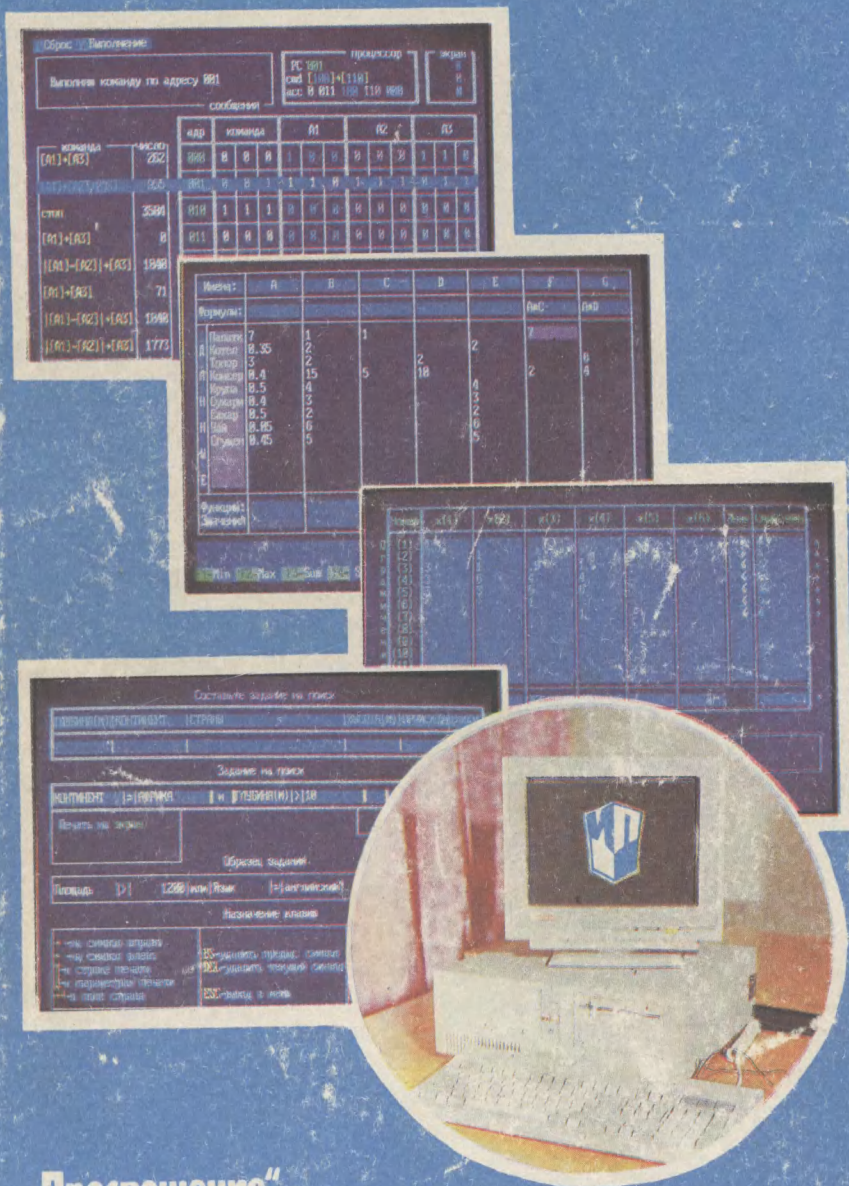


# ИНФОРМАТИКА



## “Просвещение”

F1- Чтение с диска

F2- Запись на диск

F3- Печать текста

F4-

edit

## РЕДАКТИРОВАНИЕ ТЕКСТОВ С ПОМОЩЬЮ ЗВИ

"Ванька Жуков ... достал из хозяйского шкафа пузырек с чернилами, ручку с заржавленным пером и, разложив перед собой измятый лист бумаги, стал писать... Бумага пела на скамье, а сам он стоял перед скамьей на коленях. "Милый дедушка, Константин Махарыч! писал он.- И пишу тебе письмо..."

Номер строки: 8

Номер столбца: 8

## Автоматизированный склад

1	3	6		7	8			7	5	4	2
	5		2	1	2	5	5		3		7
	6	8		3	5	7	9	9		4	

ГРУЗОВОЙ

ОТСЕК

Выбранная деталь:

5

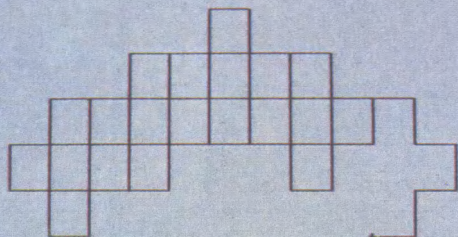
- 1 начать работу
- 5 к началу ряда
- 10 выбрать деталь
- 20 пока можно вправо повторять
- 30 если выбранная деталь то
- 40 взять деталь
- 50 конец ветвления
- 60 шаг вправо
- 70 конец цикла

Условие "можно вправо?" верно  
Условие "выбранная деталь?" верно

Остановить F2 Нажимать ВВОД F3

F4

F5



315 повернуть направо  
316 повернуть направо  
317 прыгнуть  
318 прыгнуть  
319 повернуть направо  
320 конец подпрограммы  
пуск

F1 Остановить F2

F3

F4

F5

текст  
1 запросить C,D,K  
2  $N=0$   
3 пока  $C>D$  повторять  
4  $M=N+30$   
5  $C=C/K*30$   
6 сообщить "номер суток"  
7 сообщить N  
8 сообщить "концентрация"  
9 сообщить C  
10 конец цикла  
пуск  
 $C=10$   
 $D=0.03$   
 $K=1.12$

F1 Остановить F2

F3

F4

F5

# ИНФОРМАТИКА

---

**Учебник**  
**для 8 — 9 классов средней школы**

*Рекомендовано*  
*Министерством образования*  
*Российской Федерации*

Москва "Просвещение" 1994

ББК 73я72  
И74

Авторы:

А.Г. Гейн, Е.В. Линецкий, М.В. Сапир, В.Ф. Шолохович

И74      **Информатика: Учеб. для 8—9 кл. сред. шк. /А.Г. Гейн, Е.В.Линецкий, М.В. Сапир, В.Ф. Шолохович. — М.: Просвещение, 1994. — 256 с.: ил. — ISBN 5-09-005113-5.**

и 4306024000 — 440 ДБЗ 1994/95 — 64  
103 (03) — 94

ББК 73я72

**Учебное издание**

**Гейн Александр Георгиевич  
Линецкий Евгений Викторович  
Сапир Марк Валентинович  
Шолохович Владимир Фридрихович**

**ИНФОРМАТИКА**

**Учебник**

**для 8 — 9 классов средней школы**

**Зав. редакцией Т.А. Бурмистрова  
Редактор А.К. Компанец  
Младшие редакторы Л.И. Заседателева, Н.Е. Терехина  
Художник В.В. Костин  
Художественный редактор Е.Р. Дашук  
Технический редактор И.Ю. Щукина  
Оператор А.В. Туманов  
Корректор О.В. Ивашкина, Л.Г. Новожилова**

**ИБ № 15247**

Набор и верстка выполнены в издательстве "Просвещение" на компьютерной технике с использованием редакционно-издательской системы Wave4™ Bestinfo, Inc., гарнитуры из библиотеки цифровых шрифтов PageType™. Диапозитивы изготовлены в издательстве "Просвещение" на фотопленке совместного производства А.О. Фототех и издательства "Просвещение".

Лицензия ЛР №010001 от 10.10.91. Подписано к печати 13.07.94. Формат 60х90<sup>1/16</sup>. Бумага офс. №1. Гарнитура школьная. Печать офсетная. Усл. печ. л. 16 + 0,31 форз. Усл. кр.-отт. 33,37. Уч.-изд. л. 13,89 + 0,42 форз. Тираж 397 000 экз. Заказ № 1044.

Ордена Трудового Красного Знамени издательство "Просвещение" Комитета Российской Федерации по печати. 127521, Москва, 3-й проезд Марьиной рощи, 41.

Саратовский ордена Трудового Красного Знамени полиграфкомбинат Комитета Российской Федерации по печати. 410004, Саратов, ул. Чернышевского, 59.

**ISBN 5-09-005113-5**

© Гейн А.Г. и другие, 1994

## ПРЕДИСЛОВИЕ

---

Вы уже, конечно, слышаны о многих чудесных свойствах компьютеров. Учебник, лежащий перед вами, поможет вам познакомиться с ними поближе. И в первую очередь научиться применять компьютер для решения разнообразных задач. Ведь вся наша жизнь — постоянное решение задач. Как починить велосипед и как вывести на орбиту космический корабль, как воспитать подаренного вам щенка и что предпринять для спасения редкой породы птиц, как проявить фотопленку и как рационально использовать запасы нефти и газа — многообразие жизненных задач беспредельно. Конечно, каждый школьный предмет предлагает вам свой подход к решению задач. Скажем, проблемы охраны природы можно обсуждать и на уроках математики, и на уроках химии, и на уроках литературы.

Есть свой подход к решению задач и у информатики. Суть его — в использовании ЭВМ. ЭВМ можно применять для решения задач практически из всех областей жизни. Но для того, чтобы делать это эффективно, нужно уметь:

- *четко формулировать (часто говорят: ставить) задачу;*

- *составлять алгоритмы, т. е. организовывать последовательности действий, приводящих к требуемому результату;*

- *пользоваться специальными средствами, которыми снабжают любую ЭВМ, чтобы облегчить общение человека с машиной.*

Овладев этими умениями, вы "приручите" ЭВМ, и она станет вашим надежным другом.

## КАК РАБОТАТЬ С УЧЕБНИКОМ?

Желательно, конечно, чтобы вы имели доступ к ЭВМ, снабженной комплексом учебных программных средств — "программной поддержкой" нашей книги.

Подробные описания этих программ приведены в учебнике. Их роль при изучении информатики можно сравнить с ролью циркуля и линейки при изучении геометрии и ролью физических приборов при изучении физики. Впрочем, не стоит сильно огорчаться, если в вашем компьютерном классе используются программы, несколько отличающиеся от описанных в учебнике, — учитель объяснит вам, как выполнить лабораторные работы в этом случае.

В нашей книге, как и в любом учебнике, вам встретятся новые термины. Для удобства они напечатаны жирным шрифтом. Определения, свойства и правила выделены курсивом. Разумеется, их не надо заучивать наизусть. Важно понимать их смысл и уметь применять их на практике. Если вы хотите быстро найти, что означает то или иное слово, воспользуйтесь предметным указателем. Он расположен в конце книги.

Никакое обучение невозможно без самоконтроля. Вопросы к параграфам помогут вам понять, хорошо ли вы усвоили новый материал. Книга содержит много (около 500) разнообразных задач — простых и посложнее. Более трудные (на наш взгляд) задачи помечены знаком \*. Мы надеемся, что каждый из вас найдет себе задачи по вкусу.

Решение некоторых задач необходимо записать так, чтобы им можно было воспользоваться в дальнейшем. После номера такой задачи стоит знак °.

При повторении изученного материала вам должны помочь конспекты глав — сводки основных содержащихся в них фактов, понятий, идей. Возможно, некоторые из вас предпочтут сначала ознакомиться с кратким содержанием главы и, лишь встретив непонятное место, обратиться к основному материалу. Что ж, дело вкуса, ничего плохого в этом нет.

В тексте вам встретятся короткие (и не вполне серьезные) фразы. Иногда они в сжатой форме выражают суть нового материала, а порой освещают его с неожиданной стороны. Кроме того, мы считаем, что временами и о серьезных вещах можно говорить в шуточной форме. В общем, надеемся, что читать нашу книгу вам будет не только полезно, но и приятно.

## Глава 1

# ЗНАКОМСТВО С ЭВМ

---

Пройдет не так уж много лет, и человек, не знакомый с ЭВМ, не привыкший обращаться к ее помощи в повседневной практике, окажется совершенно не приспособленным к жизни. Управление сложнейшими автоматизированными процессами, быстрая переработка необозримых объемов научно-технической, политической, экономической и другой разнообразной информации станут уделом не сотен и не тысяч, а миллиардов людей, практически каждого из вас. Информация — постоянный спутник человека. Это те сведения, которые помогают нам ориентироваться в окружающем мире. Можно сказать, что информация необходима человеку как воздух и вода. И если в предыдущие века человек имел дело только с "ручейками" информации, то теперь его окружают бездонные "моря" разнообразных сведений, способные поглотить любого в своих пучинах. Легко перейти через ручей. Но чтобы переплыть море, нужны корабли и навигационные карты, нужна наука о кораблях и кораблевождении. Информатика — это наука о навигации в "Тихом океане" информации, ЭВМ — океанские лайнеры, покоряющие информационные просторы. (Не правда ли, красиво сказано?) Если говорить более точно, **информатика изучает технологию сбора, хранения и переработки информации, а ЭВМ — основной инструмент в этой технологии.**

### § 1. ЧТО УМЕЕТ ЭВМ

Люди всегда стремились облегчить свой труд. Они придумывали разнообразные приспособления, механизмы и машины, усиливающие различные физические возможности человека. Но лишь очень немногие механизмы помогали человеку лучше выполнять умственную работу. С этим можно было мириться в течение сотен лет, пока большинство людей занималось в основном физическим трудом. Однако за последние несколько десятилетий все изменилось.

Ныне почти половина всех работающих в развитых странах занимается исключительно умственной деятельностью. Ясно, что теперь без машин, способных резко усилить умственные возможности человека, просто не обойтись.

Первая такая машина появилась в 1945 г. Она называлась ЭНИАК и предназначалась для расчета артиллерийских таблиц. Сегодня же "умных" машин миллионы. Это — электронно-вычислительные машины (ЭВМ). Часто используют и другое название: компьютеры. Они предсказывают погоду и результаты футбольных матчей, управляют стыковкой космических кораблей и заводами-автоматами, играют в шахматы и рисуют мультфильмы. Вряд ли среди вас найдется человек, который никогда не видел ЭВМ хотя бы по телевизору. Ведь редкая рекламная пауза обходится без предложения купить компьютер или программы к нему.

На лабораторных работах вы убедитесь, что компьютер обладает многими способностями человека. Одна из них — умение мысленно имитировать поведение механизмов, живых существ и даже таких объектов, которых в природе не существует. Вы увидите, что эта способность в некоторой степени присуща и ЭВМ. Именно поэтому с помощью компьютера можно еще до воплощения нового самолета в металле узнать его поведение в экстремальных погодных условиях, без карандаша и бумаги по разрозненным описаниям свидетелей создать портрет преступника, без преподавателя научиться печатать на пишущей машинке.

Человеку, конечно, мало, чтобы машина просто обладала какими-то его способностями. Ведь машины для того и существуют, чтобы в чем-то быть лучше человека: экскаватор быстрее выкопает траншею, подъемный кран поднимет большой груз, станок точнее выточит деталь. Электронно-вычислительная машина заткнет за пояс любого из вас в умении быстро и точно вычислять. ЭВМ и были созданы прежде всего для того, чтобы освободить человека от громоздких и утомительных вычислений.

А всего несколько десятилетий назад существовала многочисленная армия людей, которые всю свою трудовую жизнь занимались вычислениями. Опытный вычислитель тратил на выполнение одного арифметического действия около 20 секунд. Современные ЭВМ выполняют десятки миллионов операций в секунду. Всего за 40 лет благодаря компьютерам скорость проведения вычислений возросла в сотни миллионов раз. Для сравнения скажем, что если бы скорость ракеты во столько же раз превосходила скорость пешехода, то ракеты летали бы со скоростью света.

Быстродействие ЭВМ позволило легко решить многие практические задачи, требующие больших объемов вычис-

лений. До появления ЭВМ решение каждой такой задачи было самым настоящим научным подвигом.

Так, великий математик Л. Эйлер потерял зрение после трехдневного напряженного труда по расчету траектории кометы.

ЭВМ может решить эту задачу за считанные минуты.

Навсегда в историю науки вошло открытие "на кончике пера" планеты Нептун. У. Леверье рассчитал ее траекторию, проанализировав результаты наблюдений за планетой Уран. Он потратил на расчеты более трех лет.

ЭВМ потратила бы несколько часов.

Подобные вычисления ЭВМ проводят постоянно. Ведь ежедневно нужно рассчитывать траектории полета сотен спутников.

Конечно, труд Эйлера, Леверье и других подвижников не пропал даром. Без методов вычислений, созданных ими, многие важные задачи так и остались бы нерешенными, даже если бы ЭВМ работали в миллионы раз быстрее. О некоторых из этих методов будет рассказано в последующих главах.



## Лабораторная работа 1

### ПЕРВЫЙ РАЗ В ДИСПЛЕЙНОМ КЛАССЕ

Вы пришли в дисплейный класс. Прежде всего нужно позаботиться о безопасности работы на ЭВМ. Необходимо помнить, что к каждому рабочему месту подведено опасное для жизни напряжение. Если вы обнаружили какую-либо неисправность, немедленно сообщите об этом преподавателю.

ЭВМ, стоящая на вашем столе, за свои небольшие размеры называется микроЭВМ. Тем не менее может она многое — гораздо больше, чем ее "прабабушка" ЭНИАК, занимавшая несколько комнат. Конечно, каждому из вас интересно, как устроена микроЭВМ. Сейчас расскажем.

Самая главная часть ЭВМ — **процессор**. Именно он руководит всей работой ЭВМ, осуществляя связь между остальными частями ЭВМ, а также между ЭВМ и человеком. Конечно, "главнокомандующий" — процессор сам всего лишь исполняет команды, отдаваемые человеком. Правда, работа процессора не требует постоянного вмешательства человека: можно заранее написать инструкцию-программу, состоящую из команд, понятных процессору. А уж процессор в соответствии с этой инструкцией будет осуществлять руководство (подробнее об этом мы расскажем в последней главе учебника). Чем больше команд умеет "понимать" процессор, тем ЭВМ "умнее"; чем больше скорость

его работы, тем ЭВМ "сообразительнее". Быстродействие процессора микроЭВМ, на которой вы работаете, свыше 300 тыс. команд в секунду.

Но ЭВМ была бы бесполезна, если бы она не могла запоминать информацию, необходимую для решения задачи. Для хранения информации предназначено особое устройство — **память**. Основной вид памяти — **оперативная память** — находится, как и процессор, внутри ЭВМ. В этой памяти машина способна держать, например, около 48 страниц текста. Но часто информации требуется значительно больше. Ведь и человек, имеющий гораздо большую "оперативную память", не может все упомнить. Поэтому мы пользуемся записными книжками, магнитофонными лентами, видеокассетами и т.п. Подобные "записные книжки" — их называют **внешней памятью** — имеются и у ЭВМ. Для микроЭВМ это, главным образом, магнитные диски (дискеты) и магнитофонные кассеты. На одной дискете может храниться до 600 страниц текста. Как видите, на одну дискету можно поместить несколько школьных учебников.

Аналогия с человеком наверняка подсказывает вам, что ЭВМ должна иметь нечто заменяющее ей органы чувств (чтобы, например, воспринимать указания человека), а также какие-то устройства, позволяющие ей рапортовать о выполнении заданий.

И действительно, каждая ЭВМ снабжена такими устройствами. Они называются **устройствами ввода-вывода**. С помощью **клавиатуры** человек дает ЭВМ задания, с помощью **дисководов** (или магнитофона) ЭВМ получает информацию из внешней памяти, а итоги своей работы ЭВМ печатает на экране дисплея или на бумаге при помощи **принтера**.

А теперь познакомимся поближе с клавиатурой ЭВМ. В этом вам поможет сама ЭВМ и обучающая программа "Клавиатура". Ну как — освоились с клавиатурой? Мы надеемся, что к концу девятого класса вы научитесь самостоятельно составлять программы.

А пока мы расскажем, как заставить ЭВМ выполнять самые простые указания. Сейчас ассистент подготовит ЭВМ к этой работе... Готово? Тогда начнем. В верхней строке экрана указаны значения некоторых клавиш. Вы научитесь пользоваться ими чуть позже, поэтому пока не обращайтесь к ним внимания.

Давайте заставим ЭВМ считать. Сначала что-нибудь попроще. Пусть ЭВМ подсчитает, чему равно  $2+2$ . Для этого надо набрать на клавиатуре команду:

**СООБЩИТЬ 2+2**

(Слово "СООБЩИТЬ" надо употреблять, если ЭВМ настроена

на имитацию исполнителя **ВЫЧИСЛИТЕЛЬ**. Если же ЭВМ настроена на язык Бейсик, то вместо него надо использовать слово **PRINT**.)

Мало сформулировать команду, надо еще заставить ЭВМ выполнить ее. Поэтому, закончив набор команды, нужно нажать клавишу "**ПЕРЕВОД СТРОКИ**", иначе ЭВМ команду не воспримет. Нажмите же клавишу "**ПЕРЕВОД СТРОКИ**"! Видите — на экране появилось число 4. Впрочем, у некоторых из вас, возможно, появилось сообщение о том, что ЭВМ не поняла вашу команду. Это означает, что команда набрана неверно. Исправьте команду и получите результат.

*Ты нажал клавишу  
"ПЕРЕВОД СТРОКИ"?*

Командой "**СООБЩИТЬ**" можно заставить ЭВМ вычислить значение любого числового выражения. Например, чтобы ЭВМ подсчитала произведение чисел 1234 и 5678, надо дать ей команду:

**СООБЩИТЬ 1234\*5678**

Обратите внимание, что символом умножения является звездочка (\*). При записи выражений звездочку нельзя опускать или заменять другим символом (иначе ЭВМ вас не поймет или поймет превратно). Для деления используется знак /. Например, частное чисел 12345 и 670005 записывается так: 12345/670005. Другие знаки, скажем, горизонтальную дробную черту, использовать также нельзя. Для возведения в степень нужно использовать символ ^. Например, 434 в степени 5 записывается так: 434^5. И еще один уговор: при записи десятичных дробей ставится точка, а не привычная вам запятая. Например, вместо 1,2123 нужно писать 1.2123.

Заставьте ЭВМ вычислить выражение

$$\frac{(12,34:23-36,45^2)(-298,05+123:1,1)}{5657,323+344,444}$$

У нас получилось 41,209220850616. А у вас? Обратите внимание, как быстро ЭВМ вычислила это значение. А какая точность! Просто замечательно!

Есть, правда, одно неудобство: слишком долго приходится набирать команды. Чтобы облегчить эту работу, имеются специальные клавиши — они называются **функциональными**. При нажатии такой клавиши на экране появляется не один символ, а сразу целое слово — значение этой клавиши. В

*Пользуйтесь  
функциональными  
клавишами —  
это удобно  
и надежно!*

специальной строке экрана как раз и записаны значения функциональных клавиш. Их можно менять с помощью специальной клавиши "ВЫБОР". Нажмите на нее несколько раз, чтобы значением одной из функциональных клавиш стало слово "СООБЩИТЬ". Теперь достаточно одного нажатия на функциональную клавишу, чтобы слово "СООБЩИТЬ" появилось на экране. Воспользуйтесь этим и подсчитайте значение какого-нибудь интересного вам числового выражения.

Вот вы и познакомились с ЭВМ. Надеемся, что знакомство было приятным.

## § 2. ОРГАНИЗАЦИЯ ВЫЧИСЛЕНИЙ С ПОМОЩЬЮ ЭВМ

На первой лабораторной работе вы узнали, как одной командой заставить ЭВМ вычислить значение сложного выражения. Она выполняет эти команды быстро и с высокой точностью. Ну а если надо много раз проводить однообразные вычисления? Тогда для вычисления каждого выражения придется записывать специальную команду. Это весьма утомительное дело. А ведь с большим количеством однообразных вычислений приходится сталкиваться довольно часто. Вот типичная ситуация.

В августе все готовятся к новому учебному году. И школьники, и учителя. Но особенно много хлопот у завуча: составляется расписание уроков. Завучу надо учесть большое количество разных факторов: требования учебного плана, пожелания учителей, количество учебных помещений и так далее. И вот расписание составлено. Однако заботы завуча на этом не кончаются. Вдруг оказывается, что заболел один из учителей или не закончен ремонт в одном из классов. Каждое такое происшествие требует переделки всего расписания. При этом завучу приходится заново учитывать огромный объем данных и связей между ними.

Подобные проблемы возникают не только у завучей. Диспетчерам при составлении графиков движения транспорта, бухгалтерам при составлении сметы, экспериментаторам при проведении серий опытов — всем им приходится решать задачи, в которых изменение значения какого-то одного параметра требует пересчета большого числа результатов. И здесь компьютер готов сослужить добрую службу. Для этого программистами созданы специальные программы — электронные таблицы.

Электронная таблица позволяет хранить в табличной фор-

*Если вычислений  
много,  
а времени мало,  
доверьтесь  
электронным  
таблицам*

ме большое количество исходных данных, результатов, а также связей (математических соотношений) между ними. Но главное — при изменении исходных данных все результаты автоматически пересчитываются и заносятся в таблицу.

На лабораторной работе вы познакомитесь с учебной электронной таблицей и с ее помощью решите следующую задачу.

**Задача о трех туристах.** Три одноклассника собрались пойти в пятидневный поход. Посоветовавшись, они составили список того, что нужно взять с собой (не считая личных вещей):

Что брать	Сколько брать
Палатка	1 штука
Котелок	2 штуки
Топор	2 штуки
Консервы	15 банок
Крупа	4 пачки по 0,5кг
Сухари	3 коробки
Сахар	2 пачки по 0,5кг
Чай	6 пачек по 50 г
Сгущенка	5 банок

Эти вещи они решили разделить по справедливости: на три равные по весу части. Как это сделать?

Понятно, что результатом решения задачи будет информация о том, кто, что и сколько несет с собой. Распределение вещей может быть, например, таким: первый школьник возьмет с собой палатку и пять банок консервов, второй — оставшиеся консервы и топоры, а третий —



все остальное. Эту информацию удобно расположить в три столбца (по числу приятелей). Получится такая таблица :

Что брать	Сколько брать	А	Б	В
Палатка	1 штука	1		
Топор	2 штуки		2	
Котелок	2 штуки			2
Консервы	15 банок	5	10	
Крупа	4 пачки			4
Сухари	3 коробки			3
Сахар	2 пачки			2
Чай	6 пачек			6
Сгущенка	5 банок			5

Как видите, мы обозначили школьников буквами А, Б, В (настоящие их имена нам неизвестны). Правда, по этой таблице еще нельзя понять, удалось ли приятелям распределить груз по справедливости. Для этого надо знать, сколько весит каждый предмет из списка. Дополним таблицу еще одним столбцом, содержащим эти сведения. Поставим этот столбец после столбца "Что брать":

Что брать	Вес	Сколько брать	А	Б	В
Палатка	7	1 штука	1		
Котелок	0,35	2 штуки			2
Топор	3	2 штуки		2	
Консервы	0,4	15 банок	5	10	
Крупа	0,5	4 пачки			4
Сухари	0,4	3 коробки			3
Сахар	0,5	2 пачки			2
Чай	0,05	6 пачек			6
Сгущенка	0,45	5 банок			5

Подсчитайте и определите, является ли такой раздел справедливым. На наш взгляд, нет: ведь Б будет нести почти в полтора раза больший груз, чем В. Значит, надо перераспределить вещи и снова вычислить, какой вес понесет каждый из приятелей. Попробуйте распределить вещи по-другому, более равномерно. Скорее всего, опять не удастся добиться справедливости. Для этого, возможно, придется совершить еще несколько попыток. Ведь каждый раз, "переложив" какой-либо предмет из одного рюкзака в другой, придется выполнить немало арифметических действий,

и лишь затем станет ясно, приблизились ли мы к справедливому распределению. Вот если бы эта таблица сама могла вычислять! Что ж, это не фантазия, не сказка "По щучьему велению". Нашу таблицу можно "оживить" с помощью учебной программы "Электронная таблица". Этим вы и займетесь на лабораторной работе.



## Лабораторная работа 2

### УЧЕБНАЯ ЭЛЕКТРОННАЯ ТАБЛИЦА

Перед вами на экране появилась таблица, большинство клеток которой пусты. Это начала работать учебная электронная таблица. С ее помощью вы будете решать задачу о трех туристах. Сначала нужно определенным образом записать в таблицу исходные данные.

Как это сделать?

Имя	A	B	C	D	E	F	G	H	I
F1 — Min		F2 — Max		F3 — $\Sigma$		F4 — $\Sigma/n$		F5 — $\Sigma^2/n$	

Для начала следует познакомиться с устройством таблицы. В ее верхней строке записаны названия столбцов. В столбцах, помеченных буквами, будут стоять числа, а в столбце "Имя" — названия строк. Первые две и последние три строки таблицы особые. Первую строку изменять нельзя. Вторая строка — для записи алгебраических выражений, в которые могут входить переменные A, B, C, D, E, F, G, H, I. Сразу скажем, что в выражение, записанное в каком-либо столбце, могут входить только переменные из предыдущих столбцов. Например, если вы записываете выражение во вторую клетку столбца C, то в него могут входить лишь переменные A и B. Переменная I ни в какое выражение входить не может.

