека, имеющего инженерный калькулятор. В этой записи наряду с командами алгоритмического языка будут использованы команды калькулятора, которые будут обозначаться в виде квадрата с указанием символа команды внутри него. Команды будут разделяться точкой с запятой, как это принято в алгоритмическом языке:

нач *a*; [—]; *b;* **пока** $a \neq b$ **нц если** $a < b$ то [↔] **;** [=] **кц; кон**

Проследим на конкретном примере, как исполнитель с калькулятором будет выполнять такой алгоритм. Пусть надо найти НОД(30, 18). Запишем в таблицу (табл. 17) аргументы *a* и *b.* В эту же таблицу в процессе выполнения алгоритма будем записывать измененные значения *a* и *b.* В соответствии с алгоритмом исполнитель вводит в калькулятор значение *a*' = 30, нажимает клавишу [—] и вводит значение *b* = 18. После этого в РИ будет находиться число 18, а в РР — число 30. Так как a ;> *b,* то команда [↔] пропускается и исполнитель нажимает клавишу [=]. На индикаторе и в РИ появляется новое значение переменной *a* = 12. Исполнитель заносит это значение в таблицу (шаг № 2).

Так как $a \neq b$, то необходимо вернуться к началу цикла, что в программе отмечено стрелкой. Условие $a < b$ выполнено, поэтому исполнитель выполняет команду [↔], в ре-

зультате чего происходит обмен чисел 12 и 18, записанных в регистрах РИ и РР. Этот обмен отражается в таблице (шаг № 3).

Теперь после выполнения команды [=] на индикаторе и в регистре РИ появляется новое значение $a = 6$, которое также записывается в таблицу (шаг № 4).

Заметим, что калькулятор работает в режиме констант, «помня» операцию, которую он должен выполнять (вычитание), поэтому после выполнения команды [=] число в РР (значение b) не изменяется.

Процесс будет повторяться до тех пор, пока значения a и b не станут равными (шаг № 6).

Таблица 17

Номер шага	1	2	3	4	5	6
a	30	12	18	6	12	6
b	18	18	12	12	6	6

Теперь условие окончания цикла ($a = b$) выполнено, поэтому повторение действий, указанных в цикле, прекращается и исполнитель записывает в качестве результата значение 6.

Итак, НОД (30, 18) =6. Задача решена.

Конечно, для чисел 30 и 18 наибольший общий делитель можно легко найти и без калькулятора. Такие аргументы были выбраны лишь для того, чтобы показать, как используется калькулятор в процессе выполнения алгоритма. Для других аргументов, например 143 524 и 237 176, выполнить алгоритм Евклида без калькулятора уже не так просто.

П р и м е р 2. Вычислить значения многочлена (по схеме Горнера).

Алгоритм для исполнителя, использующего калькулятор, будет иметь следующий вид:

нач a; [×] x; i:= 1; **пока** $i \le n$ **нц** [=]; [F] [ЗП]; a_i; [F] [П+] ; [F] [ИП]; i: = i+ 1 **кц кон**

При выполнении этого алгоритма калькулятор также работает в режиме констант. Значение x находится в РР и остается неизменным в процессе выполнения алгоритма. Значение y накапливается в регистре памяти РП. Результат получается в РИ и высвечивается на индикаторе.

Рассмотрим действия исполнителя при вычислении значения многочлена $2x^3 + 7x^2 + 4x + 1$ при $x = 2$.

В этом случае $n = 3$ и цикл в программе должен быть выполнен три раза при $i = 1, 2, 3$.

Запишем значения коэффициентов многочлена в таблицу 18. Во второй строке этой таблицы будем помещать значения y после каждого выполнения цикла.

Исполнение алгоритма начинается с того, что исполнитель набирает на клавиатуре калькулятора значение $a_0 = 2$. Это значение является начальным значением y, поэтому оно заносится в таблицу в графу под значением a_0. Затем исполнитель нажимает клавишу [×] и набирает значение $x = 2$.

При первом выполнении цикла ($i = 1$) после нажатия клавиши [=] в РИ получается $a_0 \cdot x = 4$, а значение $x = 2$ переписывается в РР, где и остается неизменным до окончания работы программы. После выполнения команды [F] [П+]: значение $a_0 \cdot x = 4$ запоминается в РП. Далее исполнитель набирает на клавиатуре значение $a_1 = 7$ и выполняет команду [F] [П+]: значение $a_1 = 7$ прибавляется к значению, хранящемуся в РП. После выполнения команды [F] [ИП] полученное после первого выполнения цикла значение $a_0 \cdot x + a_1 = 11$ переписывается в РИ. Это значение y записывается в таблицу под значением a_1 (шаг № 1).