Вы уже видели на лабораторной работе 1, как по нажатию функциональной клавиши сразу появляется целое слово, заставляющее ЭВМ выполнить нужное вам действие. Этим полезным качеством обладает и электронная таблица.

Только действия здесь другие — те, которые часто приходится выполнять, работая с таблицами. Приглядитесь внимательнее — в последней строке написано, какая клавиша за что "отвечает". Можно вычислять минимальный и максимальный элементы столбца (эти действия обозначены надписями Min и Max). Сумма элементов столбца вычисляется с помощью функциональной клавиши F3, обозначенной символом  $\Sigma$ ; за вычисление среднего арифметического и среднего арифметического квадратов отвечают клавиши F4 и F5.

Если нажать на функциональную клавишу, то ее значение появится под тем столбцом, в котором стоит курсор, а еще ниже — результат.

Имя	A	B	C	D	E	F	G	H	I
						A*C	A*D	A*E	
Палатка	7	1	1			7			
Котелок	0,35	2			2			0,7	
Топор	3	2		2			6		
Консервы	0,4	15	5	10		2	4		
Крупа	0,5	4			4			2	
Сухари	0,4	3			3			1,2	
Сахар	0,5	2			2			1	
Чай	0,05	6			6			0,3	
Сгущенка	0,45	5			5			2,25	
						$\Sigma$	$\Sigma$	$\Sigma$	
						9	10	7,45	
F1 — Min		F2 — Max		F3 — $\Sigma$		F4 — $\Sigma/n$		F5 — $\Sigma^2/n$	

При заполнении таблицы можно передвигать курсор по экрану с помощью стрелок, а также пользоваться клавишами "ВСТАВКА" и "УДАЛИТЬ СИМВОЛ"; для быстрого перехода из клетки таблицы в соседнюю справа клетку используйте клавишу "ПЕРЕХОД" (обычно на этой клавише написано "TAB").

Начнем заполнять таблицу. В столбец "Имя" запишите названия вещей. В столбцы A, B — их веса и количества. Столбцы C, D, E будут соответствовать столбцам A, B, B из таблицы 2 — заполните их в соответствии с этой таблицей.

Теперь пора "оживить" таблицу. Мы будем изменять числа в столбцах C, D, E (перераспределять вещи между туристами), а таблица сама будет подсчитывать груз, доставшийся каждому туристу. Чтобы подсчитать вес вещей, доставшихся, например, туристу B, нужно найти произведения соответствующих элементов столбцов A (вес вещи) и D (количество вещей, взятых туристом B), а затем сложить получившиеся числа. Запишите во вторую сверху клетку столбца F формулу A\*C. Видите — в столбце F появились

числа 7 и 2. Ясно, что 7 — это вес одной палатки, а 2 — вес пяти банок консервов, которые получит турист А. Заставим таблицу подсчитать общий вес груза, доставшегося А. Функциональная клавиша F3 как раз и предназначена для подсчета суммы элементов столбца (помните?). Не выводя курсор из столбца F, нажмите на клавишу F3. Под столбцом F появится знак  $\Sigma$  (так обычно обозначают суммирование) и число 9 — столько килограммов "общественного" груза понесет первый школьник. Точно так же можно поручить таблице подсчитать вес грузов, предназначенных второму и третьему участникам похода, используя для этого столбцы G и H.

Вот как будет выглядеть таблица на экране вашей ЭВМ после занесения в нее исходных данных и математических формул (см. табл. 4). Как видите, второму туристу досталось больше всех. Несправедливо! Меняя числа в столбцах C, D и E, попробуйте перераспределить груз более равномерно. Авторы сами проделали эту работу и добились-таки справедливости. Правда, увлекшись перераспределением груза, мы "потеряли" две банки консервов. И даже появление лишнего топора не могло послужить утешением. Мы нашли выход: поручили таблице самой следить за "сохранностью" вещей. Для этого мы занесли в столбец E формулу... А какую — догадайтесь сами.

### § 3. ИНФОРМАЦИОННО-ПОИСКОВЫЕ СИСТЕМЫ

Представьте себе, что у вас есть небольшая студия звукозаписи, в которой вы и директор, и оператор, и продавец записей. К вам постоянно обращаются клиенты в надежде получить интересующую их запись. Кому-то требуется последний хит сезона, у кого-то есть любимый исполнитель, другие испытывают ностальгию по музыке давно прошедших лет. Вы процветаете; одно удручает вас — медленно движется очередь, поскольку нелегко в длинных списках отыскать нужное название.

Мало помогают и выложенные на прилавок списки записей. Вот только три взятых наугад заказа. Первому клиенту надо подобрать все записи английских и американских ансамблей стиля "рэп", сделанные в прошлом году. Второму — все записи группы, в которой играл его любимый гитарист в 80-е годы; более того, привередливый заказчик желает, чтобы записи обязательно располагались в хронологическом порядке. Третий клиент — диск-жокей, готовящийся к выступлению на дискотеке с рассказом об истории разных стилей рок-музыки.

Но коллекционируют ведь не только музыкальные записи. Наверняка каждый из вас пережил увлечение собирательством марок (этикеток, календариков...). Истин-

ный коллекционер не станет складывать как попало дорогие ему экспонаты, а рассортирует их по тем или иным признакам. Для марок набор признаков перечислит всякий — это, например, страна, выпустившая марку, год выпуска, тема, гашеная марка или нет... Учитывается даже состояние (сохранность зубцов, наличие клея и др.). Всю эту информацию заядлые коллекционеры хранят в специальных каталогах. Однако, даже имея каталог, порой приходится тратить много усилий, чтобы подобрать нужный вариант обмена.

Описанные ситуации имеют много общего: в большом объеме информации разыскивается та, которая нужна в данный момент. И здесь снова надежными помощниками становятся компьютеры. Чтобы облегчить хранение и поиск информации, созданы специальные программы — **информационно-поисковые системы (ИПС)**. Как правило, ИПС позволяют оперировать с большим количеством сведений об однотипных объектах. При этом для каждого из объектов существенными являются значения лишь некоторых признаков. Что понимается здесь под словами "признак" и "значение признака"? Поясним на примерах.

В первом из разобранных примеров объектами являются записи, а их признаками, например, будут название ансамбля и год выхода пластинки (компакт-диска, кассеты). Значения этих признаков могут быть, скажем, такими: "Битлз", 1966. Признаки марки (второй пример) уже указаны выше. Примеры значений этих признаков вы легко приведете сами, рассматривая какую-нибудь диковинную марку.

Каждая конкретная ИПС "нацелена" на решение своего круга задач. В свою очередь, для каждого класса задач характерен свой набор объектов и их признаков. Поэтому и соответствующие ИПС будут различными — одна ИПС нужна, чтобы разыскивать марку по каталогу, и совсем другая — для облегчения работы директора студии звукозаписи. Впрочем, для разных классов задач объекты могут быть одинаковыми, а наборы нужных признаков — разными. Например, овощевода интересуют урожайность, время посева и сроки уборки овощей, повара — калорийность овощей и рецепты овощных блюд, а директора овощехранилища — условия хранения.

Любая ИПС состоит из двух частей: большой специально организованной совокупности данных (она называется **базой данных**) и программы, позволяющей оперировать ими. Данные, хранящиеся в базе данных, — это значения заранее фиксированных призна-

**Данных  
надо позарез?  
Обращайтесь к ИПС!**

ков некоторого набора объектов. А "оперировать" — значит находить объекты по заданным признакам, изменять и дополнять сведения об объектах, а также решать некоторые другие задачи. Для того чтобы заставить ЭВМ найти интересующие вас сведения, нужно составить запрос. Правила записи запросов для каждой ИПС свои. Эти правила устанавливаются теми, кто создает ИПС.

К настоящему времени в мире созданы сотни тысяч ИПС. Они используются в библиотеках и больницах, в гидрометеоцентрах и на заводах, в магазинах и планирующих организациях. Некоторые из них объединены в крупные, централизованные ИПС, часто называемые банками данных. В нашей стране действуют десятки банков данных. Пожалуй, самый большой из них — банк данных Российского института научной и технической информации. В нем содержится более 6 млн. библиографических сведений о книгах и статьях практически по всем отраслям знаний. Многие из вас, пользуясь услугами Аэрофлота, имели дело с другим крупным банком данных нашей страны — системой "Сирена". С помощью микроЭВМ кассир Аэрофлота связывается с центральной "большой" ЭВМ, находящейся за тысячи километров от него, почти мгновенно получает сведения о наличии мест на данный рейс и печатает билет.

На лабораторной работе вы познакомитесь с небольшой учебной ИПС "Озера". В ней хранятся следующие сведения об озерах мира:

- 1) название озера;
- 2) площадь (в кв. км);
- 3) максимальная глубина (в м);
- 4) континент;
- 5) страны, на территории которых расположено озеро;
- 6) высота над уровнем моря (в м);
- 7) происхождение (тектоническое, лагуна, вулканическое, запрудное, ледниковое, карстовое);
- 8) наличие или отсутствие стока.

Вы, конечно, понимаете, что в этой ИПС может не оказаться информации об озере, на берегу которого вы отдыхаете летом. Нельзя ведь объять необъятного! Поэтому в нашу ИПС попали наиболее известные озера мира.

Какие же задачи можно решать с помощью этой ИПС? Самая простая задача — получить сведения о том или ином озере. Можно и, наоборот, найти все озера, имеющие некоторый общий признак. Скажем, найти все озера Евразии или определить, какие озера имеют площадь больше тысячи квадратных километров.

Наша ИПС "умеет" решать и более сложные задачи. Например, указать все африканские озера, имеющие глубину свыше 20 м. ИПС "Озера" позволяет получать дополнитель-

ные сведения — можно, скажем, найти суммарную площадь американских озер тектонического происхождения, определить самое глубокое и самое мелкое озеро какой-либо страны.

Наконец, с ее помощью можно изменять какие-либо устаревшие сведения об озерах (к примеру, изменилось название страны, на территории которой расположено озеро).

Для поиска информации в ИПС "Озера" в запросе указываются название признака (одного из восьми перечисленных выше) и его значение. Для того чтобы получить сведения о Байкале, надо указать признак "Название озера" и его значение "Байкал". Если мы хотим найти все австралийские озера, то указывается признак "Континент" и его значение "Австралия".

А как заставить ЭВМ найти все озера, отличные от Байкала? Ведь "Не Байкал" не является значением признака "Название озера"! Если же речь идет, скажем, о площади, то возможных вариантов еще больше: нас могут интересовать озера, у которых площадь больше или меньше некоторого значения. Чтобы решить подобную задачу, в запросе указывают, нужны ли нам озера, для которых признак равен или не равен заданному значению, больше, меньше, не больше или не меньше заданного значения. Например, если нас интересует Байкал, нужно составить следующий запрос:

Название озера равно "Байкал".

Чтобы ЭВМ нашла озера, отличные от Байкала, нужен такой запрос:

Название озера не равно "Байкал".

Для поиска озер, глубина которых больше 20 м, запрос выглядит так:

Глубина больше 20.

А как быть, если вы не помните, скажем, полное название озера или страны, но уверены, например, что в нем встречается сочетание "ар"? Наша ИПС и в этом случае придет вам на выручку. Просто в запросе надо вместо неизвестных частей слова поставить \*. К примеру,

Название страны равно "\*ар\*".

Разумеется, ИПС найдет все озера, омывающие страны, названия которых содержат такое сочетание букв. В нашем случае это будут озера: Боденское и Женевское (страна — Швейцария), Вьедма и Архентино (страна — Аргентина) и др. Знак "\*" означает произвольное начало или произвольный конец слова. Поэтому запись \*А означает произволь-

ное слово, оканчивающееся на А, а запись А\* — произвольное слово, начинающееся на А.

Для решения более сложных задач поиска (скажем, той, которая сформулирована выше, — указать все африканские озера, имеющие глубину свыше 20 м) формируют более сложный запрос. Он составляется из простых запросов с помощью союзов "и" и "или" — так же, как в русском языке сложносочиненное предложение составляется из простых. В приведенном примере запрос выглядит так:

Континент равен "Африка" и глубина больше 20.

Если же нас интересует список озер, омывающих одну из двух стран — Финляндию или Швецию, запрос будет таким:

Страна равна "Финляндия"  
или страна равна "Швеция".

Возможности ИПС "Озера" не ограничиваются поиском озер. Можно, скажем, расположить озера по алфавиту или упорядочить их по возрастанию площади.

## ? Вопросы

1. Что такое база данных?
2. Что такое информационно-поисковая система?
3. Что такое банк данных?
4. Какие операции можно производить с данными, хранящимися в базе данных?

## Задания для самостоятельного выполнения

1. Запишите запросы для поиска с помощью ИПС "Озера":

- а) всех озер-карликов (площадь меньше 100 км<sup>2</sup>);
- б) всех бессточных озер;
- в) всех озер с площадью более 10 000 км<sup>2</sup> и глубиной более 500 м;
- г) всех озер, расположенных выше 500 м, кроме африканских;
- д) всех американских озер;
- е) всех озер, названия которых начинаются с буквы "Л";
- ж) всех озер, омывающих Танзанию.

2. Какой запрос надо сформулировать, чтобы выяснить: а) все ли озера ледникового происхождения мельче 100 м;

б) все ли озера мельче 100 м имеют ледниковое происхождение?

3. Однажды школьник решил воспользоваться ИПС "Озера", чтобы найти все африканские и австралийские

озера. Тут подошел злоумышленник и быстро составил следующий запрос:

Континент равен "Африка" и континент равен "Австралия".

ИПС сообщила, что ей неизвестны озера, удовлетворяющие этому запросу. Объясните почему. Какой запрос нужно составить, чтобы получить от ИПС нужную информацию?

4. Какой запрос надо составить, чтобы узнать:

а) на каких континентах имеются озера ледникового происхождения;

б) в какой стране располагается самое высокогорное озеро?



### Лабораторная работа 3

#### РЕШЕНИЕ ЗАДАЧ С ИСПОЛЬЗОВАНИЕМ УЧЕБНОЙ ИПС

**З а д а ч а.** Найти все озера, расположенные в Евразии.

Обратимся к ИПС "Озера", на которую настроена ваша ЭВМ. Сразу отметим одно важное свойство этой ИПС. Можно сказать, что она "относится" к вам чрезвычайно заботливо. Назначение всех клавиш, которые вам потребуются при работе, указано на экране. Если вы случайно нажмете не ту клавишу, ЭВМ не станет вас ругать, а деликатно предоставит вам возможность осознать и исправить ошибку. Доверившись компьютеру, вы сможете самостоятельно, практически не прибегая к помощи учителя, решать задачи.

На экране дисплея напечатан перечень типов задач, которые можно решать с помощью ИПС:

**ЗАНЕСЕНИЕ ДАННЫХ  
ПРОСМОТР ИНФОРМАЦИИ  
СОТИРОВКА**

Наша задача относится ко второму типу. Стрелками "ВВЕРХ" и "ВНИЗ" подведите курсор ко второй строке и нажмите клавишу "ПЕРЕВОД СТРОКИ".

Руководствуясь указаниями ЭВМ, сформируйте запрос на поиск озер. Все клавиши, с помощью которых можно это сделать, указаны в нижней части экрана. Там же приведен образец запроса. Запрос готов? Нажав функциональную клавишу, вы получите на экране список интересующих вас озер. Кроме названий озер, ЭВМ сообщит вам площадь, глубину этих озер, а также название континента, на котором они расположены. С помощью функциональных клавиш вы можете тут же узнать, к примеру, общую площадь или максимальную глубину выбранных озер. Для этого надо подвести курсор в соответствующий столбец таблицы и нажать нужную функциональную клавишу.

Задача решена. Но, наверно, вас интересуют не только показанные компьютером признаки выбранных озер, но и, скажем, их происхождение. Чтобы получить более подробные сведения, нажмите клавишу "ВЫХОД" (на большинстве ЭВМ это клавиша <ESC>) и в таблице печатаемых признаков укажите интересующие вас признаки. После этого снова "попросите" ЭВМ напечатать список озер.

А теперь выполните задание 1, а—ж к § 3.

Разумеется, с течением времени данные об озерах могут меняться. Например, географическая экспедиция обнаружила сток у озера, ранее считавшегося бессточным, или ученые уточнили происхождение озера. Может измениться глубина озера и его площадь. (Вспомните, к примеру, судьбу Аральского моря!) Наконец, бывает и так, что меняется название страны, на территории которой расположено озеро.

Поэтому данные необходимо время от времени обновлять. Задачи обновления данных относятся к первому типу — "Занесение данных". Нажав несколько раз клавишу "ВЫХОД", добейтесь, чтобы на экране снова появились типы задач, решаемых с помощью ИПС, и выберите первый тип.

Выберите интересное вам озеро и, воспользовавшись каким-либо новейшим географическим справочником, выясните, какие изменения произошли с ним в последнее время. Внесите эти изменения в базу данных. Чтобы долго не искать нужное озеро в ИПС, поручите поиск компьютеру (для этого надо составить запрос на поиск).

Остался еще один тип задач, которые позволяет решать ИПС "Озера", — сортировка данных. Выберите этот тип задач и упорядочите озера по алфавиту, а затем по возрастанию площади.

#### § 4. РЕДАКТИРОВАНИЕ ТЕКСТОВ С ПОМОЩЬЮ ЭВМ

"Ванька Жуков... достал из хозяйского шкафа пузырек с чернилами, ручку с заржавленным пером и, разложив перед собой измятый лист бумаги, стал писать... Бумага лежала на скамье, а сам он стоял перед скамьей на коленях. "Милый дедушка, Константин Макарыч! — писал он. — И пишу тебе письмо..."

Вы, конечно, помните эту печальную историю, рассказанную А.П.Чеховым. Еще с пятого класса вы знаете, что в ней он отразил тяжелое положение крестьянства России после отмены крепостного права (см.: Книга для чтения. Часть 1. 5 класс). Но нас сейчас интересует совсем другое — технология написания писем. Эта технология почти не менялась в течение тысячелетий. Менялись, по сути дела,

только инструменты. Сначала надписи вырубали на камнях. Затем выдавливали стилем на глиняных дощечках. Писали палочками на папирусах и перьями на бумаге. Гусиные перья сменялись перьевыми ручками, те — авторучками, авторучки — пишущими машинками. Но не менялось главное: чтобы внести изменения в текст, его надо было заново переписывать (если, конечно, стремиться к аккуратности). Вы по себе знаете, сколько сил и времени отнимает переписывание даже обычного школьного сочинения. Может быть, поэтому многие ученики больше думают не о содержании сочинения, а о том, как бы не допустить помарки. А каково писателям! Л.Н.Толстой, например, переделывал "Войну и мир" более 8 раз, и каждый раз приходилось (целиком!) переписывать роман (правда, этим занималась главным образом жена Л.Н.Толстого, Софья Андреевна).

Появление компьютеров коренным образом изменило технологию письма. С помощью специальной программы, которая называется **редактором текстов**, на экране ЭВМ можно напечатать любой текст и внести в него (при необходимости) изменения. После этого текст можно записать на диск или магнитную ленту, где он будет храниться столько времени, сколько вы пожелаете. Если потребуется, его можно напечатать с помощью принтера.

При работе с редактором текстов роль бумаги играет экран ЭВМ, а роль карандаша и резинки играет **курсор** — небольшой прямоугольник, стоящий на экране там, где

*Братья Гримм писали  
перьями,  
Ильф и Петров —  
авторучками.  
Новое поколение  
выбирает компьютеры!*

должен будет появиться очередной символ. Курсор можно перемещать в любое место экрана. Текст можно раздвигать, вставляя новые слова. Можно стирать отдельные слова и целые строки, переставлять куски текста, автоматически заменять во всем тексте одно слово другим. Многие редакторы текстов "умеют" автоматически разбивать текст на страницы и нумеровать их. Они могут следить за размером полей, выравнивать текст по правому краю и т.д. Им можно поручить даже обнаружение и исправление орфографических ошибок!

С одним из редакторов текстов вы познакомитесь на лабораторной работе.

## **? Вопросы**

1. Что такое редактор текстов?
2. Какие изменения можно производить в тексте с помощью редактора текстов?



## Лабораторная работа 4

### УЧЕБНЫЙ РЕДАКТОР ТЕКСТОВ

Учебный редактор текстов уже введен в вашу ЭВМ. Перед вами на экране свободное поле, на котором вы будете писать текст. На экране присутствуют также значения функциональных клавиш. Эти значения — действия, которые редактор позволяет производить с текстом (добавить пустую строку, убрать строку, заменить одно слово другим и т. д.). Прежде всего вам надо освоить эти действия. Для этого проще всего сначала набрать какой-нибудь короткий текст или составить из каких-нибудь символов простую картинку, например такую, как на рисунке 1.

Затем измените текст (картинку), пользуясь функциональными клавишами, стрелками для перемещения курсора, а также клавишами "ВСТАВКА" и "УДАЛЕНИЕ СИМВОЛА" (например, "подстригите" человечка и заставьте его "улыбаться"). С помощью действия "Замена" можно быстро перерисовать ту же картинку, например, нуликами. Для этого нужно нажать функциональную клавишу, значение которой — действие "Замена". Редактор попросит сообщить ему сначала, что заменять (в нашем случае символ \*), затем на что заменять (символ 0). После этого редактор выяснит у вас, заменять ли все подряд или перед каждой заменой предварительно спрашивать разрешения. В данном случае нужно заменять все подряд. Впрочем, если хотите, можете заменить нулем, скажем, лишь каждую вторую звездочку.

Разминка закончена. Теперь пора применить полученные навыки. Мы уже говорили о том, как много времени

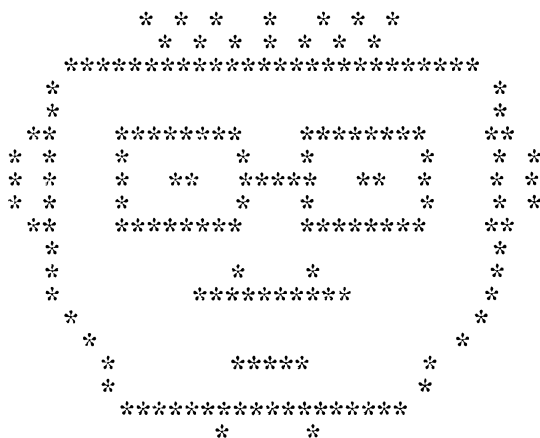


Рис. 1

отнимает у руководителей написание деловых писем. При этом часто приходится составлять много однотипных писем, отличающихся только отдельными словами (например, благодарственные письма директора школы родителям учеников). С помощью редактора текстов эту работу можно сильно облегчить.

Сейчас ассистент вызовет с диска на экран бланк благодарственного письма. Внесите в текст необходимые изменения и дополнения так, чтобы получилось письмо вашим родителям. Нажав соответствующую функциональную клавишу, вы сможете напечатать это письмо на бумаге.

Для решения следующей задачи потребуются действия поиска и замены. Перед вами на экране текст телеграммы. Каждый, кто получал телеграммы, знает, что в них вместо знаков препинания используются сокращения "тчк" (точка), "зпт" (запятая), "впр" (вопросительный знак), "вск" (восклицательный знак) и т. д. Вы должны изменить телеграфный текст так, чтобы в нем стояли обычные знаки препинания.

Конечно, несколько мешает чтению телеграмм и то, что в ней все слова напечатаны заглавными буквами. Если у вас осталось время, то преобразуйте получившийся текст так, чтобы он выглядел привычным образом.

## § 5. РЕДАКТИРОВАНИЕ ИЗОБРАЖЕНИЙ С ПОМОЩЬЮ ЭВМ

В предыдущем параграфе мы говорили о нелегкой доле Льва Толстого. Не менее тяжела была жизнь у Леонардо да Винчи. Он, в отличие от других художников Возрождения, создавал свои шедевры исключительно в одиночку. Сам перебирал тысячи вариантов расположения фигур, рисовал сотни эскизов, пока наконец не возникал тот идеальный вариант, который он переносил на холст, доводя каждую деталь картины до совершенства. В результате за всю свою долгую жизнь Леонардо создал всего двадцать живописных произведений. К тому же его жизнь постоянно отвлекали конфликты с влиятельными заказчиками, которым, естественно, не нравилась медлительность художника. Вспомним еще, что Леонардо да Винчи был не только художником, но и гениальным конструктором, чьи идеи намного опередили время. Ведь именно он придумал и подводную лодку, и летательный аппарат тяжелее воздуха. Как и каждому конструктору, ему приходилось делать множество разнообразных чертежей. А то, что черчение — дело нелегкое, вы, наверно, и сами знаете.

Был бы у Леонардо компьютер, сколько сил удалось бы сберечь для творчества! Компьютер помог бы ему создать эскиз картины: выбрать наилучший вариант расположения

фигур и соотношения их размеров на картине, подобрать цвета и т. д. Современный компьютер может сделать и точную репродукцию картины. И уж совершенно неоценима помощь компьютера при черчении.

Существует немало программ, называемых **системами автоматизированного проектирования (САПР)**, которые позволяют поручить компьютеру изготовление чертежей. Компьютер может рассчитать размеры деталей по заданным характеристикам (например, по известной нагрузке на колонну — ее толщину). Но главное достоинство любой САПР — наглядность. По трем проекциям детали компьютер легко создаст ее пространственное изображение. Более того, компьютер может "повернуть" деталь, давая возможность посмотреть на нее с разных сторон. Наглядность "общения" с САПР обеспечивается одной из важнейших ее частей — **графическим редактором**.

*Краски блекнут,  
холсты ветшают...  
Картины созданные  
на ЭВМ, —  
нетленны!*

Графические редакторы, предназначенные для профессиональных конструкторов, освоить, конечно, непросто. Поэтому на лабораторной работе мы познакомим вас с простым учебным графическим редактором. Он позволяет рисовать на экране ЭВМ фигуры, состоящие из отрезков, прямоугольников, окружностей и их частей, печатать изображение на бумаге и записывать его на диск.



## Лабораторная работа 5

### УЧЕБНЫЙ ГРАФИЧЕСКИЙ РЕДАКТОР

Сегодня с помощью ЭВМ вы будете рисовать. Компьютер, в который уже загружена программа "Графический редактор", ожидает ваших указаний.

Роль листа бумаги играет экран ЭВМ. Любое изображение на экране состоит из отдельных точек разных цветов. Расстояние между соседними точками настолько мало, что линии кажутся непрерывными.

Роль карандаша играет **графический курсор**. Он может выглядеть по-разному — в виде стрелки, крестика или точки. Курсор можно перемещать по экрану с помощью клавиш со стрелками. При этом в специально отведенном месте экрана вы увидите два изменяющихся числа. Это координаты курсора. Перемещая курсор, выясните, где находится начало системы координат и куда направлены оси.

Как и прежде, в специальной строке экрана записаны значения функциональных клавиш — действия, которые может выполнить графический редактор: нарисовать отрезок

зок, прямоугольник, окружность и т. д. Одно из этих действий "Скорость" — изменение скорости движения курсора. Значения функциональных клавиш можно менять, нажимая на клавишу "ВЫБОР".

Прежде чем рисовать тот или иной элемент фигуры, надо выбрать его цвет. Для этого имеется палитра цветов. Нажмите на функциональную клавишу "Цвет", стрелками "ВВЕРХ" и "ВНИЗ" подведите курсор к какому-нибудь цвету и снова нажмите на клавишу "Цвет".

А теперь расскажем, как изображать элементы фигур: отрезки, окружности и прямоугольники со сторонами, параллельными осям координат.

Чтобы нарисовать отрезок, надо поместить курсор в точку, где отрезок должен начинаться, и нажать на функциональную клавишу "Отрезок", затем переместить курсор в точку, где отрезок должен кончиться, и снова нажать на клавишу "Отрезок". Нарисуйте какой-нибудь отрезок.

Чтобы нарисовать прямоугольник со сторонами, параллельными осям координат, нужно сначала поместить курсор в точку, где будет располагаться одна из его вершин, и нажать на функциональную клавишу "Прямоугольник", затем переместить курсор к противоположной вершине и снова нажать на ту же клавишу. Нарисуйте какой-нибудь прямоугольник, можно другим цветом.

Наконец, чтобы нарисовать окружность, нужно, пользуясь соответствующей функциональной клавишей, отметить центр окружности, а затем какую-нибудь точку на ней. Нарисуйте несколько окружностей разных цветов. Отметим, что редактор позволяет рисовать только те окружности, которые целиком помещаются на экране.

Конечно, при рисовании сложных фигур немудрено ошибиться. Если какой-нибудь фрагмент рисунка кажется вам выполненным неудачно, его можно стереть. Для этого нужно указать на экране прямоугольник, внутри которого изображение должно быть уничтожено. Этот прямоугольник задается, как и раньше, двумя противоположными вершинами, но вместо клавиши "Прямоугольник" надо пользоваться клавишей "Черный квадрат" (клавиша названа в честь известной картины художника-авангардиста К. Малевича). Сотрите что-нибудь.