При втором выполнении цикла после выполнения команды [=] получается значение $(a_0x + a_1) \cdot x = 22$, которое снова записывается в РП, «стирая» находившееся там ранее значение. После вторичного выполнения всех команд цикла получается новое значение $y =$

$(a_0x + a_1) \cdot x + a_2 = 26$ (шаг № 2). Наконец, после третьего выполнения команд цикла в РИ получается результат $y = ((a_0x + a_1) \cdot x + a_2) \cdot x + a_3 = 53$, который исполнитель записывает в последнюю графу таблицы (шаг № 3).

Таблица 18

Номер шага		0	1	2	3
$x = 2$	a_i	2	7	4	1
	y	2	11	26	53

В рассмотренном примере вычисляли значение многочлена с целочисленными коэффициентами при $x = 2$ только для того, чтобы проще объяснить существо дела. В действительности калькулятор оказывается полезным, когда приходится вычислять значение многочлена с вещественными коэффициентами, например такого:
$62{,}135x^4 + 12{,}721x^3 + 5{,}965x^2 + 9{,}317x + 123{,}156$ при $\boldsymbol{x} = 7{,}121$.

II. БИБЛИОТЕКА АЛГОРИТМОВ

А 1. Алгоритм вычисления модуля (МОД) действительного числа

```
алг  МОД (вещ x, вещ y)
     арг x
     рез y
нач
     если x ≥ 0
          то y := x
          иначе y := −x
     все
кон
```

А 2. Алгоритм поиска большего из двух чисел *a* и *b* (БИД)

```
алг  БИД (вещ a, b, вещ y)
     арг a, b
     рез y
нач
     если a ≥ b
          то y := a
          иначе y := b
     все
кон
```

А3. Алгоритм решения линейного уравнения *ax— b* (ЛУР), где *a* и *b* — произвольные вещественные числа

```
алг  ЛУР (вещ a, b, вещ x, лит y)
     арг a, b
     рез x, y
нач
     если a ≠ 0
          то  y := „есть решение"
              x := b/a
          иначе
                    если   b = 0
                           то y := „x — любое число"
                           иначе y := „решений нет"
                    все
     все
кон
```

А 4. Алгоритм решения квадратного уравнения $ax^2 + bx + c = 0$ (КВУР), где a, b, c — произвольные вещественные числа ($a \neq 0$)

алг КВУР (**вещ** *a,b,c,* **вещ** x_1, x_2, **лит** *y*)
арг *a, b, c*
рез x_1, x_2, *y*
нач **вещ** *D*
$D := b^2 - 4 \cdot a \cdot c$
если $D < 0$
то *y* :== „нет решения"
иначе *y* := „есть решения"

$$x_1 := \frac{-b + \sqrt{D}}{2a}$$

$$x_2 := \frac{-b - \sqrt{D}}{2a}$$

все
кон

А5. Алгоритм решения неравенства $ax>b$ (НЕР), где a и b — произвольные вещественные числа

алг НЕР (**вещ** *a, b,* с, **лит** *y*)
арг *a, b*
рез *c*, *y*
нач
если $a \neq 0$
то $c := b/a$
если $a > 0$
то *y* := „$x > c$"
иначе *y* := „$x < c$"
все
иначе
есл $b <. 0$
то *y* := „*x* — любое число"
иначе *y* := „решений нет"
все
все
кон

А6. Алгоритм нахождения наибольшего общего делителя (НОД)

алг НОД (**нат** *M, N,* **нат** *НОД*)
арг *M, N*
рез *НОД*
нач **нат** *x, y*
x := M; y := N
пока $x \neq y$
нц
если $x > y$
то *x* := *x* — *y*
иначе *y* := *y* — *x*
все
кц
НОД := x
кон

А 7. Алгоритм нахождения наименьшего элемента в линейной таблице чисел

```
алг  МИНЭЛЕМЕНТ (цел K, N, вещ таб A [K:N] цел l)
     арг A, K, N
     рез l
нач цел i, вещ МИН
     МИН := A [K]
     l := K
     i := K +1
     пока i ≤ N
     нц
          если МИН > A [i]
                то
                МИН := A [i]
                l := i
          все
          i := i + 1
     кц
кон
```