Попробуйте теперь нарисовать несколько простых картинок: домик с окнами, деревце с ветками, солнце с лучами и т. д. (например, рисунки 2 — 4). Ну как, получается? Наверно, не очень. Например, солнце у вас, скорее всего, получилось черным, словно при затмении. И вообще ваш рисунок напоминает книжку-раскраску: нарисованы только контуры фигур. Давайте раскрасим рисунок. Закрашивать

фигуру можно только тем цветом, которым нарисован ее контур. Поэтому, чтобы закрасить фигуру, надо клавишей "Цвет" выбрать цвет ее контура, а затем, поместив курсор внутрь фигуры, нажать на функциональную клавишу "Закрасить". Раскрасьте свой рисунок. Получилось?

Быть может, вас постигла неудача: при попытке закрасить фигуру с разноцветным контуром весь экран оказался "залит" одной краской. Кажется, что весь труд пропал. Нет, не все потеряно! Ведь вы имеете дело не с листом бумаги. Компьютер предусмотрительно запомнил предшествующую картину и готов воспроизвести ее по первому вашему требованию. Нажмите на клавишу "Откатка" (так принято называть возврат на один шаг) и можете продолжить рисование.

А теперь подумайте, как же закрасить фигуру с разноцветной границей.

Освоившись с графическим редактором, переходите к более творческим заданиям. Мы рассказывали, что с помощью компьютера можно создавать репродукции картин. Создайте несколько репродукций упомянутой выше картины Малевича "Черный квадрат" (рис. 5). Сравните получившиеся репродукции по силе эстетического воздействия.

Следующее задание — политико-географическое. Изобразите флаг своей республики, а также флаг какого-нибудь другого государства или, скажем, эмблему Общества Красного Креста и Красного Полумесяца (рис. 6).

И наконец, общественно-политическое задание. Нарисуйте какой-нибудь плакат на одну из тем: "Нет ...!", "Да...!", "Руки прочь от ...!" или "Требуем ...!". Вместо

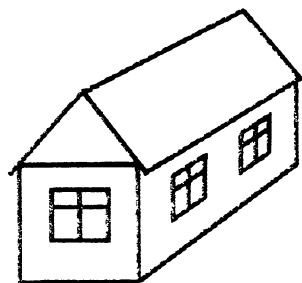


Рис. 2

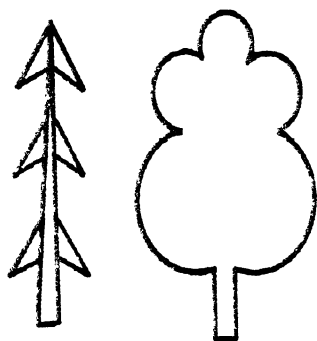


Рис. 3

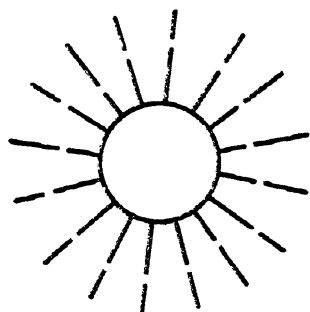


Рис. 4

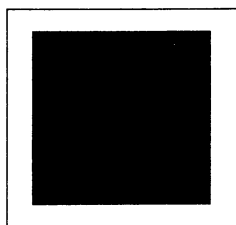


Рис. 5



Рис. 6

многоточий поставьте подходящие слова. Слова должны быть не очень длинные, ведь буквы придется рисовать отрезками.



#### КОНСПЕКТ ГЛАВЫ 1

**Информатика** изучает технологию сбора, хранения и переработки информации. Основной инструмент этой технологии — электронно-вычислительные машины (ЭВМ). Они предназначены для усиления умственных возможностей человека.

Главная способность ЭВМ — способность к имитации объектов, явлений, механизмов, даже таких, которые не существуют в природе. Эта способность в сочетании с быстродействием — до миллиардов операций в секунду — основа эффективности ЭВМ.

**Процессор** руководит всей работой ЭВМ, осуществляя связь между остальными частями ЭВМ, а также между ЭВМ и человеком. Процессор работает по специальным инструкциям-программам, состоящим из понятных ему команд. Для хранения информации предназначено особое устройство — **память**. Основной вид памяти — оперативная память — находится, как и процессор, внутри ЭВМ. Внешняя память — магнитные диски (дискеты) и магнитофонные кассеты — служит для хранения существенно большего количества информации.

К основным устройствам ввода-вывода относятся клавиатура, дисконд (магнитофон), дисплей, принтер. С их помощью ЭВМ обменивается информацией с внешним миром.

**Электронные таблицы** — специальные программы, предназначенные для организации большого объема вычислений.

Электронная таблица позволяет хранить в табличной форме исходные данные, результаты, а также связи (математические соотношения) между ними. Но главное — при изменении исходных данных все результаты автоматически пересчитываются и заносятся в таблицу.

**Информационно-поисковые системы (ИПС)** служат для переработки больших объемов информации, например для поиска нужных сведений. Крупные ИПС, предназначенные для коллективного использования, называются банками данных.

Каждая ИПС состоит из двух частей: большой специально организованной совокупности данных (она называется **базой данных**) и программы, позволяющей оперировать ими. Данные, хранящиеся в базе данных, — это значения заранее фиксированных признаков некоторого набора объектов. А "оперировать" — значит находить объекты по заданным признакам, изменять и дополнять сведения об объектах, а также решать некоторые другие задачи. Для того чтобы заставить ЭВМ найти интересующие вас сведения, нужно составить запрос. Правила записи запросов для каждой ИПС свои.

Если информационно-поисковые системы заменяют записные книжки и другие традиционные средства хранения информации, то **редакторы текстов** заменяют традиционные инструменты для написания текстов. С помощью редактора текстов на экране ЭВМ можно напечатать любой текст, внести в готовый текст изменения. Текст можно раздвигать, вставляя новые слова. Можно стирать отдельные слова и целые строки. Многие редакторы текстов "умеют" автоматически разбивать текст на страницы и нумеровать их; обнаруживать и исправлять орфографические ошибки. Можно поручить такому редактору следить за размером полей и т. д.

**Системы автоматизированного проектирования (САПР)** позволяют поручить компьютеру изготовление чертежей. Компьютер может рассчитать размеры деталей по заданным характеристикам. Но главное достоинство любой САПР — наглядность. По трем проекциям детали компьютер может создать ее пространственное изображение, может "повернуть" деталь, давая возможность посмотреть на нее с разных сторон. Такие возможности предоставляет одна из важнейших частей САПР — **графический редактор**.

## Глава 2

# ИСКУССТВО ПОСТРОЕНИЯ МОДЕЛЕЙ

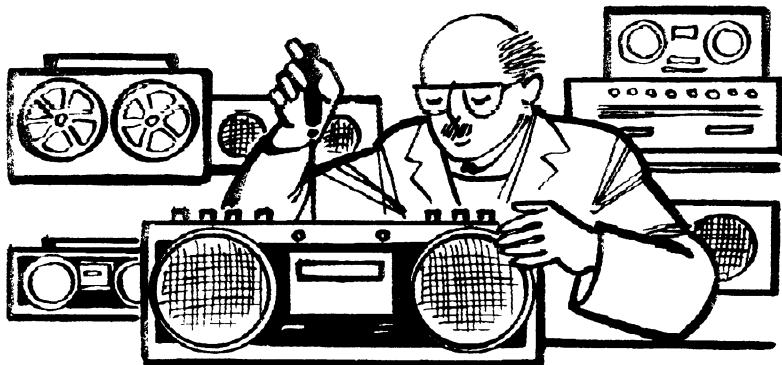
---

Пообщавшись с ЭВМ, вы, конечно, убедились в том, что она понимает лишь четкие недвусмысленные инструкции. Не скажешь ведь графическому редактору: "Нарисуй-ка мне какую-нибудь окружность". Обязательно надо указать центр и точку на окружности. Но инструкция по решению задачи может быть четкой, только если сама задача четко сформулирована. О том, что значит хорошо сформулировать задачу и каков дальнейший путь ее решения с помощью ЭВМ, рассказывается в этой главе.

### § 6. ЧТО ТАКОЕ ХОРОШО И ЧТО ТАКОЕ ПЛОХО ПОСТАВЛЕННАЯ ЗАДАЧА

Поработав летом, школьник купил себе дорогой магнитофон. К сожалению, с покупкой магнитофона затраты на него не кончаются: чем старше магнитофон, тем больше денег съест его ремонт. Со временем ремонт может обойтись дороже, чем замена старого магнитофона на новый магнитофон той же марки. Вот и возникла у школьника задача.

**Задача.** Через сколько лет после покупки магнитофона его наиболее выгодно обменять на новый магнитофон той же марки?



Вы, наверное, таких задач никогда в школе не решали. Ведь в ней неизвестно, что дано, нет никаких уравнений и неясно, откуда их извлечь. То ли дело привычные задачи о бригадах, работающих то вместе, то врозь, бассейнах с трубами и поездах, курсирующих между пунктами *A* и *B*. Там все ясно — что дано, что требуется получить, а уравнения, связывающие исходные данные и результаты, однозначно извлекаются из формулировок задач (хотя, быть может, не всегда просто это сделать). В жизни, однако, все сложнее и часто встречаются задачи, подобные задаче школьника-меломана.

*Про задачи, в которых неясно, что дано или что надо получить, либо нечетко определены связи между исходными данными и результатами,* говорят, что они **плохо поставлены**. Их решение надо начинать с четкой постановки. Отвлечемся на время (до следующего параграфа) от нашей "коммерческой" задачи и поговорим о постановках задач.

Нелегкое это дело правильно поставить задачу: не переупростить и не переусложнить, не упустить важных исходных данных и не включить лишних, соразмерить желаемое с возможным.

За примером далеко идти не надо. Допустим, что вы решили в порядке индивидуальной трудовой деятельности организовать бюро по обмену квартир. Ваша задача — для каждой поступившей заявки подобрать вариант обмена (быть может, не один), удовлетворяющий все заинтересованные стороны. Попробуйте четче сформулировать эту задачу. Сложно, не правда ли? Сразу возникает много вопросов: какие параметры имеющейся и требуемой жилплощади учитывать в заявке (количество и площадь комнат, их взаимное расположение, площадь кухни и других подсобных помещений, год постройки дома, этаж, наличие лифта, как часто он не работает и т. д.); принимать ли заявки на объединение и размен квартир; учитывать ли стоимость кооперативной квартиры?... А не отказаться ли сразу от такой индивидуальной деятельности?

Или, скажем, задача, которая упоминалась в § 2: составить расписание уроков в школе, удовлетворяющее и учителей, и школьников. Каковы исходные данные для этой задачи? Это зависит, конечно, от того, какое расписание считать хорошим. Можно задаться целью обеспечить лишь то, чтобы два разных урока не проходили в одно и то же время в одном классе. Тогда исходных данных будет немного: список учебных предметов да количество классов на каждой параллели. При этом, правда, может случиться так, что одному классу придется изучать только математику, другому — только географию, а учитель литературы

вряд ли будет доволен, если ему поручат вести уроки физики. Попробуйте сами четко сформулировать задачу составления школьного расписания. Думаем, что после этого вы будете с пониманием относиться к нелегкому труду завуча, который решает эту задачу каждые полгода, а то и чаще.

Этих примеров, наверно, достаточно, чтобы понять: постановка задачи — важная часть (иногда говорят — половина) ее решения. **Четко сформулировать (хорошо поставить) задачу** — это значит высказать те предположения, которые позволяют из всего многообразия информации об изучаемом явлении или объекте выделить исходные данные, определить, что будет служить результатом и какова связь между исходными данными и результатами. Все это — предположения, исходные данные, результаты и связи между ними — называется **моделью задачи**.

Далеко не всегда заранее ясно, какие свойства объекта окажутся существенными для решения задачи. Вот, к примеру, старинная

**Задача.** Двумя ударами топора разрубить подкову на наибольшее число частей, не перекладывая части после удара.

Приступим к постановке задачи. Сразу можно сформулировать предположения, без которых задачу решить невозможно: удар топора — прямолинейный разрез, части подковы после удара остаются на месте и не деформируются. Ясно также, что подкова — это некоторая дуга. Значит, надо нарисовать дугу и рассечь ее двумя прямыми. Таким образом, получается

Модель задачи о подкове.

**Дано:** половина окружности на плоскости.

**Требуется:** провести две прямые.

**Связь между исходными данными и результатами:** прямые надо провести так, чтобы они разбивали полуокружность на наибольшее число частей.

Мы четко поставили задачу, и нетрудно убедиться, что больше пяти частей не получится (рис. 7).

Однако если вы заглянете в сборники старинных задач (например, в книжку Е. И. Игнатьева "В царстве сме-



Рис. 7

калки”), вы увидите, что наши деды и прадеды умели разбивать подкову на 6 частей. Правда, в этих сборниках приведен другой рисунок (рис. 8). Оказывается, мы слишком упростили задачу и не учли, что у подковы есть еще и ширина. Итак, в нашей модели надо предположение, что подкова — дуга, заменить на такое: подкова — половина кольца.

Получается новая

Модель задачи о подкове.

**Дано:** половина кольца на плоскости.

**Требуется:** провести две прямые.

**Связь между исходными данными и результатами:** прямые надо провести так, чтобы они разбивали полукольцо на наибольшее число частей.

Поразмыслив, можно сообразить, что на результат может влиять еще и толщина подковы (рис. 9). На сколько частей можно разбить подкову, если учесть толщину (модель снова изменится)? У нас получилось 7 частей (а у вас?). Как видите, подкова не такой простой объект, как могло показаться.



Рис. 8

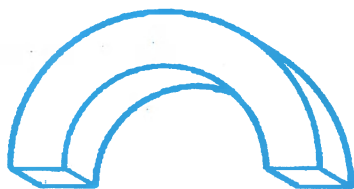


Рис. 9

А может быть, есть еще более точная модель задачи, в которой количество частей будет другим? Что бы еще учесть? Конечно же... дырки! Ведь у подковы есть дырки для гвоздей (рис. 10). В успех трудно поверить, но получается 11 частей!

Фактически нам удалось удвоить первоначальный результат за счет более полного описания объекта. Каждая из четырех моделей задачи о подкове — хорошо поставленная задача с однозначным ответом. Но четвертая модель ближе к реальности, чем остальные, и только поэтому ответ 11 ближе к ответу в исходной задаче, более правильный, чем 5, 6 или 7. Если же считать все четыре модели задачи о подкове самостоятельными задачами, то вопрос о том, какой ответ — 5, 6, 7 или 11 — более правильный, не имеет смысла. Это ответы к четырем разным задачам о дуге, о полукольце и т. д. Итак, можно сделать важный вывод: *если мы заменяем жизненную задачу ее моделью, то и ответ относится к модели. Он будет тем ближе к ответу в исходной задаче, чем ближе модель к реальности.*

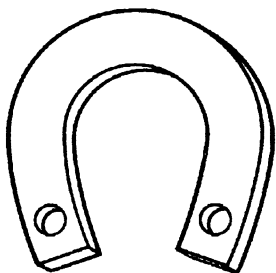


Рис. 10

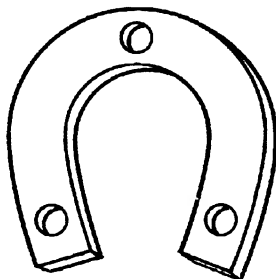


Рис. 11

Как видите, от исходных предположений существенно зависит решение задачи, и, чем больше свойств объекта мы учитываем в модели, тем сложнее решение. Однако не всегда, усложнив модель, мы получим более точный результат. Скажем, те из вас, кому довелось видеть подкову, знают, что у подковы не две дырки, а больше (рис. 11). Тем не менее число частей, на которые можно разрубить подкову, не зависит от количества дыр (если оно больше двух).

Вспомним, кстати, задачу о трех туристах. Там речь шла о справедливом разделе груза. Туристы (и мы вместе с ними) решили, что "справедливо" — значит "поровну". Принятие этой модели потребовало от вас больших усилий: нелегко разделить груз абсолютно точно. На самом деле в такой точности не было никакой необходимости. Ведь исходные данные, скажем, вес топора или палатки, известны лишь приближенно. Поэтому разумнее было принять такую модель: "раздел груза почти поровну (например, с точностью до полкилограмма)".

Искусство составления моделей как раз и заключается в том, чтобы, не переусложнив модель, учесть в ней все существенное и отбросить второстепенное. Не в этом ли состоит искусство вообще? Ведь, пожалуй, каждый вид искусства, будь то живопись, скульптура или кино, — это создание моделей жизненных явлений с использованием присущих ему выразительных средств.

## ? Вопросы

1. Что значит поставить задачу?
2. Что такое модель задачи?



## Задания для самостоятельного выполнения

1. Как разрубить подкову двумя ударами топора:
  - а) на 7 частей (рис. 9);
  - б) на 11 частей (рис. 10)?

2. Объясните, почему следующие задачи плохо поставлены. Составьте модели этих задач:

а) Семья, состоящая из дедки, бабки, внучки, Жучки и кошки, взяв в аренду колхозное поле, решила выращивать репу. Потребуется ли привлечение сезонного рабочего (мышки) для сбора урожая?

б) Винни Пух и Пятачок построили ловушку для Слонопотама. Удастся ли его поймать?

в) Винни Пух и Пятачок пошли в гости к Кролику. Сколько бутербродов с медом можно съесть Винни Пуху, чтобы не застрять в двери?

г) Малыш и Карлсон решили по-братски разделить два сладких орешка — большой и маленький. Как это сделать?

д) Аббат Фариа решил бежать из замка Иф. Сколько времени ему понадобится, чтобы осуществить свой замысел?

е) Однажды утром гражданин Н. проснулся от того, что на него с потолка упала капля воды. Сколько денег придется потратить гражданину Н. на ремонт квартиры?

ж) Приближается Новый год. Откуда лучше пригласить гостей на новогодний бал: из медицинского училища или из суворовского?

## § 7. МОДЕЛИ ЗАДАЧ И ИСПОЛНИТЕЛИ

В предыдущем параграфе мы с вами пришли к выводу, что четкая постановка любой жизненной задачи — это первый шаг на пути построения ее модели. Конечно, если браться за решение задачи не подумав, то никакая модель не нужна. Только решение в этом случае будет похоже вот на что.

Вы приехали в Москву (для москвичей — в Санкт-Петербург), и вам надо добраться к другу, который переехал недавно в совершенно незнакомое для вас место. Выйдя из здания вокзала, вы, не думая (это самое главное предположение в нашем мысленном эксперименте!), садитесь в транспорт — трамвай, троллейбус, автобус или метро. Все так же не думая, проезжаете несколько остановок, выходите, садитесь на другой вид городского транспорта или идете пешком и т. д. При таких ваших действиях очень мало шансов, что друг вас дождет.

Мы, однако, хотим заострить ваше внимание не на прописных истинах типа "Сначала думай, потом делай", "Семь раз примерь, потом отрежь" и т. п., а на том, как вы будете решать свою "жизненную" задачу.

Самый простой путь — попросить друга встретить вас на вокзале. Вам больше думать ни о чем не надо, друг сам

*Составляйте  
компьютерные  
модели —  
они сохраняют лучшие  
свойства объектов!*

предложит нужный маршрут. Что ж, такой метод воспользоваться чьим-то готовым решением достаточно распространен: от кулинарии до высших политических сфер. И в этом нет ничего зазорного, ведь потому и

бесценен опыт человечества, что каждому не нужно заново, с нуля решать все жизненные задачи.

Вернемся к проблеме поиска маршрута, ведущего к дому вашего приятеля. Если он не может вас встретить или вы хотите своим приездом сделать ему сюрприз, то решать эту жизненную задачу придется самому.

Самое простое — это взять транспортную схему города (т. е. план города с нанесенными на него линиями движения транспорта) и прокладывать маршрут по этой схеме. Теперь каждому ясно, что произошло: задача нахождения пути к дому приятеля заменена моделью найти маршрут на транспортной схеме.

Да и если вы знаете, как добираться к другу, то все равно решение этой задачи присутствует в вашем сознании не в виде реальных улиц, трамвайных или троллейбусных вагонов, а в виде некоторого представления о том, какими улицами идти и номерами каких транспортных маршрутов вам необходимо воспользоваться. Иными словами, в своей памяти вы храните решение, но не исходной жизненной задачи, а... ее модели!

Вообще, какую бы жизненную задачу ни взялся решать человек, первым делом он строит ее модель осознанно, а иногда и нет. Ведь бывает и так: вы напряженно ищете выход из трудной ситуации, пытаетесь нащупать, за что можно ухватиться. И вдруг находит озарение... Что же произошло? Это сработало замечательное свойство нашего разума — умение по какому-то волшебству, безотчетно уловить самое важное (иногда мгновенно), превратить "информационный хаос" в стройную модель стоящей перед человеком задачи.

Как видите, с моделями задач вы имеете дело ежедневно, ежечасно и, может быть, даже ежеминутно. Вы никогда об этом не задумывались, поскольку построение моделей для человека так же естественно, как ходьба или умение пользоваться ножом и вилкой. Разница же в том, что ходить, пользоваться ножом и вилкой вас довольно долго учили, и теперь вы хорошо умеете это делать. А вот строить модели вы, скорее всего, учились стихийно, сами того не подозревая.

Потому и получаются у вас иногда удачные решения "жизненных" задач, а иногда не очень, ведь, как вы видели

в предыдущем параграфе, разные модели приводят к различным результатам при решении одной и той же жизненной задачи.

А того, кто будет получать результаты из исходных данных, используя построенную модель, договоримся называть **исполнителем**. Разумеется, модель должна учитывать возможности исполнителя. Скажем, первоклассника, не умеющего пользоваться схемой транспортных маршрутов, не стоит в одиночку отпускать в путешествие по незнакомому городу, вооружив его только этой схемой.

Иными словами, приступая к созданию модели, нужно сразу сориентироваться на исполнителя, которому предстоит с ней работать. А значит, высказывая предположения, на которых будет основываться построение модели, не обойтись без информации о возможностях исполнителя. Как составлять "досье" на исполнителя, мы расскажем в следующей главе.

В нашем курсе информатики вы будете учиться решать задачи с помощью тех исполнителей, которые "живут" в ЭВМ. С некоторыми вы уже знакомы — это текстовый и графический редакторы, электронная таблица, информационно-поисковая система "Озера". Было у вас и непродолжительное знакомство с исполнителем **ВЫЧИСЛИТЕЛЬ**. Позже вы с ним будете работать бок о бок.

Как войти в контакт с компьютерным исполнителем? Старику из сказки А. С. Пушкина нужно было лишь иногда "кликать золотую рыбку". Аладдин вызывал исполнителя, потерев волшебную лампу. Чтобы вызвать исполнителя в оперативную память компьютера, его вовсе не обязательно тереть, достаточно вставить дискету и выбрать нужную программу.

*Модель задачи, составленную в расчете на исполнителя, имитированного на ЭВМ, будем называть **компьютерной моделью**.* Это означает, что исходные данные, результаты и связи между исходными данными и результатами представлены в виде, "понятном" компьютерному исполнителю.

Скажем, первые две модели задачи о подкове легко "объяснить" одному из компьютерных исполнителей — графическому редактору.

Если же исполнитель "умеет" только вычислять, то, высказывая предположения, нужно позаботиться о том, чтобы исходные данные и результаты были числами, а связи между ними — математическими соотношениями. После такого "перевода" задачи на язык математики получается модель, которую обычно называют **математической моделью**.

Вернемся к нашей коммерческой задаче. Так когда же наиболее выгодно заменить старый магнитофон новым? Мы

уже говорили, что надо начать с построения модели задачи. Желательно, конечно, построить такую модель, чтобы потом для решения задачи можно было воспользоваться ЭВМ. Иначе говоря, желательно, чтобы это была компьютерная модель. Поскольку во всякой ЭВМ "живет" исполнитель (и не один), умеющий работать с числами, будем строить математическую модель.

Прежде всего конкретизируем ситуацию (уточняющие предположения будем нумеровать).

С течением времени стоимость магнитофона убывает, а стоимость ремонта возрастает. Как же выразить этот факт математически? Поговорив со знакомым товароведом комиссионного магазина, мы поняли: 1) можно считать, что цена магнитофона убывает каждый год на один и тот же процент.

Теперь о ремонте. О его стоимости можно поинтересоваться в ремонтной мастерской. Конечно, там не скажут, сколько будет стоить ремонт именно вашего магнитофона через год, два или три. Однако: 2) средняя стоимость ремонта магнитофона в зависимости от его "возраста" и типа известна достаточно хорошо.

Естественно предполагать, что: 3) ремонт любого магнитофона данного типа не будет обходиться дороже или дешевле.

Знакомый директор радиомастерской сообщил нам, сколько в среднем стоит ремонт магнитофона не "старше" 5 лет. Про более старые магнитофоны он откровенно сказал, что их чинить бесполезно, дешевле выбросить. После этих слов мы поняли, что: 4) магнитофон невыгодно менять позже чем через 5 лет после покупки.

Осталось осознать смысл слов "наиболее выгодно" из формулировки задачи. "Наиболее выгодно заменить магнитофон" означает "сделать минимальными расходы на обмен и ремонт магнитофона". Как же подсчитать эти расходы? Для простоты предположим, что: 5) школьник собирается слушать магнитофон 5 лет и при этом хочет сменить его всего один раз. Тогда подсчет можно произвести следующим образом.

Обозначим начальную стоимость магнитофона буквой  $m$ , стоимости ремонта магнитофона через 1, 2, 3, 4, 5 лет буквами  $a, b, c, d, e$ . Процент уценки обозначим через  $k$ . Допустим для примера, что школьник решил поменять магнитофон через 3 года после покупки. Тогда стоимость ремонта за 5 лет можно подсчитать так: стоимость ремонта старого магнитофона за 3 года плюс стоимость ремонта нового магнитофона за 2 года. Иначе говоря, стоимость ремонта магнитофона будет равна  $a+b+c+a+b$ . Через 3 года школьник сможет продать магнитофон за  $m(k/100)^3$  рублей. Зна-

чит, обмен магнитофона обойдется ему в  $m - m(k/100)^3$  рублей. Стало быть, общие расходы школьника за 5 лет будут равны  $a + b + c + a + b + m - m(k/100)^3$ . Рассуждая точно так же, можно понять, что если школьник продаст магнитофон через 1 год после покупки, то расходы будут равны  $a + a + b + c + d + m - m(k/100)$ ; через 2 года  $a + b + a + b + c + m - m(k/100)^2$ ; через 4 года  $a + b + c + d + a + m - m(k/100)^4$ ; через 5 лет  $a + b + c + d + e + m - m(k/100)^5$ .

Кажется, ситуация полностью прояснилась, и мы можем сформулировать модель коммерческой задачи.

Модель коммерческой задачи.

Дано: первоначальная стоимость магнитофона ( $m$  рублей), коэффициент ежегодной уценки ( $k$  процентов), стоимости ремонта магнитофона через 1, 2, 3, 4 и 5 лет после покупки ( $a, b, c, d, e$  рублей).

Требуется определить, через какое время после покупки ( $t$  лет) надо заменить магнитофон.

Связь между исходными данными и результатом:  $t$  равно номеру минимального из пяти чисел:

$a + a + b + c + d + m - m(k/100)$  (расходы в случае замены магнитофона через 1 год);

$a + b + a + b + c + m - m(k/100)^2$  (расходы в случае замены магнитофона через 2 года);

$a + b + c + a + b + m - m(k/100)^3$  (расходы в случае замены магнитофона через 3 года);

$a + b + c + d + a + m - m(k/100)^4$  (расходы в случае замены магнитофона через 4 года);

$a + b + c + d + e + m - m(k/100)^5$  (расходы в случае замены магнитофона через 5 лет).

Как видите, мы полностью перевели нашу задачу на язык чисел. Теперь ее сможет решить не только компьютер, но даже и любой шестиклассник. Хотя, конечно же, ЭВМ сделает это более быстро и надежно. О том, как решать задачу на ЭВМ, располагая компьютерной моделью, рассказывается в следующем параграфе.

## ? Вопросы

1. Влияет ли выбор исполнителя на построение модели? Как именно?

2. Что такое компьютерная модель задачи?

3. Какую модель называют математической?



## Задания для самостоятельного выполнения

1. Укажите, какие модели вы обычно используете для решения таких задач:

а) купить билет в кино;

б) скомплектовать волейбольную команду;

в) испечь торт.

Приведите свои примеры жизненных задач и моделей, используемых для их решения.

2°. В комнате, имеющей двери и окна, будет проведен ремонт. Для расчета потребности в строительных материалах сделаны следующие предположения:

пол, потолок, стены и окна имеют форму прямоугольников;

дверь имеет форму прямоугольного параллелепипеда;

батареи центрального отопления, расположенные под окнами, имеют прямоугольную форму.

Выберите из этих предположений те, которые существенны для решения каждой из следующих задач:

а) рассчитать количество мела, необходимое для побелки стен и потолка комнаты;

б) рассчитать количество краски, необходимое для покраски пола;

в) рассчитать количество обоев, необходимое для оклейки стен;

г) рассчитать количество древесины, необходимое для изготовления двери.

3. Выскажите предположения, существенные для решения следующей задачи:

Участок цеха по производству туристского снаряжения выпускает брезентовые палатки. Требуется определить количество брезента, нужное для выполнения участком месячного плана.

4. По заказу Управления культуры была изготовлена бронзовая статуя девушки с веслом.

Определите те свойства статуи, которые существенны для решения каждой из следующих задач:

а) перевезти статую из мастерской в городской парк;

б) установить статую на площадке парка;

в) увеличить посещаемость городского парка;

г) продать статую с аукциона.

5°. Для каждой из задач 2, а — г определите исходные данные и результаты. (Обратите внимание, что задачи 2, а, 2, б, 2, г недоопределены, т.е. перечислены не все сведения, существенные для решения этих задач.)

6. Завершите построение моделей, начатое при решении задач 2 и 5.

7°. Выскажите предположения, существенные для решения следующей задачи. Определите, что будет служить исходными данными, а что результатом.

Во время ремонта корабля потребовалось заделать пробоину в обшивке. Имеется лист стали. Удастся ли с его помощью заделать пробоину?

8. Завершите составление модели, начатое при решении задачи 7.

9\*. Через иллюминатор затонувшего корабля требуется вытащить сундук с драгоценностями. Удастся ли это сделать? Постройте модель для решения этой задачи.

## § 8. МОДЕЛЬ ПОСТРОЕНА, ЧТО ДАЛЬШЕ?

Созданием компьютерной модели завершается первый этап решения задачи с помощью ЭВМ. Для того чтобы исполнитель, имитированный на ЭВМ, выдал результат, нужно составить для него четкую инструкцию, строго указать необходимую последовательность действий. Такая инструкция называется **алгоритмом** решения задачи. Составление алгоритма — второй этап решения задач с помощью ЭВМ. О том, как составлять алгоритмы, мы расскажем в последующих главах.

Многие компьютерные исполнители могут выполнять алгоритмы без участия человека, автоматически. Но для этого алгоритм должен быть записан в "понятной" для исполнителя форме, т. е. на специальном языке, который называется **языком программирования**. Алгоритм, записанный на языке программирования, называется **программой** (в главе 8 описывается один из языков программирования Бейсик).

Получив с помощью ЭВМ результаты, необходимо их проанализировать. Ведь сколько бы свойств объекта мы ни учитывали, модель всегда основана на некотором упрощении и трудно быть абсолютно уверенным, что модель соответствует реальному объекту.

Такую уверенность можно обрести, лишь сопоставив результаты с экспериментальными фактами, теоретическими воззрениями и другой информацией об изучаемом объекте.

При этом может возникнуть необходимость уточнить компьютерную модель, поскольку выяснится что при ее разработке не были учтены какие-либо существенные свойства объекта (позже вы неоднократно столкнетесь с такой ситуацией).

Уточнив модель, снова составляют алгоритм, получают и анализируют результаты. И опять может потребоваться уточнение модели... Так будет продолжаться до тех пор, пока анализ результатов не покажет их приемлемое соответствие изучаемому объекту. После этого изучение объекта можно заменить исследованием компьютерной модели. Как изменяются результаты при изменении исходных данных? Каковы границы применимости модели? Эти вопросы возникают при решении любой "жизненной" задачи. Располагая компьютерной моделью, можно ответить на них, не проводя натурных экспериментов. В этом и состоит важ-

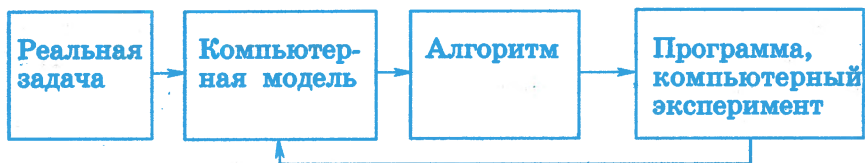
*Пользуйтесь  
компьютерным  
экспериментом —  
он не загрязняет  
окружающую среду!*

нейшее преимущество ЭВМ: натурный эксперимент, зачастую дорогостоящий или опасный, заменяется на **компьютерный эксперимент**. Наглядный пример: за несколько десятков часов работы на ЭВМ удалось достаточно

точно предсказать последствия ядерной войны. Как вы сами понимаете, натурный эксперимент обошелся бы человечеству слишком дорого.

Таким образом, в третий этап решения задачи с помощью ЭВМ, помимо написания программы, входит получение и анализ результатов — компьютерный эксперимент.

Мы теперь можем нарисовать общую схему решения задачи с помощью ЭВМ:



Эту схему мы в дальнейшем проследим на многих примерах. Построению компьютерных моделей и созданию алгоритмов будут посвящены теоретические занятия. Выполняя лабораторные работы, вы научитесь составлять программы и анализировать результаты работы ЭВМ.

## ? Вопросы

1. Каковы этапы решения задач с помощью ЭВМ?
2. Что такое программа?
3. С какой целью проводят анализ результатов работы ЭВМ?



## КОНСПЕКТ ГЛАВЫ 2

**Жизненные задачи** обычно не являются четко сформулированными. Поэтому прежде чем обратиться к ЭВМ для решения задачи, задачу нужно **четко сформулировать**, т. е. высказать те предположения, которые позволят указать исходные данные, результаты и связи между исходными данными и результатами. Предположения, исходные данные, результаты и связи образуют **модель задачи**.

Для одной и той же задачи могут быть созданы разные модели в зависимости от того, какие средства используются для их создания и какие предположения положены в их основу.

Если задача заменена ее моделью, то и ответ относится к модели и лишь опосредованно к исходной задаче.

Выбор исходных данных, описание результатов и соотношений в модели зависят также от возможностей исполнителя — того, кто будет ее решать. **Компьютерная модель задачи** — это модель, в которой исходные данные, результаты и связи между ними представлены в виде, "понятном" исполнителю, имитированному на ЭВМ.

Часто для решения задач используют **математические модели**, в которых исходные данные и результаты представлены числами, а связи между ними — математическими соотношениями. Удобство применения математических моделей состоит в том, что они обычно легче других преобразуются в компьютерные модели.

Созданием компьютерной модели завершается **первый этап решения задачи с помощью компьютера**. **Второй этап** — составление **алгоритма** (четкой инструкции, указывающей необходимую последовательность действий).

Многие компьютерные исполнители могут выполнять алгоритмы без участия человека, автоматически. Но для этого нужно составить **программу**, т. е. записать алгоритм на одном из языков программирования.

**Третий этап решения задачи с помощью ЭВМ — компьютерный эксперимент**. Он состоит в получении результатов работы ЭВМ, сопоставлении их с экспериментальными фактами, теоретическими воззрениями и другой информацией. При этом может возникнуть необходимость уточнить модель, полнее учитывая особенности изучаемого объекта. Уточнив модель, снова составляют алгоритм, проводят компьютерный эксперимент и так до тех пор, пока анализ результатов не покажет их приемлемое соответствие изучаемому объекту.

## Глава 3

# АЛГОРИТМ И ЕГО СВОЙСТВА

---

Когда компьютерная модель построена, надо приступить к следующему этапу решения задач на ЭВМ — составлению алгоритма. Что такое алгоритмы, какими общими свойствами они обладают и как они исполняются — обо всем этом рассказывается в данной главе.

### § 9. ПОНЯТИЕ АЛГОРИТМА

Каждый из нас ежедневно использует различные алгоритмы: инструкции, правила, рецепты и т. п. Обычно мы это делаем не задумываясь. Например, открывая дверь ключом, никто не размышляет над тем, в какой последовательности выполнять действия. Однако чтобы кого-нибудь (скажем, младшего брата) научить отпирать дверь, придется четко указать и сами действия, и порядок их выполнения. Например, так:

Достать ключ.

Вставить ключ в замочную скважину.

Повернуть ключ 2 раза против часовой стрелки.

Вынуть ключ.

А теперь представьте себе, что вы уговорили старшего брата разрешить вам прокатиться на его новеньком, только что из магазина мотоцикле. Инструкция, которую вы получите от брата, может быть такой:

Залить бензин в топливный бак.

Завести мотоцикл.

Проехать 3 раза вокруг дома.

Остановиться.

Заглушить мотор.

Посмотрим на эти алгоритмы. На первый взгляд между ними нет ничего общего. Одно дело — отпирать дверь, другое — кататься на мотоцикле. Однако если приглядеться внимательно, можно заметить существенное сходство между ними. Прежде всего, это строгий порядок выполнения дей-

ствий. Давайте переставим в первом алгоритме второе и третье действия:

Достать ключ.

Повернуть ключ 2 раза против часовой стрелки.

Вставить ключ в замочную скважину.

Вынуть ключ.

Вы, конечно, сможете исполнить и этот алгоритм. Только поставленной цели вряд ли достигнете — дверь-то не откроется!

А что произойдет, если второе и третье действия поменять местами во втором алгоритме? Он станет неисполнимым!

Итак, мы убедились: *для алгоритма важен не только набор действий, но и то, как они организованы*, т. е. в каком порядке выполняются.

Мы можем теперь сказать, что алгоритм — это организованная последовательность действий. Данную формулировку, конечно, нельзя считать определением алгоритма. Например, мы не объяснили, что означают слова "организованная" и "действия". Скажем сразу: абсолютно строгого определения алгоритма мы не дадим. Понятие алгоритма в информатике является фундаментальным. Таким же, какими являются понятия точки, прямой и плоскости в геометрии, пространства и времени в физике, вещества в химии и т. д. Поэтому мы не будем стремиться дать всеобъемлющее определение алгоритма, а будем уточнять смысл этого понятия в последующих параграфах.

Нам предстоит еще неоднократно записывать алгоритмы, поэтому давайте договоримся, выделяя порядок действий в алгоритме, записывать действия в столбик (как и в приведенных примерах).

## ? Вопросы

1. Почему нельзя дать строгого определения алгоритма?
2. Какое общее свойство алгоритмов вы можете назвать?
3. Какие нежелательные последствия могут возникать из-за ошибок в организации действий в алгоритме?

## Задания для самостоятельного выполнения

1. Приведите примеры неопределяемых понятий в следующих школьных предметах:
  - а) литература;
  - б) русский язык;
  - в) история;
  - г) биология.

2. Злоумышленник выдал следующий алгоритм за алгоритм получения кипятка:

Налить в чайник воду.  
Открыть кран газовой горелки.  
Поставить чайник на плиту.  
Ждать, пока вода не закипит.  
Поднести спичку к горелке.  
Зажечь спичку.  
Выключить газ.

Исправьте алгоритм, чтобы предотвратить несчастный случай.

3. Имеются цинк, 96%-ная серная кислота, вода, а также колба и пробирка. Исправьте ошибки в алгоритме получения водорода:

Поставить колбу на стол.  
Налить в колбу кислоту.  
Налить в колбу воду.  
Собрать выделяющийся газ в пробирку.  
Бросить в колбу цинк.

4°. Какие действия вы бы добавили, чтобы был выполнен следующий алгоритм переправы через Волгу в районе г. Саратова:

Подойти к реке.  
Войти в реку.  
Идти по дну, пока не выйдешь на другой берег.

5. Пусть дан отрезок  $AB$ . Определить, для решения какой задачи предназначен следующий алгоритм:

Поставить ножку циркуля в точку  $A$ .  
Установить раствор циркуля равным длине отрезка  $AB$ .  
Провести окружность.  
Поставить ножку циркуля в точку  $B$ .  
Провести окружность.  
Провести прямую через точки пересечения окружностей.

6. Что будет нарисовано на экране ЭВМ после выполнения следующего алгоритма работы с известной вам программой "Графический редактор" (см. лабораторную работу 5):

Нажать на функциональную клавишу "Отрезок".  
Нажать на клавишу "Стрелка вправо".  
Нажать на клавишу "Стрелка вправо".  
Нажать на функциональную клавишу "Отрезок".  
Нажать на функциональную клавишу "Отрезок".  
Нажать на клавишу "Стрелка вниз".  
Нажать на клавишу "Стрелка вниз".

Нажать на функциональную клавишу "Отрезок".  
Нажать на функциональную клавишу "Отрезок".  
Нажать на клавишу "Стрелка влево".  
Нажать на клавишу "Стрелка влево".  
Нажать на функциональную клавишу "Отрезок".  
Нажать на функциональную клавишу "Отрезок".  
Нажать на клавишу "Стрелка вверх".  
Нажать на клавишу "Стрелка вверх".  
Нажать на функциональную клавишу "Отрезок".

7. Дана фраза на английском языке: "Mike goes to school". Составьте алгоритм перехода от утвердительной формы к вопросительной: "Does Mike go to school?"

8. Работая с известной вам программой "Редактор текстов" (см. лабораторную работу 4), школьник напечатал на экране ЭВМ текст "Игра закончилась, а гол так и не был забит". Подошедший злоумышленник выполнил следующий алгоритм:

Нажать на функциональную клавишу "Замена".  
Напечатать букву "г".  
Нажать на клавишу "ПЕРЕВОД СТРОКИ".  
Напечатать букву "к".  
Нажать на клавишу "ПЕРЕВОД СТРОКИ".  
Напечатать букву "н".

а) Какая фраза теперь написана на экране?

б)\* С помощью какого алгоритма можно восстановить первоначальную фразу?

9. Составьте алгоритм построения биссектрисы угла с помощью графического редактора.

10. Даны число  $x$  и набор действий: разделить полученное число на 3; умножить  $x$  на 2; сообщить результат; прибавить к полученному числу 4; вычесть из полученного числа 7.

Составьте из этих действий два различных алгоритма. Любой ли алгоритм, составленный из этих действий, можно выполнить? Укажите две различные функции от  $x$ , значения которых вычисляются с помощью алгоритмов, использующих указанные действия.

11. Имеются два кувшина емкостью 3 л и 8 л. Напишите алгоритм, выполняя который можно набрать из реки 7 л воды (разрешается пользоваться только этими кувшинами).

12. (Старинная задача.) Некий человек должен перевезти в лодке через реку волка, козу и капусту. Каждый раз он может перевезти только либо волка, либо козу, либо капусту. На одном берегу нельзя оставить вместе козу и волка, а также козу и капусту (переправа капусты в же-

лудке у козы и козы в желудке у волка не разрешается). Составьте алгоритм переправы на другой берег.

13°. Разведывательный дозор в составе двух человек подошел к реке. Мост был разрушен, а река слишком глубока и широка, чтобы переправиться через нее вброд или вплавь. К счастью, около берега в маленькой лодке проплывали два мальчика. Как переправиться на этой лодке через реку, если она может выдержать только либо одного взрослого, либо двух мальчиков?

14. На полустанке одноколейной железной дороги остановился поезд в составе тепловоза и пяти вагонов, доставивший бригаду рабочих для строительства новой ветки. Пока на этом полустанке имеется только небольшой тупик, в котором в случае необходимости может поместиться тепловоз с двумя вагонами или три вагона. Вскоре следом за поездом со строительной бригадой к этому же полустанку подошел пассажирский поезд. Составьте алгоритм, позволяющий пропустить пассажирский поезд.

15. В сказке Г.-Х. Андерсена "Волшебное огниво" колдунья предлагает солдату следующий алгоритм добывания огнива из подземелья:

Войти в первую комнату.  
Поймать собаку (которая бросилась на солдата).  
Посадить ее на платок (собака сразу прismsиреет).  
Взять, что пожелаешь.  
Войти во вторую комнату.  
Поймать собаку.  
Посадить ее на платок.  
Взять, что пожелаешь.  
Войти в третью комнату.  
Поймать собаку.  
Посадить ее на платок.  
Взять огниво.  
Взять, что еще пожелаешь.  
Выйти.  
Отдать огниво.

Переставьте какие-либо действия в этом алгоритме и проверьте, исполнимый ли алгоритм получился в результате такой перестановки.

## § 10. ИСПОЛНИТЕЛИ АЛГОРИТМОВ

В предыдущем параграфе мы составили несколько алгоритмов и каждый раз предполагали, что исполнять наши алгоритмы будет человек. При этом оказалось, что не все они выполнимы. К счастью, человек далеко не единственный исполнитель алгоритмов. Роботы-манипуляторы и

станки с программным управлением, живая клетка и даже животные в ширке исполняют различные алгоритмы, в том числе и те, которые человек выполнить не в силах. Например, алгоритм перетравы через Волгу из задачи 4 к предыдущему параграфу человек исполнить не сможет, но его легко выполнит робот-"подводник". Значит, выполнимость алгоритма зависит от того, какие действия может совершать исполнитель.

Что же такое **исполнитель**? Исполнителя можно представлять себе как *некоторое устройство управления, соединенное с набором инструментов. Устройство управления понимает алгоритмы и организует их выполнение*, командуя соответствующими инструментами. А **инструменты производят действия, выполняя команды управляющего устройства**. Скажем, если человека рассматривать как исполнителя алгоритмов, то мозг — его управляющее устройство, а инструменты — руки, ноги, глаза, нос, рот, уши, ... (продолжите список самостоятельно). У роботов-манипуляторов, станков с программным управлением и ЭВМ управляющее устройство — процессор; что же касается набора инструментов, то он зависит от того, для решения каких задач предназначен тот или иной исполнитель.

Ясно, что, как бы ни были разнообразны возможности исполнителя, они всегда ограничены. Иначе для решения любой задачи годился бы один-единственный алгоритм:

**Получить исходные данные.**

**Найти решение.**

**Сообщить ответ.**

Поэтому, прежде чем составлять алгоритм решения задачи, нужно узнать, какие действия предполагаемый исполнитель может выполнить. Эти действия называются **допустимыми действиями исполнителя**. При составлении алгоритмов только их и можно использовать. Список допустимых действий исполнителя и есть то самое "досье", о котором мы упомянули в § 7.

**Исполнитель!**  
**Не допускай**  
**недопустимых**  
**действий!**

Поясним сказанное на простейшем примере. Допустим, требуется изобразить с помощью графического редактора, скажем, рисунок 12.

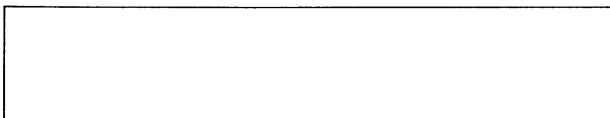


Рис. 12

Для того, кто умеет с помощью графического редактора рисовать прямоугольники, алгоритм будет состоять из одного действия:

Нарисовать прямоугольник.

Тот, кто успел изучить лишь рисование отрезков, должен будет воспользоваться более длинным алгоритмом:

Нарисовать горизонтальный отрезок.

Нарисовать вертикальный отрезок, начало которого совпадает с концом предыдущего отрезка.

Нарисовать горизонтальный отрезок, начало которого совпадает с концом предыдущего отрезка, а длина равна длине первого отрезка.

Нарисовать вертикальный отрезок, начало которого совпадает с концом предыдущего отрезка, а конец — с началом первого отрезка.

Можно составить алгоритм даже для того, кто впервые подошел к ЭВМ. Этот алгоритм будет очень длинным и сложным (нужно будет указать все клавиши, на которые надо нажать, — одну за другой). Как видите, чем меньше запас умений, тем более подробный алгоритм приходится составлять.

В хорошо поставленной задаче набор допустимых действий (т. е. действий, которыми можно пользоваться при решении задачи) в явной или неявной форме указывается в формулировке задачи. Например, в школьных задачах на построение обычно подразумевается, что можно пользоваться только циркулем и линейкой. А иногда ставится задача на построение только циркулем или только линейкой. Таким образом, можно сделать вывод, что различные классы задач требуют различных наборов допустимых действий. Поэтому, *чтобы можно было применить ЭВМ для решения того или иного класса задач, необходимо "научить" ЭВМ имитировать нужный набор допустимых действий* (можно назвать это **имитацией исполнителя с помощью ЭВМ**).

Задачи на вычисления — один из наиболее широких классов задач, решаемых на ЭВМ. Исполнитель, который будет выполнять вычислительные алгоритмы, вам уже знаком — это **ВЫЧИСЛИТЕЛЬ**. Он имеет дело с числами и переменными, обозначающими числа. На первых порах мы ограничимся следующим набором его допустимых действий:

1. Запросить исходные данные и обозначить их буквами. Соответствующую команду можно записывать, например, так: "Запросить...", указывая вместо многоточия буквы, обозначающие соответствующие данные. Например:

**Запросить А.**  
**Запросить А, В и С.**

2. Сообщить результаты вычислений или какой-нибудь текст: "Сообщить..." Здесь вместо многоточия ставятся буквы или алгебраические выражения либо текст, заключенный в кавычки. Выполняя это действие, ВЬЧИСЛИТЕЛЬ печатает значения указанных выражений или текст (без кавычек). Вот примеры таких действий:

**Сообщить Х.**  
**Сообщить Х и Y, а также Z.**  
**Сообщить "Я работаю ВЬЧИСЛИТЕЛЕМ".**

3. Обозначить какой-либо буквой значение алгебраического выражения. Вот возможная краткая запись соответствующей команды:

**Присвоить ... значение ...**

Вместо второго многоточия пишется выражение, значение которого надо вычислить, а вместо первого многоточия — латинская буква, которой ВЬЧИСЛИТЕЛЬ обозначит это значение. Например:

**Присвоить С значение  $A+B$ .**  
**Присвоить Х значение  $2C$ .**

Разумеется, записывая в своих тетрадах алгоритмы, вы можете писать команды по-разному. Можно сокращать слова или, наоборот, добавлять слова, поясняющие, скажем, смысл переменных. Например:

**Запр. v**

или

**Запросить значение скорости v.**

Важно лишь, чтобы смысл каждой команды был понятен вам и в точности соответствовал допустимому действию исполнителя. Нельзя, например, записывать команду

**Сообщить Х или Y**

поскольку ВЬЧИСЛИТЕЛЬ не может сам выбрать, какое из чисел ему сообщать (у него нет соответствующего допустимого действия).

К сожалению, ЭВМ не способна понимать всевозможные фразы русского языка. Поэтому приходится жестко фиксировать запись команд. Например, при работе на ЭВМ переменные разрешается обозначать только латинскими буквами, в записи команды присваивания слово "Присвоить" не пишется, а вместо слова "значение" ставится знак "=". Скажем, вместо команды

### Присвоить $C$ значение $A+B$

вы будете писать

$C=A+B$

Обратите внимание, что точка в конце команд не ставится. О том, как записываются при работе на ЭВМ другие команды ВЫЧИСЛИТЕЛЯ, вы узнаете на лабораторной работе 6.

Вот пример алгоритма для ВЫЧИСЛИТЕЛЯ:

Запросить  $A, B$ .

Присвоить  $C$  значение  $(A+B)/2$ .

Сообщить  $C$ .

Этот алгоритм выполняется так. Сначала ВЫЧИСЛИТЕЛЬ запросит у нас два числа. Когда мы сообщим их ему, он обозначит первое число буквой  $A$ , второе число буквой  $B$ . Затем ВЫЧИСЛИТЕЛЬ найдет полусумму чисел  $A$  и  $B$ , а результат обозначит через  $C$ . После этого он сообщит значение  $C$ .

У вас, наверно, возникло ощущение, что с допустимыми действиями ВЫЧИСЛИТЕЛЯ вы освоились. И действительно, действия "Запросить" и "Сообщить" довольно просты. А вот действие "Присвоить" похитрее. Давайте немного изменим разобранный только что алгоритм:

Запросить  $A, B$ .

Присвоить  $A$  значение  $(A+B)/2$ .

Сообщить  $A$ .

Теперь полусумма чисел  $A$  и  $B$  будет обозначена не буквой  $C$ , а буквой  $A$ . Что же произойдет с тем числом, которое было обозначено буквой  $A$  до выполнения действия "Присвоить..."? ВЫЧИСЛИТЕЛЬ его забудет! И следующим своим действием он сообщит новое значение переменной  $A$  (т. е. полусумму исходных чисел).

Переменную удобно представлять себе как ящик, в котором можно хранить любое число. Когда в этот "ящик" кладут новое число, старое бесследно исчезает — допустимые действия ВЫЧИСЛИТЕЛЯ не позволяют никакой переменной присваивать два значения одновременно. Букву, обозначающую переменную, будем называть именем этой переменной. Имена переменных можно использовать в записи различных алгебраических выражений. При вычислении значений этих выражений имена заменяются числами из соответствующих "ящиков". Запросив число, ВЫЧИСЛИТЕЛЬ "кладет" его в "ящик", предварительно обозначив этот "ящик" соответствующей буквой.

Вот как можно представлять себе выполнение последнего из приведенных алгоритмов.

По команде "Запросить  $A$ ,  $B$ " **ВЫЧИСЛИТЕЛЬ** возьмет два "ящика", "прикрепит" на первый табличку " $A$ ", на второй — " $B$ ":

$A$



$B$



Затем он обратится к нам с вопросом, какие числа надо обозначить буквами  $A$ ,  $B$ . Узнав эти числа, скажем 2 и 12, **ВЫЧИСЛИТЕЛЬ** "положит" их в соответствующие "ящики":

$A$



$B$



После этого он приступит к выполнению второго действия. Взяв значения переменных  $A$  и  $B$ , **ВЫЧИСЛИТЕЛЬ** найдет их полусумму — число 7. Это число он "положит" в "ящик" с табличкой " $A$ ":

$A$



$B$



Как видите, число 2 уже забыто. Следующим действием **ВЫЧИСЛИТЕЛЬ** сообщит значение переменной  $A$  — число 7, взяв его из соответствующего "ящика".

С имитацией на ЭВМ действий **ВЫЧИСЛИТЕЛЯ** вы познакомитесь во время лабораторной работы 6.

Итак, для решения задачи надо выбрать исполнителя и лишь затем составлять алгоритм решения, используя только допустимые действия этого исполнителя. А что требуется от исполнителя? Только четкое выполнение каждого действия, входящего в алгоритм. Ему не надо знать, для каких целей предназначается алгоритм (компьютеру "безразлично", управляет он доменным процессом или предсказывает погоду на завтра).

*Ничто не исчезает  
так бесследно,  
как старое значение  
переменной  
после присваивания  
нового значения*

Впрочем, чтобы понять разницу между творцом и исполнителем, обязательно самому сочинить поэму, а потом переписать ее от руки в пяти экземплярах. Достаточно вникнуть в решение задачи 5 из § 9. Выполнить указанный в этой задаче алгоритм не представит труда для любого умеющего держать в руках циркуль и линейку. А вот придумать алгоритм построения серединного перпендикуляра может лишь существо разумное, владеющее геометрическими знаниями.

Теперь мы можем уточнить понятие алгоритма: **алгоритм** — это организованная последовательность действий, допустимых для некоторого исполнителя.

Один и тот же исполнитель может быть симитирован на ЭВМ многими способами. При этом смысл действий остается неизменным, а их названия могут быть разными. Так, при имитации ВЫЧИСЛИТЕЛЯ средствами языка Бейсик вместо слов "Запросить" и "Сообщить" используются английские слова "INPUT" и "PRINT".

## ? Вопросы

1. Что такое исполнитель алгоритмов?
2. Что такое допустимые действия исполнителя?
3. Какие действия допустимы для ВЫЧИСЛИТЕЛЯ?
4. Что означают для ВЫЧИСЛИТЕЛЯ следующие команды?
  - а) Запросить  $a, b, c$ .
  - б) Сообщить "Спасибо!".
  - в) Сообщить  $a+b$ .
  - г) Присвоить  $a$  значение  $x+y-z$ .
5. Что значит симитировать исполнителя с помощью ЭВМ?



## Задания для самостоятельного выполнения

1. Исполнитель умеет:  
умножать число на 2;  
увеличивать число на 1.
  - а) Составьте для этого исполнителя алгоритм получения числа 100 из единицы.
  - б)\* Сколько действий в самом коротком из таких алгоритмов?
2. Исполнитель умеет из любой дроби  $a/b$  получать любую из дробей  $(a-b)/b$ ,  $(a+b)/b$  и  $b/a$ . Например, из дроби  $5/8$  можно получить дроби  $-3/8$ ,  $13/8$  и  $8/5$ . Как получить из дроби  $1/2$  дробь  $1/4$ ? А как получить  $67/91$ ?
3. С числом, записанным на доске, разрешается производить два действия: умножать на 2 и стирать последнюю цифру. Например, из числа 56 можно получить числа 112 и 5.

- а) Как получить из числа 458 число 14?
- б)\* Как получить из произвольного натурального числа любое другое натуральное число?
4. Составьте алгоритм нахождения с помощью карандаша и линейки центра тяжести фигур на рисунке 13.