А 8. Алгоритм упорядочения по возрастанию элементов линейной таблицы чисел

```
алг  УПОРЯДОЧЕНИЕ (цел n, M, вещ таб C [n:M])
     арг C, n, M
     рез C
нач цел i, l, вещ R
     i := n
     пока i < M
     нц
          МИНЭЛЕМЕНТ (i, M, C, l)
          R := C[i]
          C[i] := C[l]
          C[l] := R
          i := i + 1
     кц
кон
```

А 9. Алгоритм вычисления значения многочлена $P_n(x) = a_0x^n + a_1x^{n-1} + ... + a_{n-1}x + a_n$ по схеме Горнера

```
алг  СХЕМА ГОРНЕРА (цел n, вещ x, вещ таб a[0:n],
     вещ y)
     арг n, a, x
     рез y
нач цел i
     i := 0
     y := a[0]
     пока i ≠ n
     нц
       i := i + 1
       y := y · x + a[i]
     кц
кон
```

А 10. Алгоритм нахождения площади криволинейной трапеции, ограниченной прямыми $x = a$, $x = b$, $y = 0$ и кривой $y = f(x)$ при условии, что для всех x из отрезка $[a; b]$ значения $f(x) \geq 0$

```
алг  ПЛОЩАДЬ (вещ a, b, S, нат n)
     арг a, b, n
```

рез S
нач **вещ** *h, x,* **нат** *i*
$h := \frac{b-a}{n}$
$S := 0$
$x := a$
$i := 1$
пока $i \le n$
нц
$S := S + h \cdot f(x)$
$x := x + h$
$i := i + 1$
кц
кон

All. Алгоритм уточнения корня уравнения $f(x) = 0$ на отрезке $[a; b]$ с заданной точностью ε при условии, что функция $f(x)$ на отрезке $[a; b]$ монотонна и меняет знак

алг УТОЧНЕНИЕ КОРНЯ (**вещ** *a, b,* ε, **вещ** *x*)
арг *a, b,* ε
рез *x*
нач **вещ** *c*
пока $b - a > 2 \cdot \varepsilon$
нц
$c := \frac{a+b}{2}$
если $f(a) \cdot f(c) \le 0$
то $b := c$
иначе $a := c$
все
кц
$x := \frac{a+b}{2}$
кон

СОДЕРЖАНИЕ

Андрей Петрович Ершов
Вадим Макариевич Монахов
Сергей Александрович Бешенков
Ян Эдуардович Гольц
Александр Андреевич Кузнецов
Эдуард Иванович Кузнецов
Михаил Павлович Лапчик
Дмитрий Олегович Смекалин

Сдано в набор 17.04.85. Подписано к печати 25.04.85. Формат 60X90'/16. Бумага офсетная № 2. Гарнит. литерат. Печать офсетная. Усл. печ. л. 6. Усл. ко.-отт. 6,25. Уч.-изд. л. ,4.91. Тираж 3 300 000 (2 000 001—2 600 000) экз. Заказ № 1014. Цена 15 кОП.

ОСНОВЫ ИНФОРМАТИКИ И ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ
Пробное учебное пособие
для средних учебных заведений.
В двух частях
Часть первая

Ордена Трудового Красного Знамени издательство «Просвещение» Государственного комитета РСФСР по делам издательств, полиграфии и книжной торговли. 129846, Москва, 3-й проезд Марьиной рощи, 41.
Зав. редакцией *Р. А. Хабиб* Редакторы *Т. А. Бурмистрова,*
И. И. Никитина
Мл. редакторы *Л. Е. Козырева,*
Е. А. Сафронова
.Художник *Б. Л. Николаев*
Художественный редактор *Е. Н. Карасик*
Технический редактор *В. Ф. Коскина*
Корректор *Н. И. Новикова*
Отпечатано с диапозитивов ордена Трудового Красного Знамени фабрики «Детская книга» № 1 Росглавполиграфпрома Государственного комитета РСФСР по делам издательств, полиграфии и книжной торговли на Смоленском полиграфком-бинате Росглавполиграфпрома Государственного комитета РСФСР по делам издательств, полиграфии и книжной торговли. Смоленск, 20, м/р Поповка.