Рис. 13

5. На столе лежат две двухкопеечные монеты и три пятак так, как показано на рисунке 14.



Рис. 14

Исполнитель может одновременно перемещать две соседние монеты либо в начало или конец цепочки, либо на освободившееся в результате таких перемещений место в середине цепочки. Например, перенеся вторую и третью монеты к правому концу цепочки, он получит такое расположение монет (рис. 15):



Рис. 15

При этом исполнитель не может раздвигать монеты или менять их местами. Составьте для этого исполнителя алгоритм преобразования цепочки монет на рисунке 14 в цепочку монет на рисунке 16:



Рис. 16

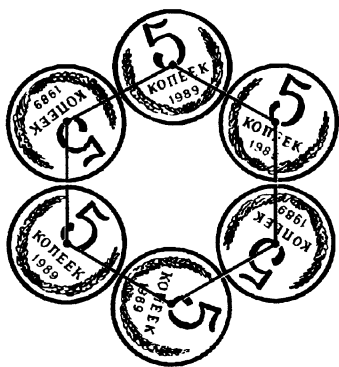


Рис. 17

6. Имеется шесть одинаковых монет. Надо расположить их на столе так, чтобы их центры были вершинами правильного шестиугольника (рис. 17). Исполнитель может лишь манипулировать имеющимися монетами, не производя никаких измерений и вычислений.

а) Составьте необходимый алгоритм.

б)\* Решите эту задачу при дополнительном ограничении: монеты запрещается поднимать над поверхностью стола.

7\*. Исполнитель умеет заменять в слове одну букву на другую так, чтобы получившееся слово имело смысл. Например:

"СЛОН" → "СЛОГ".

Напишите алгоритм превращения "МУХИ" в "СЛОНА".

8. Злоумышленник поменял местами действия в алгоритме вычисления среднего арифметического из квадратов трех чисел, предназначенном для ВЫЧИСЛИТЕЛЯ:

Присвоить  $a$  значение  $(a^2 + b^2 + c^2)/3$ .

Запросить  $a, b, c$ .

Сообщить "Среднее арифметическое квадратов равно".

Сообщить  $a$ .

Восстановите правильный порядок действий.

9. Исправьте следующий алгоритм решения уравнения  $(x-2)(x+2)=0$ , предназначенный для ВЫЧИСЛИТЕЛЯ:

Присвоить  $x$  значение  $\pm 2$ .

Сообщить "Корни уравнения равны".

Сообщить первое значение  $x$ .

Сообщить второе значение  $x$ .

10. Автомобиль проехал три участка пути разной длины с разными скоростями. Составьте для ВЫЧИСЛИТЕЛЯ алгоритм нахождения средней скорости автомобиля.

11. Даны длины сторон треугольника.

а) Составьте для ВЫЧИСЛИТЕЛЯ алгоритм, выполняя который он сначала вычислит (используя действие "Присвоить"), а затем сообщит полупериметр треугольника, его площадь и радиус вписанной окружности.

б) Составьте алгоритм решения той же задачи, в котором используются только четыре переменные.

в) Составьте алгоритм решения той же задачи, в котором используются только три переменные.

г)\* Можно ли составить алгоритм решения этой задачи, используя лишь одну переменную?

12. Перечислите допустимые действия известных вам электронной таблицы, информационно-поисковой системы, графического редактора, редактора текстов.

13\*. Какими допустимыми действиями вы снабдили бы исполнителя, который будет производить построения с помощью циркуля и линейки?



## Лабораторная работа 6

### КОММЕРЧЕСКАЯ ЗАДАЧА

Как вы уже знаете, для решения задачи с помощью ЭВМ нужна компьютерная модель этой задачи и симитированный на ЭВМ исполнитель. Модель задачи была создана в § 7. Что касается исполнителя, то у нас есть **ВЫЧИСЛИТЕЛЬ** (см. § 10): ведь наша модель математическая, а **ВЫЧИСЛИТЕЛЬ** как раз и предназначен для работы с числами.

Прежде чем приступить к решению задачи, давайте познакомимся с реализацией допустимых действий **ВЫЧИСЛИТЕЛЯ** ("Сообщить", "Запросить", "Присвоить") на ЭВМ. С командой "**СООБЩИТЬ**" вы уже знакомы по лабораторной работе 1. Помните ли вы, как выполняется эта команда?

Проверьте себя, заставив ЭВМ вычислить значение числового выражения (должно получиться ~ 235.43560086956)

$$\frac{(2,2^2+3,15:23)(-12,34+56)}{2}.$$

Команда

**ЗАПРОСИТЬ А,В**

выполняется так: на экране появится надпись "**ВВЕДИ ЧИСЛО**", после чего ЭВМ подождет, пока вы наберете число и нажмете клавишу "**ПЕРЕВОД СТРОКИ**". Затем **ВЫЧИСЛИТЕЛЬ** обозначит ваше число буквой А и попросит ввести еще одно число, обозначив его буквой В. И вообще по команде "**ЗАПРОСИТЬ**" **ВЫЧИСЛИТЕЛЬ** будет столько раз требовать у вас число, сколько букв записано после слова "**ЗАПРОСИТЬ**".

Обратите внимание: в команде "**ЗАПРОСИТЬ**" переменные надо разделять запятой, а не точкой, не пробелом и не точкой с запятой. На первый взгляд это мелочь, но при работе с машиной такая мелочь способна свести на нет весь ваш труд. Если заменить запятую на какой-нибудь

другой знак, то из хорошо знакомой ВYЧИСЛИТЕЛЮ команды получится бессмысленный для него набор символов. ВYЧИСЛИТЕЛЬ вежливо сообщит вам об этом.

Заставьте ВYЧИСЛИТЕЛЯ выполнить команду

**ЗАПРОСИТЬ А,В,С**

Функциональные клавиши облегчат вам набор команды и уберегут от многих ошибок при записи команд. Вспомните, как этими клавишами пользоваться.

Как проверить, правильно ли ВYЧИСЛИТЕЛЬ выполнил вашу команду? Хороший способ — заставить его напечатать значения А, В и С на экране, дав ему команду

**СООБЩИТЬ А,В,С**

Введите эту команду и убедитесь, что все в порядке.

А теперь давайте поручим ВYЧИСЛИТЕЛЮ выполнить несколько присваиваний. Введите команду

$A=2*3$

Курсор на мгновение исчез, а затем появился вновь. Это означает, что ВYЧИСЛИТЕЛЬ исполнил вашу команду. Проверьте с помощью команды

**СООБЩИТЬ А**

что значение переменной А стало равно 6.

Можно изменить значение А. Скажем, увеличить А на 3, дав, например, ВYЧИСЛИТЕЛЮ команду

$A=9$

Впрочем, даже не зная значения А, вы все равно можете заставить ВYЧИСЛИТЕЛЯ увеличить А на 3. Достаточно ввести команду

$A=A+3$

Эта запись может вас удивить: она похожа на уравнение. Но дело в том, что команда присваивания только с виду похожа на уравнение. В уравнениях одни и те же буквы обозначают одни и те же числа. А команда присваивания нужна как раз для того, чтобы изменять значения переменных. Действие команды  $A=A+3$  можно представить себе так: в "ящике", на котором написана буква А, лежит число (неважно какое); ВYЧИСЛИТЕЛЬ берет это число, увеличивает его на 3 и кладет обратно.

Итак, с имитацией ВYЧИСЛИТЕЛЯ на ЭВМ вы познакомились. Можно приступать к решению коммерческой задачи. Напомним построенную нами модель этой задачи.

Модель коммерческой задачи.

Дано: первоначальная стоимость магнитофона ( $m$  рублей), коэффициент ежегодной уценки ( $k$  процентов), стоимости ремонта магнитофона через 1, 2, 3, 4 и 5 лет после покупки ( $a, b, c, d, e$  рублей).

**Требуется определить**, через какое время после покупки ( $t$  лет) надо заменить магнитофон.

Связь между исходными данными и результатом:  $t$  равно номеру минимального из пяти чисел:

$a+a+b+c+d+m-m(k/100)$  (расходы в случае замены магнитофона через 1 год);

$a+b+a+b+c+m-m(k/100)^2$  (расходы в случае замены магнитофона через 2 года);

$a+b+c+a+b+m-m(k/100)^3$  (расходы в случае замены магнитофона через 3 года);

$a+b+c+d+a+m-m(k/100)^4$  (расходы в случае замены магнитофона через 4 года);

$a+b+c+d+e+m-m(k/100)^5$  (расходы в случае замены магнитофона через 5 лет).

Теперь, когда мы вспомнили модель задачи, пора приниматься за составление алгоритма ее решения. Ясно, что ВЫЧИСЛИТЕЛЬ должен подсчитать все пять чисел, сравнить их, а затем сообщить порядковый номер того, которое окажется самым маленьким. Однако тут нас ждет разочарование. Подсчитать и сообщить пять чисел ВЫЧИСЛИТЕЛЬ может, но как ему сравнить их? Ведь среди его допустимых действий нет сравнения чисел. Приходится сделать вывод: ВЫЧИСЛИТЕЛЬ не способен решить нашу задачу. Что же делать? Одно из двух: либо выбирать более "умелого" исполнителя, либо менять задачу. Впоследствии нам придется "научить" ВЫЧИСЛИТЕЛЯ сравнивать числа. Однако освоить использование этого действия нелегко. Поэтому сейчас мы пойдем по другому пути. Пусть ВЫЧИСЛИТЕЛЬ сообщит все пять чисел, а уж мы сами потом посмотрим, которое из них меньше. Итак, возможности исполнителя потребовали от нас изменения модели коммерческой задачи. Вместо одного числа результатом теперь являются пять чисел.

Новая модель коммерческой задачи.

**Дано:** первоначальная стоимость магнитофона ( $m$  рублей), коэффициент ежегодной уценки ( $k$  процентов), стоимости ремонта магнитофона через 1, 2, 3, 4 и 5 лет после покупки ( $a, b, c, d, e$  рублей).

**Требуется определить** пять чисел — расходы на прослушивание музыки в случае замены магнитофона через 1, 2, 3, 4 и 5 лет.

**Связь между исходными данными и результатами:**

Расходы в случае замены магнитофона через 1 год равны  $a+a+b+c+d+m-m(k/100)$  рублей.

Расходы в случае замены магнитофона через 2 года равны  $a+b+a+b+c+m-m(k/100)^2$  рублей.

Расходы в случае замены магнитофона через 3 года

равны  $a+b+c+a+b+m-m(k/100)^3$  рублей.

Расходы в случае замены магнитофона через 4 года равны  $a+b+c+d+a+m-m(k/100)^4$  рублей.

Расходы в случае замены магнитофона через 5 лет равны  $a+b+c+d+e+m-m(k/100)^5$  рублей.

Как видите, мы еще раз столкнулись с важнейшим принципом, что модель и исполнитель должны соответствовать друг другу. Только в этом случае задача может быть решена с помощью этого исполнителя.

Нетрудно написать для ВЫЧИСЛИТЕЛЯ алгоритм решения новой задачи. Вот он:

Запросить стоимость магнитофона  $m$  и процент уценки  $k$ .

Запросить  $a, b, c, d, e$  — стоимости ремонта через 1, 2, 3, 4 и 5 лет.

Присвоить  $z$  значение  $a+a+b+c+d+m-m(k/100)$ .

Сообщить  $z$ .

Присвоить  $z$  значение  $a+b+a+b+c+m-m(k/100)^2$ .

Сообщить  $z$ .

Присвоить  $z$  значение  $a+b+c+a+b+m-m(k/100)^3$ .

Сообщить  $z$ .

Присвоить  $z$  значение  $a+b+c+d+a+m-m(k/100)^4$ .

Сообщить  $z$ .

Присвоить  $z$  значение  $a+b+c+d+e+m-m(k/100)^5$ .

Сообщить  $z$ .

Чтобы ЭВМ выполнила этот алгоритм, превратим его в программу. Сделать это просто. Надо только для каждого действия из алгоритма записать соответствующую команду и эти команды перенумеровать:

1 ЗАПРОСИТЬ  $M, K$

2 ЗАПРОСИТЬ  $A, B, C, D, E$

3  $Z = A+A+B+C+D+M-M*(K/100)$

4 СООБЩИТЬ  $Z$

5  $Z = A+B+A+B+C+M-M*(K/100)^2$

6 СООБЩИТЬ  $Z$

7  $Z = A+B+C+A+B+M-M*(K/100)^3$

8 СООБЩИТЬ  $Z$

9  $Z = A+B+C+D+A+M-M*(K/100)^4$

10 СООБЩИТЬ  $Z$

11  $Z = A+B+C+D+E+M-M*(K/100)^5$

12 СООБЩИТЬ  $Z$

Если ваша ЭВМ не умеет имитировать ВЫЧИСЛИТЕЛЯ, но понимает язык Бейсик, переведите эту программу на Бейсик. Для этого достаточно заменить в ней слово СООБЩИТЬ на PRINT, а слово ЗАПРОСИТЬ на INPUT.

Теперь наберите текст составленной программы на клавиатуре ЭВМ. Только не забывайте нажимать на клавишу "ПЕРЕВОД СТРОКИ" после набора каждой строки (последний раз напоминаем вам об этом!). Если же вы допустили синтаксическую или орфографическую ошибку (не поставили в нужном месте запятую или написали "СААПЧ-ЩИТЬ"), машина сообщит вам об этом. Не теряйтесь, найдите ошибку и исправьте ее, как вас учили на лабораторной работе 1.

Вводя программу, вы могли случайно пропустить какую-нибудь строку. Не беда, строки программы можно набирать в любом порядке, главное — верно их нумеровать. ВЫЧИСЛИТЕЛЬ сам расставит строки в нужном порядке.

Может случиться и так, что вы ввели лишнюю строку. Для того чтобы уничтожить ее, надо набрать номер этой строки, а затем клавишу "ПЕРЕВОД СТРОКИ". Недостаточно просто стереть строку на экране — в памяти ВЫЧИСЛИТЕЛЯ она останется.

Вы, возможно, удивились, что машина не торопится, как раньше, выполнять ваши команды после нажатия клавиши "ПЕРЕВОД СТРОКИ". Чем же эти команды отличаются от тех, которые вы вводили раньше? Тем, что они занумерованы. В этом-то все и дело. Команду без номера ЭВМ выполняет немедленно, но не запоминает ее; занумерованную команду ЭВМ запоминает, но будет исполнять ее только по специальному указанию. Таким указанием служит команда "ПУСК" (на Бейсике — команда "RUN").

Не спешите, однако, вводить команду "ПУСК", т. е., как говорят программисты, "запускать программу на исполнение". Лучше сначала вызвать текст программы на экран, чтобы проверить, правильно ли ЭВМ ее запомнила. Для этого служит команда "ТЕКСТ" (на Бейсике — команда "LIST"). Если найдете ошибку — исправьте ее, снова проверьте и т. д. Только после того, как вы убедитесь, что машина правильно поняла, что ей надо будет делать, запускайте программу.

Ура! Машина начала выполнять первую вашу программу: появилась надпись "ВВЕДИ ЧИСЛО" (или вопросительный знак, если вы используете Бейсик). ВЫЧИСЛИТЕЛЬ просит ввести  $m$  — стоимость магнитофона. Для начала примем ее равной 5000 р. Как вы помните, сведения о проценте уценки магнитофона и стоимости ремонта мы получили от знакомых товароведов комиссионного магазина и директора радиомастерской. Вот эти числа:  $k=90\%$ ,  $a=0$  р.,  $b=100$  р.,  $c=150$  р.,  $d=700$  р.,  $e=1500$  р. Введите эти числа. Немного подумав, ЭВМ сообщит вам пять чисел — расходы на эксплуатацию магнитофона в случае за-

мены магнитофона через 1, 2, 3, 4 и 5 лет. Минимальное из этих чисел покажет вам, через сколько лет наиболее выгодно поменять магнитофон.

Итак, мы составили программу и получили ответ. Задача вроде бы решена. Однако самое интересное — исследовательская часть вычислительного эксперимента — начинается только теперь, после получения ответа.

Вспомним: исходные данные известны нам лишь приближенно. Зависит ли результат от небольших изменений этих чисел? Что произойдет, если магнитофон оказался немного дешевле или дороже? А что, если изменились стоимости ремонтов или процент уценки? Не знаем, как вы, но мы с удивлением обнаружили, что наиболее выгодный год замены магнитофона не зависит от изменений начальных данных (если изменения не слишком сильны).

А может быть, расходы на эксплуатацию магнитофона можно уменьшить? Для этого (вспомните задачу о подкове) надо построить модель, более близкую к реальности. Проанализировав нашу модель еще раз, мы поняли, что нелогично запрещать школьнику-меломану замену магнитофона чаще одного раза в пять лет. Давайте разрешим школьнику менять магнитофон дважды. При этом модель, конечно, изменится. Изменится и программа (она дополнится несколькими строками). Внесите эти изменения. Удалось ли сократить расходы?

## § 11. ДОСТИЖИМЫЕ ЦЕЛИ

Обычно никто не составляет алгоритмы просто так: как правило, алгоритмы служат для решения какой-либо задачи, для достижения какой-либо цели. Допустим, вы выбрали исполнителя для решения некоторого класса задач. Но тогда сразу встает вопрос, может ли имеющийся в вашем распоряжении исполнитель решить поставленную задачу. Чтобы уметь отвечать на такой вопрос, надо

*Алгоритм  
не роскошь,  
а средство  
достижения цели*

знать, какие цели достижимы тем или иным исполнителем, т. е. *какие результаты он может получить с помощью своих допустимых действий.*

Как правило, задача определения всех достижимых целей исполнителя весьма сложна. Поэтому давайте рассмотрим простого исполнителя, который может передвигаться по бумаге, рисуя линии. Назовем его **ЧЕРТЕЖНИК**. Будем изображать направление движения **ЧЕРТЕЖНИКА**

стрелкой. Пишущий узел находится в начале стрелки. Для ЧЕРТЕЖНИКА допустимы всего два действия:

1) пройти 1 см в направлении стрелки, рисуя линию (рис. 18,а);

2) повернуться на  $90^\circ$  влево вокруг начала стрелки (рис.18,б).

Будем считать, что первое действие ЧЕРТЕЖНИК выполняет по команде "Сделать шаг", а второе действие — по команде "Повернуть налево".

Давайте с помощью ЧЕРТЕЖНИКА нарисует линию, изображенную на рисунке 19. Длина каждого отрезка равна 2 см. Начальное положение ЧЕРТЕЖНИКА указано стрелкой. Приведем алгоритм, с помощью которого решается эта задача:

Повернуть налево.  
Сделать шаг.  
Сделать шаг.  
Повернуть налево.  
Сделать шаг.  
Сделать шаг.  
Повернуть налево.  
Сделать шаг.  
Сделать шаг.

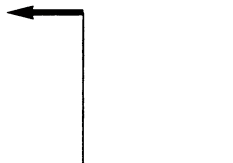


Рис. 19

А такой рисунок он начертить не сможет:

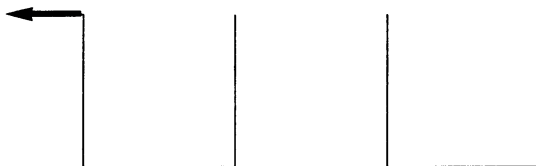


Рис. 20

Ясно, что ЧЕРТЕЖНИК вообще не может изображать разрывные линии. А какие рисунки он может изобразить? Другими словами, какие цели достижимы для ЧЕРТЕЖНИКА? Ответ несложен: ЧЕРТЕЖНИК может рисовать только непрерывные ломаные, у которых длины звеньев целочисленны и все углы прямые.

Что надо сделать, чтобы ЧЕРТЕЖНИК смог рисовать разрывные линии? Ясно, что для этого надо увеличить набор допустимых действий. Например, добавить в список команд ЧЕРТЕЖНИКА команду "Прыгнуть", по которой он делает шаг длины 1 см, не рисуя линии. Теперь можно со-

ставить алгоритм рисования, например, фигуры, изображенной на рисунке 20.

Повернуть налево.  
Сделать шаг.  
Повернуть налево.  
Сделать шаг.  
Повернуть налево.  
Сделать шаг.  
Повернуть налево.  
Повернуть налево.  
Повернуть налево.  
Прыгнуть.  
Повернуть налево.  
Повернуть налево.  
Повернуть налево.  
Сделать шаг.  
Повернуть налево.  
Сделать шаг.  
Повернуть налево.  
Сделать шаг.

В дальнейшем мы будем считать, что ЧЕРТЕЖНИК умеет выполнять команду "Прыгнуть".

### ? Вопросы

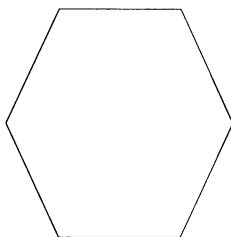
1. Что называется достижимыми целями исполнителя?
2. Каковы допустимые действия ЧЕРТЕЖНИКА?



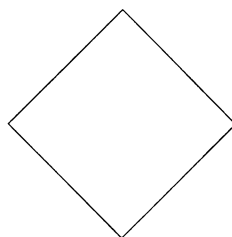
а)



б)



в)



г)

Рис. 21

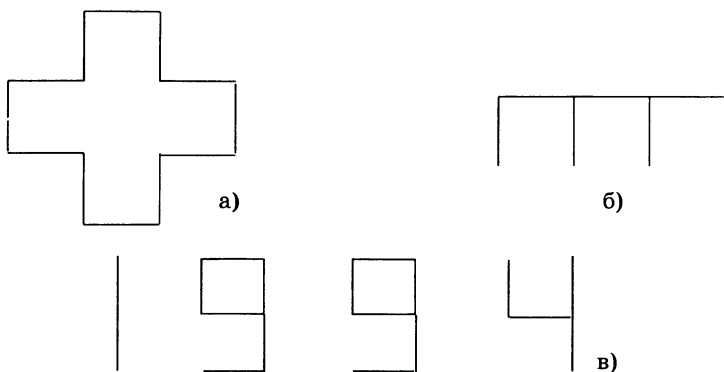


Рис. 22



### Задания для самостоятельного выполнения

1. Какие из этих линий (рис. 21) может нарисовать ЧЕРТЕЖНИК?

2. Написать для ЧЕРТЕЖНИКА алгоритмы, выполнив которые он нарисует заданные фигуры (рис. 22).

3. Какие цели достижимы для ЧЕРТЕЖНИКА, умеющего выполнять лишь следующие действия:

- "Сделать шаг" и "Прыгнуть";
- "Прыгнуть" и "Повернуть налево";
- "Повернуть налево"?

4. ЧЕРТЕЖНИК-30 и ЧЕРТЕЖНИК-45 очень похожи на обычного ЧЕРТЕЖНИКА. Единственное отличие состоит в том, как они понимают команду ПОВЕРНУТЬ НАЛЕВО. По этой команде ЧЕРТЕЖНИК-30 поворачивается на  $30^\circ$ , а ЧЕРТЕЖНИК-45 — на  $45^\circ$  против часовой стрелки. Какого из этих двух ЧЕРТЕЖНИКОВ вы предпочтете для рисования следующих фигур:

- квадрат со стороной 1 см;
- правильный шестиугольник со стороной 1 см;
- правильный восьмиугольник со стороной 1 см;
- отрезок длины 5 см?

5. Для каждого натурального  $N$  создан ЧЕРТЕЖНИК- $N$ , который по команде ПОВЕРНУТЬ НАЛЕВО поворачивается на угол  $N^\circ$  против часовой стрелки. Найти такое  $N$ , что для ЧЕРТЕЖНИКА- $N$  достижима каждая из двух целей: "нарисовать правильный десятиугольник" и "нарисовать квадрат". Для какого  $N$  суммарное число действий в алгоритмах рисования этих фигур наименьшее?

6. Предположим, что в условии задачи 11 из § 9 вместо трехлитрового кувшина дан двухлитровый. Можно ли теперь набрать 7 л?

7. Даны три листа бумаги. Исполнитель берет лист, разрезает его на четыре части и кладет их обратно. Какое

количество листов может получиться в результате его работы?

8\*. На доске написаны числа 1, 2, 3, ...,  $n$ . Исполнитель может стереть два числа и записать вместо них абсолютную величину их разности. Через  $n-1$  шаг на доске останется одно число. Цель — получить число 0. Для каких  $n$  эта цель достижима?

9. Какими допустимыми действиями вы снабдили бы автомат, заменяющий:

- а) кассира в магазине;
- б) дворника;
- в) вахтера;
- г) директора школы?

10. Можно ли создать исполнителя, который все может?



### КОНСПЕКТ ГЛАВЫ 3

Каждый человек постоянно выполняет различные алгоритмы. Обычно нет необходимости думать о том, какие действия и в каком порядке при этом совершаются. Если же алгоритм требуется объяснить человеку, ранее с ним не знакомому (или, скажем, ЭВМ), то алгоритм необходимо представить в виде четкой последовательности простейших действий. При этом важен не только набор действий, но и их порядок. Если в алгоритме изменить порядок действий, то он может стать непригодным для решения исходной задачи и даже вообще невыполнимым.

Чтобы подчеркнуть порядок действий в алгоритме, их обычно записывают в столбик одно под другим.

Исполнять алгоритмы может не только человек, но и автоматическое устройство. **Исполнитель алгоритмов** — это устройство управления, соединенное с набором инструментов. Устройство управления воспринимает и анализирует алгоритм, а затем организует его выполнение, командуя соответствующими инструментами. А инструменты производят действия, выполняя команды управляющего устройства.

От исполнителя требуется лишь четкое выполнение каждого действия, входящего в алгоритм. Мы не должны объяснять ему, для каких целей предназначается алгоритм.

Возможности любого исполнителя всегда ограничены. Поэтому, прежде чем составлять алгоритм решения задачи, нужно узнать, какие действия исполнитель может выполнить. Эти действия называются **допустимыми действиями** исполнителя. При составлении алгоритмов только их и можно использовать.

Для решения одних и тех же задач исполнители с

более бедным набором допустимых действий требуют более сложных и подробных алгоритмов.

Итак, алгоритм — это организованная последовательность действий, допустимых для некоторого исполнителя. Это не строгое определение алгоритма. Строгого определения алгоритма нет: в информатике это понятие фундаментальное.

Разные классы задач требуют разных наборов допустимых действий — разных исполнителей. Обычно набор допустимых действий явно или неявно указывается в формулировке задачи.

Чтобы для решения того или иного класса задач можно было применить ЭВМ, необходимо симитировать на ЭВМ нужный набор допустимых действий. Это называют имитацией исполнителя с помощью ЭВМ. Один и тот же исполнитель может быть симитирован на ЭВМ многими способами. При этом содержание действий остается неизменным, а их названия могут быть разными.

Задачи на вычисления — один из наиболее широких классов задач, решаемых на ЭВМ. Исполнитель вычислительных алгоритмов называется ВЫЧИСЛИТЕЛЕМ. Он имеет дело с числами и переменными. Вот некоторые из его допустимых действий:

1. Запросить исходные данные и обозначить их латинскими буквами. Это можно записывать так: "Запросить...", указывая вместо многоточия соответствующие буквы. Например:

Запросить *A, B, C*.

2. Сообщить результаты вычислений или какой-нибудь текст: "Сообщить..." Здесь вместо многоточия ставятся буквы или алгебраические выражения, значения которых надо напечатать, или текст, заключенный в кавычки. Например,

Сообщить *X, Y*.

Сообщить "Я работаю **ВЫЧИСЛИТЕЛЕМ**".

3. Обозначить какой-либо буквой значение алгебраического выражения:

Присвоить ... значение...

Вместо второго многоточия пишется выражение, значение которого надо вычислить, а вместо первого многоточия — латинская буква, которой **ВЫЧИСЛИТЕЛЬ** обозначит это значение. Например:

Присвоить *C* значение  $A+2B$

Остальные действия будут перечислены в следующих главах.

Записывая алгоритмы в тетради, можно использовать сокращения или, скажем, добавлять пояснения. Например: "Запр.  $v$ " или "Запросить значение скорости  $v$ ". Важно лишь, чтобы каждая команда в точности соответствовала допустимому действию исполнителя. Нельзя, например, записывать для ВЫЧИСЛИТЕЛЯ команду "Сообщить  $X$  или  $Y$ ", поскольку он не может сам выбрать, какое из чисел сообщать (у него нет соответствующего допустимого действия).

При записи программ для ВЫЧИСЛИТЕЛЯ, имитируемого на ЭВМ, запись команд жестко фиксируется и все команды нумеруются (необязательно подряд). Например, в команде присваивания сначала пишется имя переменной, которой присваивается значение выражения, затем знак равенства и само выражение.

Вот пример записи алгоритма для ВЫЧИСЛИТЕЛЯ:

Запр. знач. переменных  $A, B$ .

Присвоить  $C$  знач.  $(A+B)/2$ .

Сообщить полученное значение  $C$ .

Для ЭВМ соответствующая программа выглядит так:

1 ЗАПРОСИТЬ  $A, B$

4  $C=(A+B)/2$

10 СООБЩИТЬ  $C$

Этот алгоритм выполняется следующим образом. Сначала ВЫЧИСЛИТЕЛЬ запросит у нас два числа. Когда мы сообщим их ему, он обозначит первое число буквой  $A$ , второе число буквой  $B$ . Затем ВЫЧИСЛИТЕЛЬ найдет полусумму чисел  $A$  и  $B$ , а результат обозначит через  $C$ . После этого он сообщит значение  $C$ .

Каждую переменную, с которой работает ВЫЧИСЛИТЕЛЬ, удобно представлять себе как ящик, в котором можно хранить любое число. Когда в такой "ящик" кладут новое число, старое бесследно исчезает. Допустимые действия ВЫЧИСЛИТЕЛЯ не позволяют никакой переменной присваивать два значения одновременно. Букву, обозначающую переменную, называют именем этой переменной. Имена переменных можно использовать в записи различных алгебраических выражений. При вычислении значений этих выражений имена заменяются числами из соответствующих "ящиков". Запросив число, ВЫЧИСЛИТЕЛЬ "кладет" его в "ящик", предварительно обозначив этот "ящик" соответствующей буквой.

Поясним сказанное на примере команды

Присвоить  $A$  значение  $A+3$

В ящике, на котором написана буква  $A$ , лежит число; ВЫ-

**ЧИСЛИТЕЛЬ** берет это число, увеличивает его на 3 и кладет обратно.

Исполнитель **ЧЕРТЕЖНИК** предназначен для рисования различных фигур. Он может передвигаться по бумаге, рисуя линии. Направление движения **ЧЕРТЕЖНИКА** изображается стрелкой. Пишущий узел находится в начале стрелки. Для **ЧЕРТЕЖНИКА** допустимы всего три действия:

1. Пройти 1 см в направлении стрелки, рисуя линию.
2. Повернуться на  $90^\circ$  влево вокруг начала стрелки.
3. Прыгнуть на 1 см в направлении стрелки, не рисуя линии.

При имитации на ЭВМ **ЧЕРТЕЖНИК** выполняет первое действие по команде "СДЕЛАТЬ ШАГ", второе действие по команде "ПОВЕРНУТЬ НАЛЕВО", а третье — по команде "ПРЫГНУТЬ".

Алгоритмы служат для решения каких-либо задач, для достижения каких-либо целей. Поэтому очень важен ответ на вопрос: может ли данный исполнитель решить поставленную задачу, т. е. какие **цели достижимы** этим исполнителем? Как правило, задача определения всех достижимых целей исполнителя весьма сложна.

Для расширения класса достижимых целей исполнителя надо расширить набор его допустимых действий.

## Глава 4

# ВЕТВЛЕНИЯ В АЛГОРИТМАХ

---

В главе 3 мы рассматривали только такие алгоритмы, в которых все действия совершаются одно за другим независимо ни от чего. Такие алгоритмы называются **линейными**; характерная для них форма организации действий — последовательное выполнение. В этой главе мы обсудим алгоритмы, в которых какое-либо действие совершается в зависимости от выполнения или невыполнения некоторого условия.

### § 12. ВЕТВЛЕНИЯ

Легко и просто было бы жить (и даже неинтересно), если бы удалось раз и навсегда расписать, какие поступки и в какой последовательности совершать. На самом деле нам постоянно приходится принимать решения в зависимости от создавшейся ситуации. Если идет дождь, то мы надеваем плащ. Если жарко, то идем купаться. Разумеется, встречаются и более сложные положения, когда надо сделать выбор. Рассмотрим два примера.

**Пример 1.** Допустим, вы собрались пойти в кинотеатр на сеанс 12.00. Алгоритм покупки билетов может выглядеть так:

Подойти к кассе.

Если билеты на сеанс 12.00 есть, то купить билеты.

Отойти от кассы.

**Пример 2.** Представьте себе, что вам нужно проехать к автозаправочной станции (АЗС) по дороге, участок которой ремонтировался, и вам неизвестно, закончился ли ремонт. Подъезжая к этому участку, вы будете вынуждены воспользоваться алгоритмом, подобным следующему:

Уменьшить скорость.

Если ремонт участка закончен, то проехать 5 км по отремонтированному участку, иначе проехать 10 км в объезд.

Остановиться у АЗС.

"Билеты на сеанс 12.00 есть" — это условие, которое

надо проверить в первом примере. Во втором примере проверяется условие "ремонт участка закончен".

Что же происходит после проверки условия? В первом примере, если условие выполнено, совершается действие

купить билеты,  
а затем действие  
отойти от кассы.

Если же условие не выполнено, то сразу совершается действие

отойти от кассы.

Во втором примере в случае выполнения условия совершается действие

проехать 5 км по отремонтированному участку,  
а затем действие  
остановиться у АЗС.

В противном случае совершается действие

проехать 10 км в объезд,  
а затем действие  
остановиться у АЗС.

Мы видим, что при выполнении каждого из этих алгоритмов наступает такой момент, когда появляется несколько направлений для продолжения. Алгоритм как бы раздваивается, разветвляется (словно дорога). В этом случае говорят, что алгоритм содержит **ветвление**.

В рассмотренных ветвлениях как "прямой путь", так и "объезд" содержит только одно действие. Но так бывает далеко не всегда. Скажем, в первом примере слова "Купить билеты" предполагают выполнение нескольких действий, например таких: протянуть кассиру деньги, назвать сеанс и количество билетов, получить билеты. Учитывая это, попробуем записать алгоритм покупки билетов более подробно. Как и раньше, будем записывать действия в столбик:

*Только ветвление  
поможет  
в сложных условиях  
сделать выбор*

**Подойти к кассе.**  
**Если билеты на сеанс 12.00 есть, то:**  
**Протянуть кассиру деньги.**  
**Назвать сеанс и количество билетов.**  
**Получить билеты.**  
**Отойти от кассы.**

Казалось бы, все в порядке: мы лишь разъяснили, что значит "купить билеты". Но по этой записи стало невозможно понять очередность выполнения действий, поскольку не-

**Указатель**  
**"Конец ветвления" —**  
**лучшее средство**  
**от двусмысленности!**

ясно, какое именно действие следует выполнять, если билетов на сеанс 12.00 нет. Значит, в записи алгоритма необходим специальный указатель, показывающий последнее действие, участвующее в ветвлении. Договоримся в качестве такого указателя употреблять слова "Конец ветвления" и писать его в строке, следующей за последним действием ветвления:

Подойти к кассе.  
 Если билеты на сеанс 12.00 есть, то:  
 Протянуть кассиру деньги.  
 Назвать сеанс и количество билетов.  
 Получить билеты.  
 Конец ветвления.  
 Отойти от кассы.

Теперь ясно: если условие не выполнено, то сразу совершается действие, записанное после слов "Конец ветвления".

Ветвления в алгоритмах записывают одним из следующих двух способов.

*Первый способ:*

Если  $Q$ , то:  
 $P_1$   
 $P_2$   
 $\dots$   
 $P_n$   
 Иначе:  
 $T_1$   
 $T_2$   
 $\dots$   
 $T_m$   
 Конец ветвления

*Второй способ:*

Если  $Q$ , то:  
 $P_1$   
 $P_2$   
 $\dots$   
 $P_n$   
 Конец ветвления

Здесь  $Q$  — условие,  $P_1, P_2, \dots, P_n$  — действия, которые совершаются в случае выполнения условия  $Q$  (первая ветвь ветвления), а  $T_1, T_2, \dots, T_m$  — действия, которые совершаются, если это условие не выполнено (вторая ветвь ветвления). Первый из этих способов записи называется ветвлением в полной форме, второй — ветвлением в неполной форме.

Правда, договорившись, как оформлять ветвления в алгоритмах, мы должны быть уверены в том, что исполнители сумеют исполнить такие алгоритмы. Мы уже говорили, что любой исполнитель снабжен специальным устройством управления, которое "воспринимает" алгоритмы и организует их исполнение. Будем считать, что все устройства

управления "понимают" и ветвления и вообще любые формы организации действий, о которых речь пойдет далее.

Как же исполнители воспринимают алгоритмы с ветвлениями? Встретив в алгоритме ветвление, устройство управления исполнителя проверяет, выполняется ли условие, и в зависимости от этого дает команду на выполнение одной из последовательностей действий  $P_1, P_2, \dots, P_n$  или  $T_1, T_2, \dots, T_m$ . Как только исполнитель совершил выбранную последовательность действий, ветвление заканчивается, т. е. исполнение алгоритма продолжается с действия, следующего за словами "Конец ветвления".

Любое ли условие можно записывать в качестве  $Q$ ? Конечно, нет! Исполнитель должен уметь проверять, выполняется ли это условие. Иными словами, *проверка условия должна быть допустимым действием исполнителя*. Например, бессмысленно включать в алгоритм подготовки к экзамену такое ветвление:

Если завтра попадется билет № 13, то:

Учить билет № 13.

Конец ветвления.

(Не пользовался ли кто-нибудь из вас подобным алгоритмом?)

Очевидно, что пока вы не можете составлять разветвляющиеся алгоритмы для ЧЕРТЕЖНИКА: набор его допустимых действий не содержит проверок каких-либо условий. По той же причине нельзя составить разветвляющийся алгоритм для ВЫЧИСЛИТЕЛЯ, используя только известные вам действия "Запросить", "Сообщить" и "Присвоить". Именно поэтому на лабораторной работе 6 вы не смогли решить с помощью ВЫЧИСЛИТЕЛЯ первую модель коммерческой задачи.

Итак, ветвление (развилка) — это такая форма организации действий, при которой в зависимости от выполнения или невыполнения некоторого условия совершается либо одна, либо другая последовательность действий.

Чтобы более наглядно представлять те или иные формы организации действий, очень полезны так называемые блок-схемы. Каждое действие алгоритма, кроме проверки условия, будем помещать в прямоугольник, а вопрос о том, выполняется ли некоторое условие, — в ромб. Блоки будем соединять отрезками или отрезками со стрелками, показывая очередность выполнения действий. Если на таком отрезке стрелка отсутствует, то блок, расположенный ниже, должен выполняться позже. Блок-схемы на рисунке 23, а, б, в изображают соответственно последовательное выполнение действий, ветвления в полной и неполной формах. На рисунке 24 изображена блок-схема алгоритма покупки би-

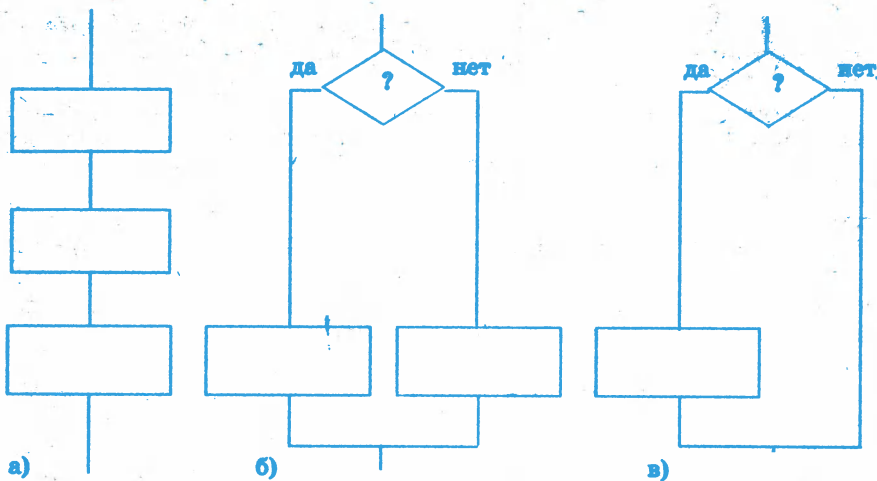


Рис. 23

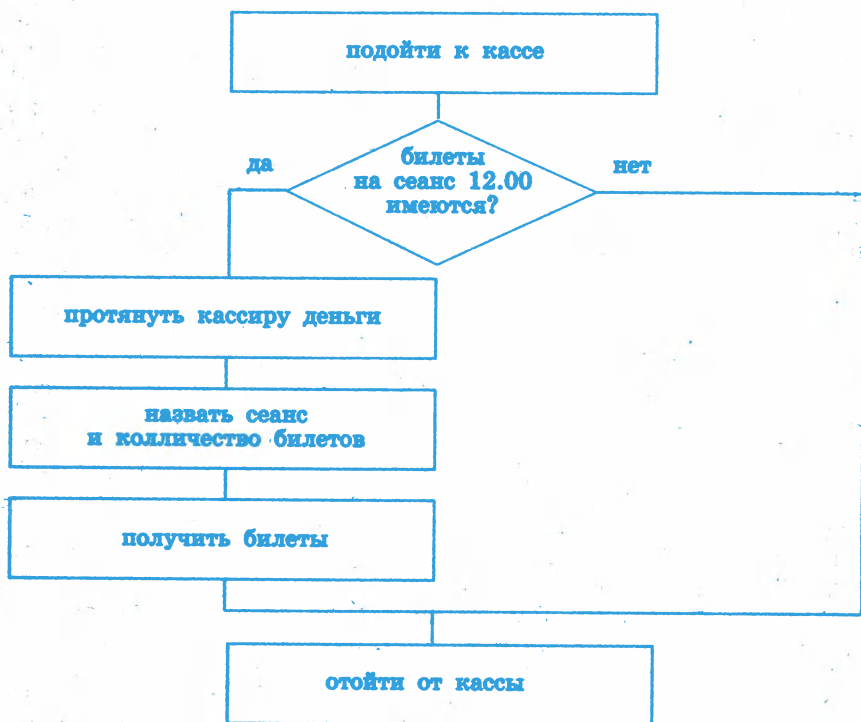


Рис. 24

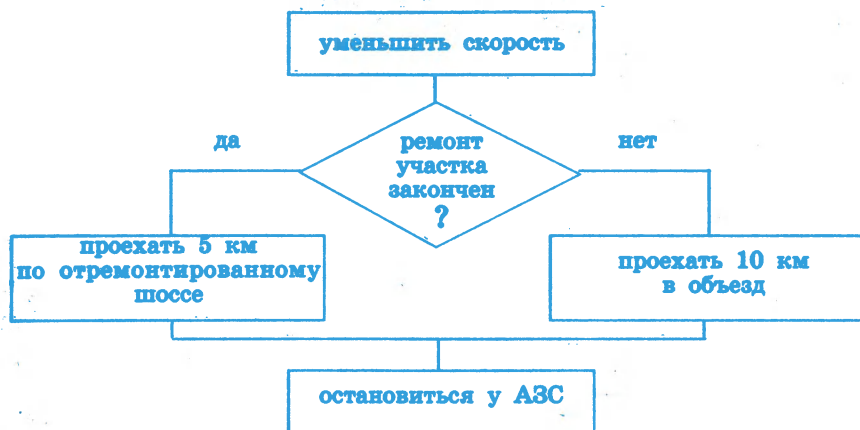


Рис. 25

летов, а на рисунке 25 изображена блок-схема алгоритма поездки на автозаправочную станцию.

## ? Вопросы

1. Какие алгоритмы называются линейными?
2. Какая форма организации действий называется ветвлением?
3. Как оформляются в алгоритмах ветвления:
  - а) в неполной форме;
  - б) в полной форме?
4. Для чего служит указатель "Конец ветвления"?
5. Почему ВЫЧИСЛИТЕЛЬ и ЧЕРТЕЖНИК могут выполнять только линейные алгоритмы?
6. Как изображаются с помощью блок-схем:
  - а) линейные алгоритмы;
  - б) ветвления в неполной форме;
  - в) ветвления в полной форме?



## Задания для самостоятельного выполнения

1. а) Проснувшись утром, школьник почувствовал недомогание. Находившийся рядом злоумышленник тут же составил для него следующий алгоритм:

Измерить температуру.  
 Если температура выше  $37^{\circ}$ , то:  
 Вызвать врача.  
 Пойти в школу.

Несмотря на недомогание, школьник исправил этот алгоритм, добавив всего две строки. Какие строки добавил школьник?

б) Однажды школьник решил из своего дома позвонить приятелю. Злоумышленник, который и на этот раз оказался рядом, предложил ему следующий алгоритм:

Подойти к телефону.  
Снять трубку.  
Набрать номер.  
Подождать 6 секунд.  
Если знакомый ответит, то:  
Сказать: "Здравствуй!"  
Сообщить последние новости.  
Узнать, что нового и как жизнь.  
Сказать: "До свидания!"  
Положить трубку.  
Конец ветвления.  
Отойти от телефона.

Школьник воспользовался этим алгоритмом, и через некоторое время у него отключили телефон. Объясните почему.

2. Для решения каких задач предназначены следующие алгоритмы? Укажите, какие условия в них проверяются, какие действия совершаются, если условия выполнены, и какие — если условия не выполнены. Нарисуйте блок-схемы этих алгоритмов.

а) Присвоить  $x$  значение суммы углов  $A$  и  $C$  четырехугольника  $ABCD$ .

Присвоить  $y$  значение суммы углов  $B$  и  $D$  четырехугольника  $ABCD$ .

Если  $x=y$ , то:

Построить серединный перпендикуляр к отрезку  $AB$ .

Построить серединный перпендикуляр к отрезку  $BC$ .

Найти пересечение построенных перпендикуляров.

Иначе:

Сообщить "Построение невозможно".

Конец ветвления.

б) Присвоить  $x$  значение суммы сторон  $AB$  и  $CD$  четырехугольника  $ABCD$ .

Присвоить  $y$  значение суммы сторон  $BC$  и  $AD$  четырехугольника  $ABCD$ .

Если  $x=y$ , то:

Построить биссектрису угла  $A$ .

Построить биссектрису угла  $B$ .

Найти пересечение построенных биссектрис.

Иначе:

Сообщить "Построение невозможно".

Конец ветвления.

3. На экране ЭВМ, в которую загружен учебный редак-

тор текстов, напечатано некоторое слово. Курсор находится на первой букве этого слова. Школьник выполнил следующий алгоритм:

Нажать клавишу "Стрелка вправо".  
Нажать клавишу "Стрелка вправо".  
Если курсор находится на букве "а", то:  
Нажать на клавишу с буквой "о".  
Конец ветвления.

Какое слово будет написано на экране после выполнения этого алгоритма, если первоначально было записано слово:

- а) слава;
- б) олово?

4. Однажды зимой учитель физкультуры объявил, что занятия на улице будут проходить только при температуре не ниже 15 градусов мороза. Школьник, собираясь утром в школу, размышляет, брать ему лыжный костюм или форму для зала. Какой алгоритм должен исполнить школьник, чтобы отправиться в школу в нужной экипировке?

5. Запишите в виде алгоритмов правила определения знака:

- а) произведения двух действительных чисел;
- б) суммы двух действительных чисел.

### § 13. ВЕТВЛЕНИЯ И ИСПОЛНИТЕЛИ АЛГОРИТМОВ

Мы уже отмечали, что, используя действия ВЫЧИСЛИТЕЛЯ "Запросить", "Сообщить", "Присвоить", нечего и пытаться составить для него разветвляющийся алгоритм: среди этих действий нет проверок условий. А ведь в реальной жизни постоянно возникают вычислительные задачи, при решении которых надо проверять те или иные условия. Какие же условия обычно приходится проверять? Конечно, эти условия связаны со сравнением чисел. Поэтому будем считать, что ВЫЧИСЛИТЕЛЬ "умеет" сравнивать числа, т. е. проверять, выполняется ли для двух чисел заданное соотношение ( $<$ ,  $>$ ,  $=$ ,  $\neq$ ,  $\leq$ ,  $\geq$ ).

Теперь, используя сравнения чисел, мы можем записывать ветвления для ВЫЧИСЛИТЕЛЯ, например:

Если  $a < b$ , то:  
Присвоить  $c$  значение  $a$ .  
Иначе:  
Присвоить  $c$  значение  $b$ .  
Сообщить  $c$ .  
Конец ветвления.

Легко понять, что этот алгоритм служит для определения наименьшего из двух чисел  $a$  и  $b$ .

Еще пример. Составим для ВЫЧИСЛИТЕЛЯ алгоритм нахождения значения функции  $|2x-3|$ . ВЫЧИСЛИТЕЛЬ должен запросить у нас число  $x$ , вычислить значение  $y$ , равное  $2x-3$ , а затем, если  $y < 0$ , присвоить  $y$  значение  $-y$ .

Запросить  $x$ .  
Присвоить  $y$  значение  $2x-3$ .  
Если  $y < 0$ , то:  
Присвоить  $y$  значение  $-y$ .  
Конец ветвления.  
Сообщить  $y$ .

Обратимся теперь ко второму нашему исполнителю — ЧЕРТЕЖНИКУ. Раньше мы предполагали, что он движется по неограниченному листу бумаги. Это, конечно, нереально. Будем считать, что лист бумаги ограничен. Тогда ЧЕРТЕЖНИК должен уметь проверять, не находится ли он в "опасной" близости к краю листа (на самом краю он находиться не может). Опишем это действие более четко: ЧЕРТЕЖНИК может определять, находится ли край листа на расстоянии одного шага впереди него. В развилках будем писать так:

Если впереди край, то:

или так:

Если впереди не край, то:

Для примера разберем следующую задачу.

**Задача.** ЧЕРТЕЖНИК находится у края листа, но не в углу, причем ЧЕРТЕЖНИКУ неизвестно, около какого края (левого, правого, верхнего или нижнего) и в каком из четырех возможных положений (рис. 26) он находится. Составить алгоритм, с помощью которого ЧЕРТЕЖНИК отойдет от края на один шаг.

Сразу заметим, что совершенно неважно, около какого края стоит ЧЕРТЕЖНИК: поверните лист бумаги, и левый край окажется нижним, правым или верхним. А вот положение ЧЕРТЕЖНИКА относительно края существенно. Если он находится в положении а, то ему достаточно прыгнуть; в остальных положениях ему надо сначала несколько раз повернуть налево, чтобы стать в положение а, и только затем прыгнуть. Понятно, что для каждого исходного положения написать свой (линейный) алгоритм несложно. Од-

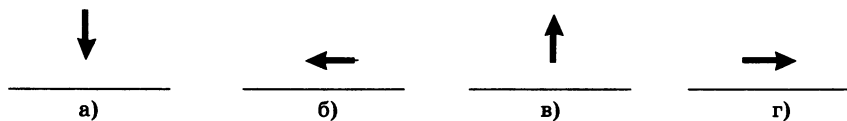
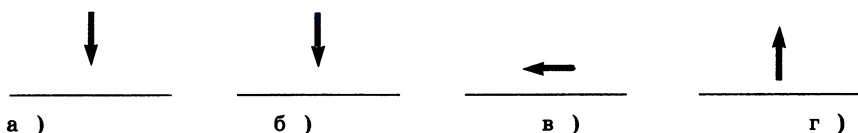


Рис.26

нако алгоритм, "обслуживающий", скажем, положение  $a$ , не годится для остальных положений. Конечно, можно было бы попытаться составить из этих четырех алгоритмов единый алгоритм для всех случаев... Но может ли ЧЕРТЕЖНИК сразу узнать, в каком из четырех положений он находится? Увы, не всегда! Это возможно лишь в положении  $a$  — достаточно проверить условие "впереди край". Возникает идея: сначала составить алгоритм, приводящий ЧЕРТЕЖНИКА из любого исходного положения в положение  $a$ . Один из возможных алгоритмов таков: поворачиваем ЧЕРТЕЖНИКА налево, пока впереди него не окажется более трех поворотов: из положения  $a$  — 0 поворотов, из положения  $b$  — 1 поворот, из положения  $c$  — 2 поворота, из положения  $d$  — 3 поворота. Запишем этот алгоритм, демонстрируя движение ЧЕРТЕЖНИКА: первый рисунок показывает движение ЧЕРТЕЖНИКА из первого положения, второй — из второго положения, третий — из третьего, четвертый — из четвертого.

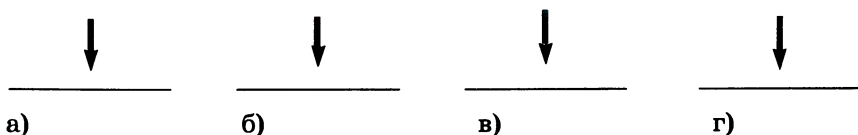
Если впереди не край, то:  
Повернуть налево.  
Конец ветвления.



Если впереди не край, то:  
Повернуть налево.  
Конец ветвления.



Если впереди не край, то:  
Повернуть налево.  
Конец ветвления.



Теперь несложно завершить составление алгоритма отхода от края, дописав еще три строки:

Повернуть налево.  
Повернуть налево.  
Прыгнуть.

В языке Бейсик ветвления записываются так:

IF <условие> THEN <действие>  
(ветвление в неполной форме) или так:

IF <условие> THEN <действие> ELSE <действие>  
(ветвление в полной форме).

Как видите, слову "если" соответствует "IF", слову "то" — "THEN", слову "иначе" — "ELSE", а указатель конца ветвления отсутствует: дело в том, что в Бейсике запись ветвления должна размещаться в одной строке.

## ? Вопросы

1. Какие условия умеет проверять ВYЧИСЛИТЕЛЬ?
2. Какие условия умеет проверять ЧЕРТЕЖНИК?



## Задания для самостоятельного выполнения

1. Что нарисует ЧЕРТЕЖНИК, выполнив следующий алгоритм из исходных положений, показанных на рисунке 27?

Если впереди край, то:

Повернуть налево.

Конец ветвления.

Если впереди край, то:

Повернуть налево.

Сделать шаг.

Повернуть налево.

Сделать шаг.

Иначе:

Сделать шаг.

Повернуть налево.

Сделать шаг.

Повернуть налево.

Сделать шаг.

Повернуть налево.

Сделать шаг.

Конец ветвления.

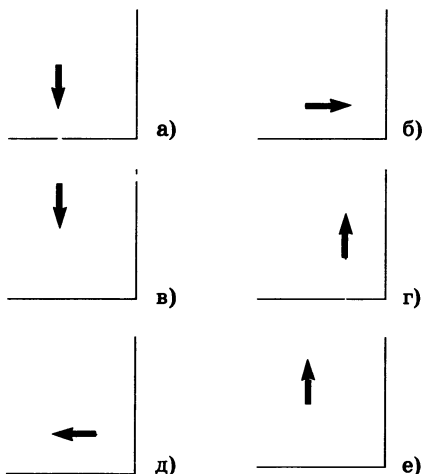


Рис. 27

2. Какие значения примут переменные  $a$  и  $b$  после выполнения следующих алгоритмов? Укажите действия, которые не будут совершаться при выполнении этих алгоритмов.

- а) Присвоить  $a$  значение 13.  
Присвоить  $b$  значение 12.  
Если  $a < b + 1$ , то:  
Присвоить  $b$  значение  $b - a$ .

Присвоить  $a$  значение  $a-b$ .

Иначе:

Присвоить  $a$  значение  $a-b$ .

Присвоить  $b$  значение  $b-a$ .

Конец ветвления.

Присвоить  $a$  значение  $ab$ .

Если  $a > b$ , то:

Присвоить  $b$  значение  $a-b$ .

Конец ветвления.

б) Присвоить  $a$  значение 13.

Присвоить  $b$  значение 12.

Присвоить  $a$  значение  $ab$ .

Если  $a > b$ , то:

Присвоить  $b$  значение  $a-b$ .

Конец ветвления.

Если  $b > a-1$ , то:

Присвоить  $b$  значение  $b-a$ .

Присвоить  $a$  значение  $a-b$ .

Иначе:

Присвоить  $a$  значение  $a-b$ .

Присвоить  $b$  значение  $b-a$ .

Конец ветвления.

3. Составьте алгоритм, с помощью которого ЧЕРТЕЖНИК отойдет от края листа, если первоначально он находится в одном из следующих положений:

а) слева от ЧЕРТЕЖНИКА находится край листа, и ЧЕРТЕЖНИК не находится в углу листа;

б) ЧЕРТЕЖНИК находится в углу листа, и впереди него — край листа (где другой край — справа или слева — неизвестно);

в) ЧЕРТЕЖНИК находится в углу листа, и слева от него — край листа (где второй край — впереди или сзади — неизвестно);

г) ЧЕРТЕЖНИК находится в углу листа, и неизвестно, с какой стороны от него края листа;

д) ЧЕРТЕЖНИК находится у края листа, но неизвестно, в углу или нет.

4. В записи алгоритма вычисления значения выражения

$$(x^2 - 5x + 5)/(x^6 - 4x^2 + 3)$$

злоумышленник одно действие поставил не на свое место. Вот как стал выглядеть алгоритм:

Запросить  $x$ .

Если  $x^6 - 4x^2 + 3 = 0$ , то:

Сообщить "При таком  $x$  значение выражения не определено".

Иначе:

Присвоить  $y$  значение  $(x^2 - 5x + 5)/(x^6 - 4x^2 + 3)$ .

Конец ветвления.

Сообщить "Я нашел значение  $y$ . Вот оно:".

Сообщить  $y$ .

Верните действие на свое место.

5. Составьте для ВЫЧИСЛИТЕЛЯ алгоритм нахождения модуля выражения  $ax^2 + bx + c$  при заданных значениях  $a$ ,  $b$ ,  $c$  и  $x$ .

6. Даны координаты двух точек в прямоугольной системе координат. Составьте для ВЫЧИСЛИТЕЛЯ алгоритм, с помощью которого он сможет определить, какая из этих точек находится дальше:

а) от начала координат;

б) от окружности данного радиуса с центром в начале координат.

7. На координатной плоскости нарисованы два круга  $K1$  и  $K2$  и два прямоугольника  $P1$  и  $P2$  со сторонами, параллельными осям координат. Даны координаты центров каждого из кругов и их радиусы, а также координаты левой нижней и правой верхней вершин каждого прямоугольника. Составить для ВЫЧИСЛИТЕЛЯ алгоритмы нахождения площади пересечения:

а) кругов  $K1$  и  $K2$ ;

б)\* прямоугольников  $P1$  и  $P2$ ;

в)\* круга  $K1$  и прямоугольника  $P1$ .

## § 14. ЗАДАЧА О ДОРОЖНОМ ПРОИСШЕСТВИИ

Каждый шофер знает: в пути часто возникают ситуации, когда приходится делать выбор. То "кирпич" помещает проехать (куда повернуть — налево или направо?), то с машиной что-нибудь случится (чинить самому или вызывать техпомощь?), то черная кошка перебежит дорогу (ехать дальше или нет?). А бывает и так:

**Задача.** На узкой улице внезапно заклинило тормоза у "Волги". В результате немедленно образовалась "пробка". Шофер стоявшего сзади грузовика, у которого лопнуло терпение, предложил помочь убрать "Волгу" с проезжей части дороги, оттащив ее на обочину с помощью троса. Удастся ли оттащить "Волгу"?

Составим математическую модель и алгоритм решения этой задачи. Сначала выделим те предположения, на которых будет основываться наша модель. Для простоты будем считать, что поверхность дороги горизонтальна. Главное — сдвинуть "Волгу" с места, поскольку из физики известно, что трение покоя всегда больше трения скольжения. Мы

будем предполагать, что сила трения равна произведению коэффициента трения на вес "Волги". Но сила трения — не единственное препятствие. Помешать могут и недостаточная мощность грузовика, и слишком малая прочность троса. Нетрудно понять, что перечисленные препятствия — самые главные. Если "не заметить" какое-либо из них, то получится гораздо менее достоверная модель задачи.

С другой стороны, в модели можно учесть и менее существенные сведения о происшествии (время суток, силу и направление ветра, личные качества водителя "Волги" и т. д.). Тогда получится гораздо более сложная модель, причем усложнение модели скорее всего "не окупится" повышением ее точности (желающие могут в этом убедиться самостоятельно).

Вот перечень наших предположений:

1) поверхность дороги горизонтальна;

2) трение покоя не меньше трения скольжения;

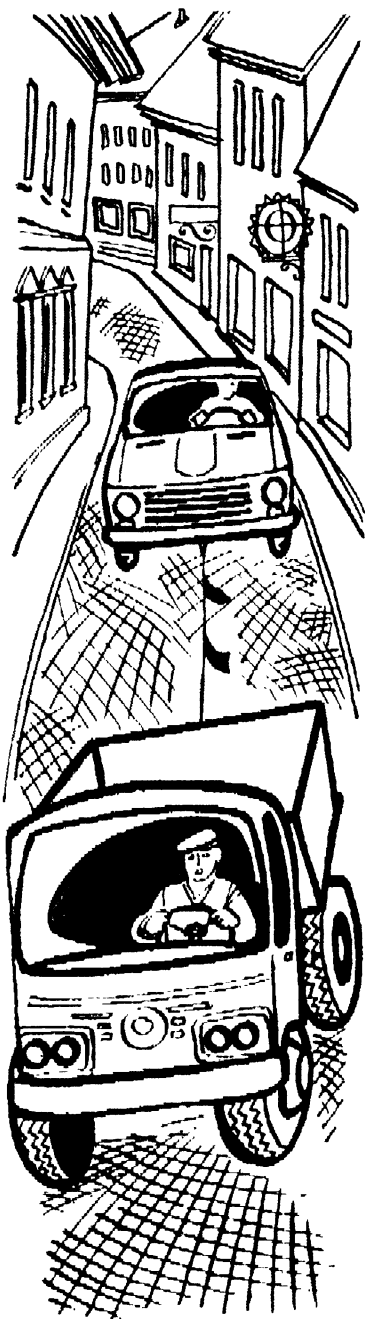
3) сила трения покоя равна произведению веса "Волги" на коэффициент трения покоя;

4) из всех сил, действующих на "Волгу", будем рассматривать только две: силу тяги и силу трения;

5) сила тяги грузовика ограничена;

6) усилие, которое может выдерживать трос, ограничено.

Теперь определим, что следует считать исходными данными и что требуется найти. Понятно, что в силу сделанных только что предположений исходными данными являются вес



"Волги"  $P$ , коэффициент трения  $K$ , максимальная сила  $F$ , с которой грузовик может тянуть "Волгу", максимальная нагрузка  $R$ , которую может выдержать трос. Требуется определить, сдвинется ли "Волга" с места. Итак, результатом является одно из сообщений "Сдвинется" или "Не сдвинется".

Запишем теперь математические соотношения, связывающие результат с исходными данными. Легко понять, что вполне достаточно тянуть "Волгу" с силой чуть-чуть большей, чем сила трения, действующая на "Волгу". Обозначим силу трения через  $Q$ . Если эта сила больше, чем наибольшая сила тяги грузовика  $F$ , то сдвинуть "Волгу" с места не удастся. Значит, сила  $Q$  должна быть меньше  $F$ . Если  $Q$  окажется больше, чем усилие, которое может выдержать трос (сила  $R$ ), то трос лопнет, а "Волга" опять останется на месте. Значит, сила  $Q$  должна быть меньше  $R$ . (Подумайте, почему сила  $Q$  не может быть в точности равна  $F$  или  $R$ .) Итак, чтобы "Волга" сдвинулась с места, нужно, чтобы  $Q$  была меньше и  $R$ , и  $F$ . Мы можем, таким образом, записать связи между исходными данными и результатами:

сила трения  $Q = K \cdot P$ ;

если  $Q < R$  и  $Q < F$ , то "Волга" сдвинется с места, в противном случае "Волга" с места не сдвинется.

Попробуем теперь составить алгоритм:

**Запросить вес "Волги"  $P$ , коэффициент трения  $K$ , максимальную силу тяги  $F$ , максимальную нагрузку  $R$  на трос.**

**Присвоить силе трения  $Q$  значение  $K \cdot P$ .**

**Если  $Q < R$  и  $Q < F$ , то:**

**Сообщить "Сдвинется с места".**

**Иначе:**

**Сообщить "Не сдвинется с места".**

**Конец ветвления.**

Однако, внимательно посмотрев на этот алгоритм, легко убедиться, что ВЫЧИСЛИТЕЛЬ выполнить его не в состоянии. В самом деле, при формулировке математической модели (поэтому и в записи ветвления) мы записали условие " $Q < R$  и  $Q < F$ ", а проверка такого условия не является допустимой для ВЫЧИСЛИТЕЛЯ. За один раз он может проверить лишь одно соотношение. Значит, мы должны изменить модель, чтобы избежать проверки сразу двух соотношений.

Поясним на примере, как это можно сделать. Допустим, в ваш класс пришел новый ученик и вы хотите узнать, меньше ли он всех ростом. Для этого вы не станете последовательно сравнивать его со всеми учениками класса, а сравните его с самым маленьким по росту учеником — с

тем, кто стоит на левом фланге при построении на уроках физкультуры.

Мы поступим точно так же: сначала найдем наименьшее из чисел  $R$  и  $F$ , обозначив это число буквой  $Z$ , а затем сравним это число с  $Q$ . Соотношения в нашей модели примут вид:

сила трения  $Q = K \cdot P$ ;

$Z$  — минимальное из чисел  $R$  и  $F$ ;

если  $Q < Z$ , то "Волга" сдвинется с места,

в противном случае "Волга" с места не сдвинется.

Исправим алгоритм решения задачи. Мы уже записывали алгоритм нахождения минимального из двух чисел. Включим его как составную часть в наш алгоритм:

Запросить вес "Волги"  $P$ , коэффициент трения  $K$ , максимальную силу тяги  $F$ , максимальную нагрузку  $R$  на трос.

Если  $R < F$ , то:

Присвоить  $Z$  значение  $R$ .

Иначе:

Присвоить  $Z$  значение  $F$ .

Конец ветвления.

Присвоить силе трения  $Q$  значение  $K \cdot P$ .

Если  $Q < Z$ , то:

Сообщить "Сдвинется с места".

Иначе:

Сообщить "Не сдвинется с места".

Конец ветвления.

## Задания для самостоятельного выполнения

1. Дан алгоритм:

Запросить  $a$  и  $b$ .

Присвоить  $c$  значение  $a^2 + b^2$ .

Присвоить  $d$  значение  $ab:(a-b)$ .

Присвоить  $s$  значение  $c - 2d$ .

Сообщить  $s$ .

Что сообщит ВЫЧИСЛИТЕЛЬ, выполнив этот алгоритм при  $a=4$  и  $b=2$ ? Сможет ли ВЫЧИСЛИТЕЛЬ выполнить алгоритм при  $a=3$  и  $b=3$ ? Какими действиями надо дополнить этот алгоритм, чтобы ВЫЧИСЛИТЕЛЬ мог выполнить его при любых значениях  $a$  и  $b$ ?

2. Составьте для ВЫЧИСЛИТЕЛЯ алгоритм, с помощью которого он определит, можно ли поместить:

а) круг данного радиуса в квадрат с данной стороной;

б) квадрат с данной стороной в круг данного радиуса.

3. На плоскости нарисованы три точки. Известны расстояния между каждыми двумя точками. Составьте для

**ВЫЧИСЛИТЕЛЯ** алгоритм, с помощью которого он сможет определить, лежат ли эти точки на одной прямой.

4. Составьте для **ВЫЧИСЛИТЕЛЯ** алгоритм решения линейного уравнения  $ax+b=0$  при заданных значениях  $a$  и  $b$  (учтите, что и  $a$ , и  $b$  могут равняться 0).

5. Составьте математическую модель и алгоритм решения физической задачи, разобранной в данном параграфе, считая участок дороги не горизонтальным.

6. Составьте математические модели и алгоритмы решения следующих задач по химии:

а) В смеси водорода и кислорода произошла реакция образования воды. Какие вещества и в каком количестве получились в результате реакции?

б) В колбу с едким натром добавили некоторое количество 15%-го раствора соляной кислоты. Какие вещества и в каком количестве получились в результате реакции?

7°. а) В киоск "Мороженое" завезли неограниченное количество мороженого. Составьте математическую модель и алгоритм (для **ВЫЧИСЛИТЕЛЯ**) работы продавщицы мороженого по обслуживанию одного покупателя.

б) На уроке математики учитель захотел проверить, хорошо ли ученики знают таблицу умножения. Составьте математическую модель и алгоритм (для **ВЫЧИСЛИТЕЛЯ**) опроса одного ученика.

Многие профессиональные программисты считают, что каждая программа — это имитация одного исполнителя с помощью другого исполнителя. Согласны ли вы с этим тезисом?



## Лабораторная работа 7

### РЕШЕНИЕ ЗАДАЧ С ИСПОЛЬЗОВАНИЕМ РАЗВЕТВЛЯЮЩИХСЯ АЛГОРИТМОВ

На предыдущих уроках вы составили несколько разветвляющихся алгоритмов для **ЧЕРТЕЖНИКА** и **ВЫЧИСЛИТЕЛЯ**. Сейчас вы посмотрите, как ЭВМ будет их исполнять. А заодно проверите, насколько точно вам удалось выразить в алгоритме то, что задумали. Ведь человеку свойственно ошибаться: обычно человек думает, что составил правильную и совершенно точную инструкцию, и обычно эта инструкция оказывается неточной и неправильной. Машина — лучший контролер. Она не ищет скрытый смысл программы, не читает между строк, а четко и педантично выполняет то, что ей предписано.

Пока ассистент настраивает ЭВМ на работу в роли **ЧЕРТЕЖНИКА**, найдите в своих тетрадях алгоритм, выполнив который **ЧЕРТЕЖНИК** отойдет от края листа.

Допустимые действия **ЧЕРТЕЖНИКА**, которые вы изучили, для ЭВМ записываются точно так же: **СДЕЛАТЬ ШАГ, ПОВЕРНУТЬ НАЛЕВО, ПРЫГНУТЬ**. В развилках нужно писать **ЕСЛИ ВПЕРЕДИ КРАЙ, ТО:** или **ЕСЛИ ВПЕРЕДИ НЕ КРАЙ, ТО:**.

Кроме того, при имитации **ЧЕРТЕЖНИКА** на ЭВМ оказалось необходимым добавить ему еще одно допустимое действие **НАЧАТЬ РАБОТУ**. По этой команде ЭВМ рисует на экране лист бумаги и просит вас установить **ЧЕРТЕЖНИКА** в исходное положение (нужно подвести **ЧЕРТЕЖНИКА** к выбранной вами точке листа, задать его направление и нажать на клавишу "ПЕРЕВОД СТРОКИ").

Итак, запишите в начале вашего алгоритма действие **НАЧАТЬ РАБОТУ**, занумеруйте строки алгоритма и введите получившуюся программу в ЭВМ. Вы еще не забыли, как это делается?

Запустите программу несколько раз при разных начальных положениях **ЧЕРТЕЖНИКА**. Всегда ли, выполнив вашу программу, **ЧЕРТЕЖНИК** отходит от края листа? Уверены ли вы, что проведенных испытаний достаточно? Можно ли теперь с чистой совестью утверждать, что программа написана верно? Для каких начальных положений **ЧЕРТЕЖНИКА** надо запустить программу, чтобы убедиться в ее правильности?

Теперь посмотрим, как **ВЫЧИСЛИТЕЛЬ** выполняет программы с ветвлениями. Помните "дорожно-транспортную" задачу? Вот она:

**Задача.** На узкой улице заклинило тормоза у "Волги". Шофер грузовика предложил помочь убрать "Волгу" с проезжей части дороги, оттащив ее на обочину с помощью троса. Удастся ли оттащить "Волгу"?

В свое время вы составили математическую модель и алгоритм решения этой задачи. Исходными данными в ней являются вес "Волги"  $P$ , коэффициент трения  $K$ , максимальная сила тяги  $F$ , максимальная нагрузка  $R$  на трос. Алгоритм решения этой задачи выглядит так:

Запросить  $P, K, F, R$ .

Если  $R < F$ , то:

Присвоить  $Z$  значение  $R$ .

Иначе:

Присвоить  $Z$  значение  $F$ .

Конец ветвления.

Присвоить  $Q$  значение  $K \cdot P$ .

Если  $Q < Z$ , то:

Сообщить "Сдвинется с места".

Иначе:

Сообщить "Не сдвинется с места".

Конец ветвления.

Запишите программу, соответствующую этому алгоритму.

На язык Бейсик этот алгоритм переводится так:

```
1 INPUT P,K,F,R
2 IF R<F THEN LET Z=R ELSE LET Z=F
3 LET Q=K*P
4 IF Q<Z THEN PRINT "Сдвинется с места"
  ELSE PRINT "Не сдвинется с места"
```

Введите в ЭВМ программу и запустите ее, приняв вес "Волги" равным 20 000 Н; коэффициент трения резины по асфальту равным 0,7; силу тяги грузовика равной 25 000 Н, максимальное усилие, которое выдерживает трос, равным 13 000 Н. Если программа и данные введены правильно, то ВЫЧИСЛИТЕЛЬ сообщит, что "Волга" не сдвинется с места. Почему же она не сдвинется? Потому ли, что мощности грузовика не хватит, или потому, что трос лопнет? ВЫЧИСЛИТЕЛЬ ответа не дает. Впрочем, мы его и не "просим". Так давайте "попросим"! Для этого придется дополнить математическую модель. Можно, скажем, предусмотреть в ней еще один результат — сообщение о том, что трос порвется. Это сообщение должно появляться, если сила трения  $Q$  окажется больше максимального усилия  $R$ , выдерживаемого тросом.

Изменения модели всегда приводят к изменениям в алгоритме. В данной ситуации надо дополнить алгоритм ветвлением:

```
Если  $Q > R$ , то:
Сообщить "Трос порвется".
Конец ветвления.
```

Ясно, что это ветвление надо записать после сообщения о том, что "Волга" останется на месте. Можно, например, просто дописать его в конце прежнего алгоритма:

```
Запросить  $P, K, F, R$ .
Если  $R < F$ , то:
Присвоить  $Z$  значение  $R$ .
Иначе:
Присвоить  $Z$  значение  $F$ .
Конец ветвления.
Присвоить  $Q$  значение  $K \cdot P$ .
Если  $Q < Z$ , то:
Сообщить "Сдвинется с места".
Иначе:
Сообщить "Не сдвинется с места".
Конец ветвления.
Если  $Q > R$ , то:
Сообщить "Трос порвется".
Конец ветвления.
```

Теперь если по расчетам **ВЫЧИСЛИТЕЛЯ** трос не выдержит нагрузки, то **ВЫЧИСЛИТЕЛЬ** не станет об этом умалчивать. Если же **ВЫЧИСЛИТЕЛЬ** сообщит, что "Волга" останется на месте и больше ничего не скажет, то все ясно — не хватает мощности грузовика.

Внесите изменения в программу и проведите расчет с теми же исходными данными. Оказывается, в прошлый раз "Волга" не смогла бы сдвинуться с места из-за того, что лопнул бы трос. (Как видите, ЭВМ предотвратила порчу казенного имущества.) Значит, надо взять более прочный трос. Запустите программу несколько раз, постепенно увеличивая  $R$ , и определите минимальную допустимую прочность троса. А теперь проведите такие же расчеты, меняя значения коэффициента трения (он зависит от состояния дороги и шин).

Только что выполненные расчеты еще раз демонстрируют главное преимущество использования ЭВМ: физический эксперимент заменяется компьютерным экспериментом. Одно дело тянуть "Волгу" по асфальту и смотреть, лопнет ли дефицитный трос, и совсем другое дело — запускать программу. Для физического эксперимента вам нужно брать разные автомобили и разные тросы (где их столько взять?) и каждый раз с новой силой тянуть машины по дорогам с разными типами покрытия. А на ЭВМ вы можете имитировать эксперимент при любых значениях  $P$ ,  $K$ ,  $F$ ,  $R$ .

Как мы уже говорили, важно и другое: компьютерные эксперименты позволяют определить границы применимости построенной математической модели, т. е. выяснить, для каких значений исходных данных результаты согласуются с физической реальностью. Посмотрим еще раз на нашу модель. Как вы думаете, любые ли числа можно подставлять вместо  $P$ ,  $F$ ,  $K$ ,  $R$ ? Введите-ка в качестве  $F$  отрицательное число. Какое бы отрицательное число вы ни подставили, "Волга" не сдвинется с места. Не правда ли, странный результат? Выходит, что если тянуть "Волгу" в одну сторону, то она движется, а если с такой же силой тянуть ее в противоположном направлении, то стоит на месте. Каждый скажет: такого быть не может. А все дело в том, что наша модель отражает реальный процесс далеко не при всех значениях аргументов. Так, при ее построении мы молчаливо предполагали, что сила  $F$  не может быть отрицательной. Для остальных переменных  $P$ ,  $K$ ,  $R$  также есть естественные границы: их значения должны быть положительными, а  $K$ , кроме того, не может превосходить 1.

Итак, наш компьютерный эксперимент показал необходимость уточнения математической модели. В нее надо вне-

сти следующие ограничения на исходные данные:  $F > 0$ ,  $P > 0$ ,  $K > 0$ ,  $K < 1$ ,  $R > 0$ .

Соответствующим образом надо изменить алгоритм решения задачи: после запроса начальных данных надо проверить, удовлетворяют ли они указанным ограничениям. Если данные не удовлетворяют ограничениям, то ВЫЧИСЛИТЕЛЬ должен прекратить выполнение алгоритма. Для этого у него есть допустимое действие "Стоп".

Запросить  $P$ ,  $F$ ,  $K$ ,  $R$ .

Если  $P \leq 0$ , то:

Сообщить "Вес должен быть положительным".

Стоп.

Конец ветвления.

Если  $F \leq 0$ , то:

Сообщить "Нельзя вводить отрицательные значения силы".

Стоп.

Конец ветвления.

Если  $K \leq 0$ , то:

Сообщить "Коэффициент трения должен быть положительным".

Стоп.

Конец ветвления.

Если  $K > 1$ , то:

Сообщить "Коэффициент трения не может превосходить 1".

Стоп.

Конец ветвления.

Если  $R \leq 0$ , то:

Сообщить "Нагрузка на трос не может быть отрицательной".

Стоп.

Конец ветвления.

Если  $R < F$ , то:

Присвоить  $Z$  значение  $R$ .

Иначе:

Присвоить  $Z$  значение  $F$ .

Конец ветвления.

Присвоить  $Q$  значение  $P \cdot K$ .

Если  $Q < Z$ , то:

Сообщить "Сдвинется с места".

Иначе:

Сообщить "Не сдвинется с места".

Конец ветвления.

Если  $Q > R$ , то:

Сообщить "Трос порвется".

Конец ветвления.

Запишите соответствующую этому алгоритму программу и введите ее в ЭВМ. При этом можно воспользоваться предыдущей программой, вызвав ее на экран и изменив номера строк. Запустите программу при недопустимых значениях аргументов и убедитесь в том, что ЭВМ не станет проводить вычисления для этих значений.

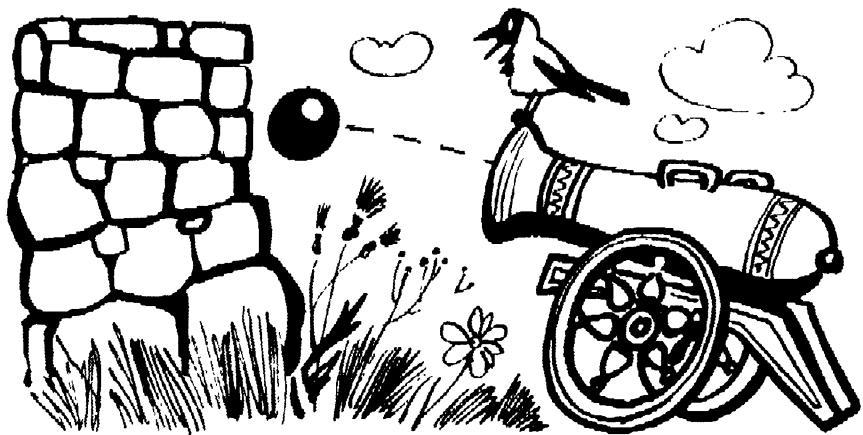
## § 15. АРТИЛЛЕРИЙСКАЯ ЗАДАЧА

Сразу предупредим, что этот параграф не надо воспринимать как пропаганду войны. Просто давайте постреляем.

**З а д а ч а.** На заданном расстоянии от пушки находится стена. Известны угол наклона пушки и начальная скорость снаряда. Попадет ли снаряд в стену?

Сначала, как обычно, выскажем упрощающие предположения. Снаряд считаем материальной точкой, сопротивлением воздуха и размерами пушки пренебрегаем. Разумеется, мы считаем также, что справедливы полученные из физических законов соотношения, описывающие движение материальной точки.

Из условия задачи видно, каковы в ней исходные данные: это угол наклона пушки ( $A$  градусов), начальная скорость ( $v$  м/с), расстояние до стены ( $S$  м) и высота стены ( $h$  м). Результатом является одно из сообщений "попал" или "не попал". Начальные данные подчиняются естественным ограничениям:  $0^\circ < A < 90^\circ$ ,  $S > 0$ ,  $h > 0$ ,  $0 < v < 1000$  м/с (подумайте, как можно обосновать эти ограничения). Теперь нам надо записать на языке математики тот факт, что снаряд попал в стену. Для этого надо найти высоту  $L$  снаряда на расстоянии  $S$  от пушки: ведь попадание снаряда в стену означает, что  $L$  находится в пределах от 0 до  $h$ .



Итак, нам нужна формула для определения  $L$ . Такая формула давно выведена в физике. Вот она:

$$L = Stg(\pi A/180) - \frac{gS^2}{2v^2 \cos^2(\pi A/180)}.$$

В этой формуле, кроме уже введенных ранее обозначений, присутствуют символы  $\pi$  и  $g$ . С  $\pi$  вы давно знакомы и знаете, что  $\pi \approx 3,1415$ . А буквой  $g$  в физике обозначают так называемое ускорение свободного падения. Вблизи земной поверхности значение  $g$  принято считать постоянным и равным  $\approx 9,8 \text{ м/с}^2$ . Такие формулы вы научитесь выводить во втором полугодии девятого класса. Впрочем, часто бывает и так: кто-нибудь выведет формулу, а пользуются ею все, даже не задумываясь, откуда она взялась (самый простой пример — формула площади круга). В нашем случае важно не столько само по себе значение  $L$ , сколько ответ на вопрос, попадает ли  $L$  в промежуток от 0 до  $h$ . Заметим, что значение  $L$ , вычисленное по нашей формуле, может оказаться и отрицательным. Это означает, что снаряд до стены не долетит. Если же  $L$  окажется больше  $h$ , то снаряд перелетит через стену. Вот модель и построена.

Составим алгоритм для ВЫЧИСЛИТЕЛЯ:

Сообщить "Введите начальную скорость снаряда, расстояние до стены, высоту стены".

Запросить  $v$ ,  $S$ ,  $h$ .

Сообщить "Введите угол наклона ствола пушки".

Запросить  $A$ .

Если  $v \leq 0$ , то:

Сообщить "Скорость должна быть положительной".

Стоп.

Конец ветвления.

Если  $v \geq 1000$ , то:

Сообщить "Скорость слишком велика".

Стоп.

Конец ветвления.

Если  $S \leq 0$ , то:

Сообщить "Расстояние должно быть положительным".

Стоп.

Конец ветвления.

Если  $h \leq 0$ , то:

Сообщить "Высота стены должна быть положительной".

Стоп.

Конец ветвления.

Если  $A \leq 0$ , то:

Сообщить "Пушка не должна стрелять в землю".

Стоп.

Конец ветвления.

Если  $A \geq 90$ , то:

Сообщить "Пушка направлена не в сторону стены".

Стоп.

Конец ветвления.

Присвоить  $L$  значение  $\text{Stg}(\pi A/180) - \frac{gS^2}{2v^2 \cos^2(\pi A/180)}$ .

Если  $L > h$ , то:

Сообщить "Снаряд не попал в стену...".

Стоп.

Конец ветвления.

Если  $L < 0$ , то:

Сообщить "Снаряд не попал в стену...".

Иначе:

Сообщить "Попал!!!".

Конец ветвления.

Стоп.

Итак, алгоритм построен. Ну а стрельба, конечно, возможна только в лабораторных условиях.



### Задание для самостоятельного выполнения

Составьте компьютерную модель следующей задачи и соответствующий алгоритм для ВЫЧИСЛИТЕЛЯ.

**З а д а ч а.** Пушка стреляет в направлении движения градового облака в тот момент, когда оно проплывает над пушкой. Попадет ли снаряд в облако?



### Лабораторная работа 8 МЕТОД ДЕЛЕНИЯ ПОПОЛАМ

Надеемся, что вам было нетрудно записать и ввести в ЭВМ программу, соответствующую алгоритму решения артиллерийской задачи. Заставьте ВЫЧИСЛИТЕЛЯ выполнить эту программу при следующих значениях исходных данных:  $v=200$  м/с,  $S=3000$  м,  $h=6$  м,  $A=45^\circ$ . Попал ли снаряд в стену? По нашим подсчетам, нет.

Попробуйте, меняя угол наклона ствола, произвести пристрелку пушки по цели. Для этого лучше всего использовать известный артиллерийский прием "взятие в вилку". Математики называют этот прием методом деления пополам. Заключается он в следующем. Сначала находят угол наклона ствола, при котором снаряд перелетит через стену, и угол, при котором он не долетит до стены. Эти два угла и образуют "вилку".

*Нужную точность  
всегда обеспечит вам  
метод  
деления пополам!*

Затем берут среднее значение углов, составляющих "вилку", и смотрят, куда попадет снаряд: если он попадет в стену, то стрельба прекращается, если он не попадет в стену, то образуется новая "вилка" и надо взять среднее значение между углами этой "вилки".

Чтобы воспользоваться методом деления пополам, надо знать, перелетел снаряд через стену или не долетел до нее. Однако в программе такие сообщения не предусмотрены. Измените соответствующим образом программу и с ее помощью проведите пристрелку пушки по цели.

Метод деления пополам универсален. Можно применить его, например, к решению уравнений. Рассмотрим какое-нибудь сложное уравнение, скажем, такое:

$$x^5 - x^4 - 2 = 0.$$

Вы знаете, что корень уравнения — это значение  $x$ , которое превращает уравнение в верное равенство.

Поиск этого значения  $x$  можно уподобить стрельбе по цели. Ведь мы хотим найти такое  $x$ , чтобы "попасть" в число 0. При  $x=0$  получился "недолет": левая часть равна  $-2$ . При  $x = 2$  "перелет": левая часть равна 14. Видите, мы взяли число 0 в "вилку"! Что делать дальше — вы уже знаете. Надо брать середину отрезка  $[0;2]$ , вычислять значение левой части, смотреть, получился "перелет" или "недолет", определять новую "вилку" и т. д., пока  $x$  не будет найдено с требуемой точностью, скажем, до тысячных. Попробуйте написать соответствующую программу и найти вручную хорошее приближение для  $x$ .



#### КОНСПЕКТ ГЛАВЫ 4

Алгоритмы, в которых все действия совершаются одно за другим независимо ни от каких условий, называются **линейными**. Характерная для них форма организации действий — последовательное выполнение. Если же при решении задачи необходимо принимать решения в зависимости от создавшейся ситуации, то в алгоритме нужно использовать ветвления. **Ветвление (развилка)** — это такая форма организации действий, при которой в зависимости от выполнения или невыполнения некоторого условия совершается либо одна, либо другая последовательность действий. Имеются две существенно различные формы записи ветвлений — **полная** и **неполная**. В каждой из них указывается условие, которое надо проверять, и наборы действий, которые надо исполнять при выполнении или невыполнении условия. Ясно, что проверка условия должна быть допустимым действием исполнителя.

Ветвление в полной форме записывается так:

Если  $Q$ , то:

$P_1$

$P_2$

...

$P_n$

Иначе:

$T_1$

$T_2$

...

$T_m$

Конец ветвления

Здесь  $Q$  — условие,  $P_1, P_2, \dots, P_n$  — действия, которые совершаются в случае выполнения условия  $Q$  (первая ветвь), а  $T_1, T_2, \dots, T_m$  — действия, которые совершаются, если это условие не выполнено (вторая ветвь). После исполнения всех действий ветви исполнитель переходит к исполнению действий, записанных после строки "Конец ветвления". Указатель "Конец ветвления" служит для устранения двусмысленности в алгоритме. Без него, в частности, неясно, к какому действию переходить после исполнения первой ветви ветвления.

Ветвление в неполной форме записывается следующим образом:

Если  $Q$ , то:

$P_1$

$P_2$

...

$P_n$

Конец ветвления

Здесь  $Q$  — условие,  $P_1, P_2, \dots, P_n$  — действия, которые совершаются в случае выполнения условия  $Q$ . Если же условие не выполнено, то сразу совершается действие, записанное после слов "Конец ветвления".

Для более наглядного представления алгоритмов используются так называемые **блок-схемы**. При этом каждое действие, кроме проверки условия, записывается в прямоугольник, а каждое условие — в ромб. Блок-схема 13, а соответствует последовательному выполнению действий. Блок-схема 13, б соответствует ветвлению в полной, а 13, в в неполной форме.

**ВЫЧИСЛИТЕЛЬ** умеет проверять соотношения  $<, >, =, \neq, \leq, \geq$  между значениями арифметических выражений. Примеры условий, которые умеет проверять **ВЫЧИСЛИТЕЛЬ**:  $2 < 3, A > 3, X + Y \geq Z / T - 25, 232$ . Вот пример разветвляющегося алгоритма для **ВЫЧИСЛИТЕЛЯ**:

Запросить  $A, B$ .  
Если  $\cos A < \cos B$ , то:  
Сообщить  $B$ .  
Иначе:  
Сообщить  $A$ .  
Конец ветвления.

При записи программ для ВЫЧИСЛИТЕЛЯ, имитируемого на ЭВМ, вместо знаков  $\leq$ ,  $\geq$  и  $\neq$  пишется соответственно  $=$ ,  $>=$  и  $<>$ .

Исполнитель ЧЕРТЕЖНИК умеет определять, находится ли край листа на расстоянии менее одного шага впереди него. Вот пример разветвляющегося алгоритма для ЧЕРТЕЖНИКА:

Если впереди край, то:  
Повернуть налево.  
Конец ветвления.  
Повернуть налево.  
Если впереди не край, то:  
Сделать шаг.  
Повернуть налево.  
Иначе:  
Повернуть налево.  
Конец ветвления.

При записи программ для ЧЕРТЕЖНИКА, имитируемого на ЭВМ, соответствующие условия записываются так: ВПЕРЕДИ КРАЙ или ВПЕРЕДИ НЕ КРАЙ.

Ветвления неизбежно возникают при построении алгоритмов для решения жизненных задач. Ведь любая модель применима лишь в определенных границах, порождающих те или иные ограничения на исходные данные. Проверка того, удовлетворяют ли исходные данные этим ограничениям, должна присутствовать в алгоритме. А значит, алгоритм будет содержать ветвления.

Правда, границы применимости модели нелегко указать заранее. Тогда границы выявляются в ходе вычислительного эксперимента. Одно из основных достоинств ЭВМ состоит как раз в том, что натурный эксперимент (физический, биологический, экологический и т. п.) она позволяет заменить компьютерным экспериментом. Проведение эксперимента на ЭВМ имитирует проведение натурального эксперимента, будучи при этом, как правило, гораздо дешевле и безопаснее.

## Глава 5

# ЦИКЛЫ В АЛГОРИТМАХ

---

Вы легко заметите, что алгоритмы, которые мы составляли в предыдущих главах, обладают одним общим свойством: при их выполнении каждое действие совершается один раз или вообще не совершается. В жизни, однако, часто встречаются инструкции, в которых требуется один и тот же набор действий выполнять много раз подряд: "иди, пока не придешь", "закручивай гайку, пока не завернешь ее до отказа" и т. д. Используя только развилки, такие алгоритмы не запишешь. Для этого нужна новая форма организации действий. О ней и рассказывается в данной главе.

### § 16. ЦИКЛЫ

Кто из нас не помнит поучительную историю о том, как Том Сойер по заданию тети Полли красил забор: "Вздыхая, он окунул кисть в ведро, провел ею по доске забора, повторил эту операцию, проделал ее снова..." (Марк Твен. "Приключения Тома Сойера").

Давайте составим алгоритм покраски забора. Допустим, что у нас есть малярная кисть и достаточное количество краски. Напишем, например, такую последовательность действий:

Подойти к левому краю забора.  
Покрасить одну доску.  
Шагнуть вправо на ширину доски.  
Покрасить одну доску.  
Шагнуть вправо на ширину доски.  
Покрасить одну доску.

...

Ясно, что если мы соберемся писать этот алгоритм до конца, покраску забора придется надолго отложить. Если бы мы знали, сколько досок в заборе, мы могли бы завершить составление алгоритма, приписав нужное количество строк. Однако это — долгое и однообразное занятие. Да и тетя Полли

*Любуясь Луной,  
помните:  
и Луна выполняет  
циклический  
алгоритм*

никогда не считала доски в заборе. Она просто сказала: "Будешь красить, пока забор не кончится". Сама того не зная, она воспользовалась очень распространенным способом организации действий — **циклом (повтором)**. Задание тети Полли можно записать в виде следующего алгоритма:

Подойти к левому краю забора.  
Пока забор не кончился, повторять:  
  Покрасить одну доску.  
  Шагнуть вправо на ширину доски.  
Конец цикла.  
Уйти.

Вообще если действия  $P_1, P_2, \dots, P_n$  надо повторять, пока выполняется некоторое условие  $Q$ , то мы будем использовать следующую запись:

Пока  $Q$ , повторять:  
   $P_1$   
   $P_2$   
  ...  
   $P_n$   
Конец цикла.

Эта запись означает, что исполнитель сначала проверяет, выполняется ли условие  $Q$ . Если да, то совершаются действия  $P_1, P_2, \dots, P_n$  (последовательность этих действий называют **телом цикла**), после чего условие  $Q$  проверяется снова и т. д. Если же  $Q$  не выполняется, то исполнитель переходит к действию, записанному после строки "Конец цикла". Видно, что слова "Конец цикла" играют здесь ту же роль, что и слова "Конец ветвления" в записи развилки. Может, конечно, случиться и так, что условие  $Q$  не выполнено с самого начала (в заборе вообще нет досок!). Ну что ж, тогда действия, составляющие тело цикла, не совершатся ни разу.

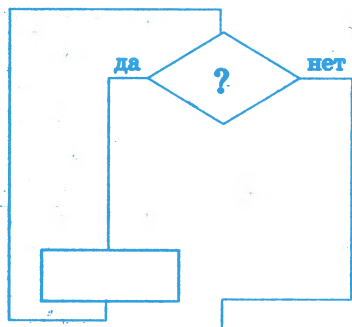


Рис. 28

Итак, **циклом (повтором)** называется такая форма организации действий, при которой одна и та же последовательность действий совершается несколько раз (или ни разу) до тех пор, пока выполняется некоторое условие.

С помощью блок-схемы цикл можно изобразить так, как показано на рисунке 28. Например, блок-схема алгоритма покраски забора выглядит так:

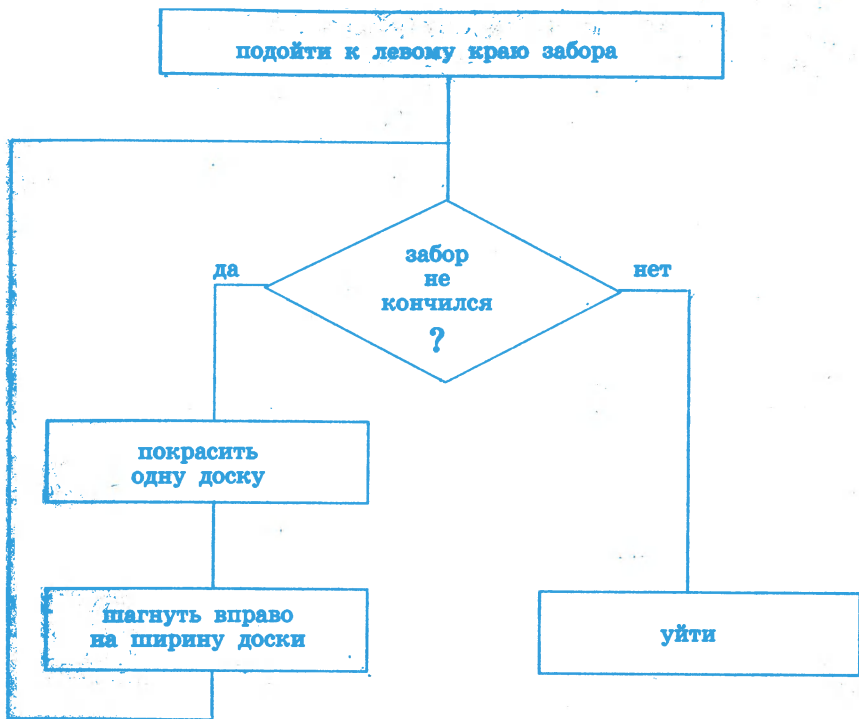


Рис. 29

## ? Вопросы

1. Какая форма организации действий называется циклом?
2. Как в алгоритмах оформляются циклы?
3. Что такое тело цикла?
4. Для чего служит указатель "Конец цикла"?
5. Как с помощью блок-схемы изображаются циклы?

## 📌 Задания для самостоятельного выполнения

1. "Приключения Тома Сойера" начинаются с того, что тетя Полли зовет Тома:

— Том!  
 Нет ответа.  
 — Том!  
 Нет ответа.  
 — Том!  
 Нет ответа...

Составьте алгоритм вызова Тома.

2. Используя циклическую форму организации действий, запишите следующий алгоритм выполнения домашнего задания по переводу текста с иностранного языка:

Найти первое предложение.  
Перевести его.  
Записать перевод.  
Найти следующее предложение.  
Перевести его.  
Записать перевод...

3. а) Во время большой перемены проголодавшийся школьник зашел в столовую с намерением поесть пирожков. Находившийся рядом злоумышленник тут же посоветовал ему воспользоваться следующим алгоритмом:

Пока не исчезло чувство голода, повторять:  
Купить пирожок.  
Конец цикла.  
Съесть пирожок.

Сумеет ли школьник поесть пирожков? Исправьте алгоритм так, чтобы школьник ушел сытым.

б) Однажды школьнику задали на дом несколько задач по математике. Придя домой, он решил сначала выполнить домашнее задание, а затем пойти гулять. Злоумышленник, который снова как назло оказался рядом, посоветовал воспользоваться следующим алгоритмом:

Пока не решены все задачи, повторять:  
Решить очередную задачу.  
Пойти гулять до ужина.  
Конец цикла.

Назавтра доверчивый школьник получил двойку за домашнее задание. Объясните почему.

4. Дан алгоритм ("решето Эратосфена"):

Написать все натуральные числа от 2 до  $n$ .  
Пока есть необведенные числа среди невычеркнутых, повторять:  
Среди невычеркнутых чисел обвести самое маленькое из необведенных.  
Из необведенных чисел вычеркнуть те, которые кратны последнему обведенному числу.  
Конец цикла.

а) Выполните алгоритм при  $n=6, 12, 1000$ . Какие числа будут обведены после окончания выполнения алгоритма в каждом из этих случаев?

б)\* Для решения какой задачи предназначен этот алгоритм? Обоснуйте ваш ответ.

5. Что будет нарисовано на экране ЭВМ после выполнения следующего алгоритма работы с графическим редактором?

Пока можно сдвинуть курсор вниз, повторять:  
Нажать на функциональную клавишу "Отрезок".  
Нажать на клавишу "Стрелка вправо".  
Нажать на функциональную клавишу "Отрезок".  
Нажать на функциональную клавишу "Отрезок".  
Нажать на клавишу "Стрелка вниз".  
Нажать на функциональную клавишу "Отрезок".  
Конец цикла.

6. Вот что написал злоумышленник на экране ЭВМ, работая с текстовым редактором:

рорпобуй пасшифпюй!

Школьник догадался, что это шифровка, и составил дешифрующий алгоритм:

Поместить курсор в начало сообщения.  
Пока курсор не вышел за пределы сообщения, повторять:  
Если курсор находится на букве "р", то:  
Нажать клавишу "п".  
Нажать клавишу "Стрелка вправо".  
Конец ветвления.  
Если курсор находится на букве "п", то:  
Нажать клавишу "р".  
Нажать клавишу "Стрелка вправо".  
Конец ветвления.  
Если курсор находится на букве "ю", то:  
Нажать клавишу "у".  
Нажать клавишу "Стрелка вправо".  
Конец ветвления.  
Нажать клавишу "Стрелка вправо".  
Конец цикла.

Какую фразу зашифровал злоумышленник? Какой способ шифровки он применил?

7. Составьте алгоритм нахождения фальшивой монеты среди настоящих монет того же достоинства с помощью чашечных весов, если известно, что фальшивая монета тяжелее настоящей.

8. Для определения количества кислоты в растворе в колбу, содержащую раствор кислоты и индикатора, по каплям добавляют щелочь (титрование раствора) до тех пор, пока индикатор не изменит цвет. Составьте алгоритм титрования.

9. (Продолжение задачи 13 из § 9.) Вслед за разведывательным дозором к той же реке подошел полк. Около берега по-прежнему плавали в лодке два мальчика. Составьте алгоритм переправы полка.

## § 17. ЦИКЛЫ И ИСПОЛНИТЕЛИ АЛГОРИТМОВ

Итак, вы познакомились с новой формой организации действий — циклом. Естественно, что и наши знакомые — исполнители **ВЫЧИСЛИТЕЛЬ** и **ЧЕРТЕЖНИК** — владеют этой замечательной формой. Начнем с **ЧЕРТЕЖНИКА**. Составим для него алгоритм, выполнив который он окажется у края листа независимо от своего начального положения:

Пока впереди не край, повторять:  
Прыгнуть.  
Конец цикла.

А вот как с помощью циклов можно записать алгоритм решения задачи, разобранный в конце § 13 (отход на один шаг от края):

Пока впереди не край, повторять:  
Налево.  
Конец цикла.  
Налево.  
Налево.  
Прыгнуть.

А теперь займемся **ВЫЧИСЛИТЕЛЕМ**. Допустим, требуется найти и сообщить все положительные члены последовательности, начинающейся с числа 10. Каждый член этой последовательности, начиная со второго, меньше предыдущего на 0,15. Алгоритм будет выглядеть так:

Присвоить  $a$  значение 10.  
Пока  $a > 0$ , повторять:  
Сообщить  $a$ .  
Присвоить  $a$  значение  $a - 0,15$ .  
Конец цикла.

В языке Бейсик цикл "пока" в зависимости от реализации на ЭВМ оформляется одним из двух способов:

10 WHILE <условие>	10 IF NOT <условие>
20 <действие>	THEN 110
30 <действие>	20 <действие>
.....	30 <действие>
90 <действие>	.....
100 WEND	90 <действие>
110 ...	100 GOTO 10
	110 ...

Следующую задачу можно назвать экологической.

**Задача.** Расположенный на берегу реки металлургический завод осуществил сброс сточных вод, в результате чего концентрация вредных веществ в реке резко увеличилась. С течением времени эта концентрация, естественно, уменьшается. Требуется сообщить, каков будет уровень загрязнения реки через сутки, двое суток и т. д. до тех пор, пока концентрация не станет меньше предельно допустимой.

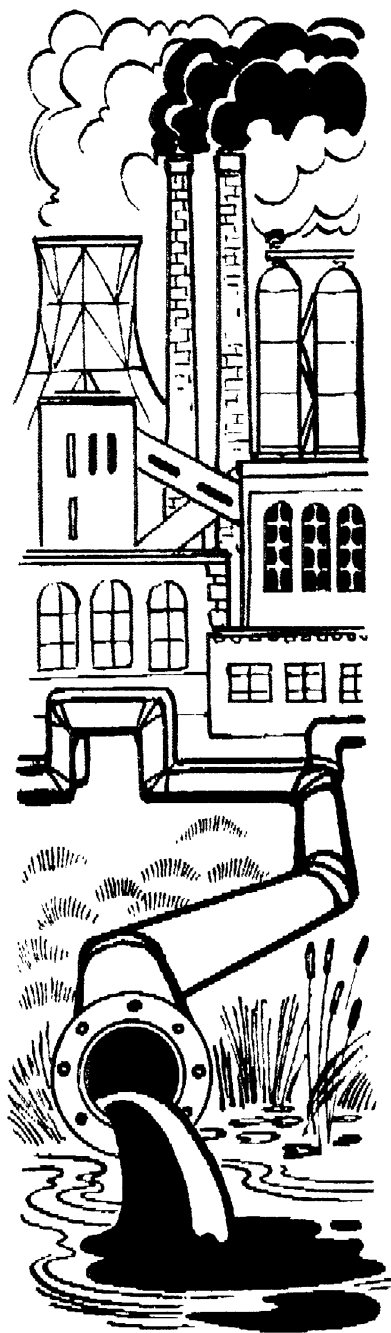
Сначала построим математическую модель изучаемого явления. Специалистами-экологами установлена следующая приближенная закономерность: в каждом конкретном случае можно указать такое число  $K > 1$ , что концентрация примесей  $C$  уменьшается в  $K$  раз за сутки. При этом коэффициент  $K$  зависит от района, где протекает река, типа примесей и т. п. Значение  $K$  можно узнать из соответствующего справочника. Эту закономерность мы примем в качестве исходного предположения для нашей математической модели.

Исходными данными будут начальная концентрация  $C$  вредных веществ в реке, предельно допустимая концентрация  $D$  и коэффициент  $K$ . Результат — последовательность значений концентрации вредных веществ через сутки, двое суток и т. д. Связь между исходными данными и результатом дается следующими соотношениями:

$$C_0 = C; \quad C_{n+1} = C_n/K,$$

где  $C_n$  — концентрация вредных веществ через  $n$  суток после сброса.

Руководствуясь этой математической моделью, составим алгоритм решения задачи:



Запросить начальное значение концентрации  $C$ , предельно допустимую концентрацию  $D$  и коэффициент  $K$ .

Присвоить номеру суток  $n$  значение 0.

Пока  $C > D$ , повторять:

Присвоить номеру суток  $n$  значение  $n+1$ .

Присвоить концентрации  $C$  значение.

Сообщить "Номер суток; концентрация:".

Сообщить  $n, -C$ .

Конец цикла.

Во время лабораторной работы 9 вы убедитесь, что первоначальное качество воды, как правило, восстанавливается годами.



### Задания для самостоятельного выполнения

1. Что нарисует ЧЕРТЕЖНИК, выполнив алгоритм из исходных положений, показанных на рисунке 30?

Пока впереди край, повторять:

Повернуть налево.

Конец цикла.

Сделать шаг.

Пока впереди край, повторять:

Повернуть налево.

Конец цикла.

Сделать шаг.

Нарисуйте блок-схему этого алгоритма.

2. Сократите запись решений задач 3, б—д к § 13, используя циклы.

3°. Составьте алгоритм, выполнив который ЧЕРТЕЖНИК дойдет до угла листа независимо от того, где он находился вначале.

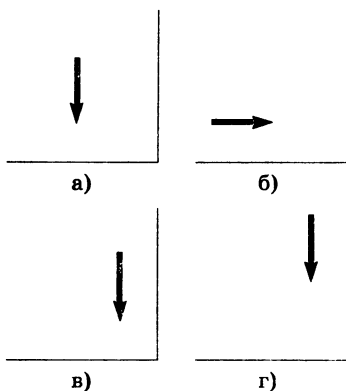


Рис. 30

4°. ЧЕРТЕЖНИК находится у края листа, но не в углу. Составьте алгоритм, выполнив который он нарисует прямоугольную рамку, отстоящую от края на расстоянии одного шага. (У к а з а н и е: воспользуйтесь решением задачи 3.)

5. ЧЕРТЕЖНИК находится в углу листа. Составьте алгоритм, выполнив который он окажется в противоположном углу листа.

6°. ЧЕРТЕЖНИК находится в углу листа бумаги. Составьте алгоритм, выполнив который ЧЕРТЕЖНИК разлунет лист на полосы шириной 1 см.

7. Чему будут равны  $a$  и  $b$  после выполнения ВЫЧИСЛИТЕЛЕМ каждого из следующих алгоритмов? Сколько раз будет выполняться тело цикла в каждом из них?

- а) Присвоить  $a$  значение  $-5$ .  
Присвоить  $b$  значение  $-2$ .  
Пока  $a+b < ab$ , повторять:  
Присвоить  $a$  значение  $2a$ .  
Присвоить  $b$  значение  $b+1$ .  
Конец цикла.
- б) Присвоить  $a$  значение  $25$ .  
Присвоить  $b$  значение  $1$ .  
Пока  $a-b > a/b$ , повторять:  
Присвоить  $a$  значение  $a+10$ .  
Присвоить  $b$  значение  $-3b$ .  
Конец цикла.
- в) Присвоить  $a$  значение  $-3$ .  
Присвоить  $b$  значение  $-1$ .  
Пока  $a < b$ , повторять:  
Если  $b < 3$ , то:  
Присвоить  $a$  значение  $a-b$ .  
Присвоить  $b$  значение  $b+2$ .  
Иначе:  
Присвоить  $a$  значение  $a+2$ .  
Присвоить  $b$  значение  $b-a$ .  
Конец ветвления.  
Конец цикла.

8. В алгоритме нахождения суммы квадратов первых ста чисел злоумышленник поменял местами несколько строк. Вот как выглядит его алгоритм:

Присвоить  $S$  значение  $0$ .  
Пока  $I \leq 100$ , повторять:  
Присвоить  $I$  значение  $1$ .  
Присвоить  $S$  значение  $S+I^2$ .  
Присвоить  $I$  значение  $I+1$ .  
Сообщить "Я нашел сумму. Вот она:".  
Сообщить  $S$ .  
Конец цикла.

Сколько раз будет ВЫЧИСЛИТЕЛЬ повторять тело цикла в этом алгоритме? Устраните путаницу в строках.

9°. Составьте для ВЫЧИСЛИТЕЛЯ алгоритм нахождения суммы чисел, обратных квадратам первых 100 натуральных чисел.

10°. Составьте для ВЫЧИСЛИТЕЛЯ алгоритм, пользуясь которым он запросит последовательно  $N$  чисел и сообщит их среднее арифметическое.

11°. Составьте для ВЫЧИСЛИТЕЛЯ алгоритм нахождения

ния остатка от деления одного натурального числа на другое.

12°. Последовательность  $a_n$  строится так:

$$a_1 = 1, a_2 = 3 \text{ и } a_n = a_{n-1} - 2a_{n-2}$$

для каждого  $n > 2$ . Составьте для ВЫЧИСЛИТЕЛЯ следующие алгоритмы:

а) алгоритм нахождения первого элемента последовательности, большего 1000;

б) алгоритм нахождения суммы первых 15 членов этой последовательности;

в) алгоритм нахождения первых десяти положительных членов этой последовательности;

г) алгоритм нахождения наибольшего из первых 20 членов последовательности.

13°. Леспромхоз ведет заготовку деловой древесины. Ее первоначальный объем на территории леспромхоза был равен 120 000 м<sup>3</sup>. Ежегодный естественный прирост составляет 5,5%. Годовой план заготовки древесины — 9500 м<sup>3</sup>. Какой объем деловой древесины на данной территории будет через год, через два года и т. д. до тех пор, пока этот объем не станет меньше минимально допустимого значения (23 000 м<sup>3</sup>)? Составьте модель для этой задачи и алгоритм ее решения.

14. а) В киоск "Мороженое" завезли неограниченное количество мороженого. Составьте модель и алгоритм (для ВЫЧИСЛИТЕЛЯ) обслуживания очереди продавщицей мороженого.

б) На уроке математики учитель захотел проверить, хорошо ли ученики усвоили таблицу умножения. Составьте модель и алгоритм (для ВЫЧИСЛИТЕЛЯ) опроса всех учеников класса.

У к а з а н и е: воспользуйтесь решением задачи 7 к параграфу 14.

15°. Составьте программу нахождения корней уравнений с точностью до 0,01 методом деления пополам, предварительно определив два числа, образующие "вилку":

а)  $x^3 - 3x + 3 = 0$ ;

б)  $2^x = 3x$ .

На лабораторной работе 8 вы решали уравнение методом деления пополам, вручную выбирая очередные концы "вилки". Теперь требуется автоматизировать этот процесс.



## Лабораторная работа 9

### РЕШЕНИЕ ЗАДАЧ С ИСПОЛЬЗОВАНИЕМ ЦИКЛИЧЕСКИХ АЛГОРИТМОВ

И вот вы снова в дисплейном классе. Сегодня вы увидите, как **ЧЕРТЕЖНИК** и **ВЫЧИСЛИТЕЛЬ** исполняют циклические алгоритмы. Начнем с **ЧЕРТЕЖНИКА**. Вы должны были приготовить для него алгоритм, выполнив который он начертит рамку на расстоянии одного шага от края листа (задача 4 из § 17; мы не сомневаемся, что вы решили эту задачу). Оформите ваш алгоритм в виде программы (не забудьте команду "НАЧАТЬ РАБОТУ"), введите программу в ЭВМ и запустите ее на исполнение. Хорошо, если **ЧЕРТЕЖНИК** и в самом деле нарисовал требуемую рамку. А что делать, если рамка не получилась или, скажем, **ЧЕРТЕЖНИК** дошел до края и дальше идти отказывается (не потому, что не хочет, а потому, что не может)? Не отчаивайтесь, мы уже говорили, что редко кому удается сразу составить правильную программу. Конечно, можно попытаться написать все заново. Но где гарантия, что она будет правильной? Лучше найти и исправить ошибки в уже созданной программе, или, как говорят программисты, отладить программу. В этом помогает специальный режим работы **ЧЕРТЕЖНИКА** — режим отладки. Нажмите на функциональную клавишу "Отладка" и запустите программу. Работая по программе в этом режиме, ЭВМ печатает на экране команды, которые она выполняет, и останавливается после исполнения каждой команды. Если вы убедились, что все правильно, то для продолжения работы надо нажать на клавишу **ПЕРЕВОД СТРОКИ**. Если же вы обнаружили ошибку, прервите работу программы (учитель покажет клавишу, на которую для этого надо нажать) и, исправив ошибку, снова запустите программу и т. д., пока не исправите все ошибки.

А теперь заставим поработать **ВЫЧИСЛИТЕЛЯ**. Для него на теоретическом занятии вы составили алгоритм решения экологической задачи (см. § 17). Введите соответствующую программу в ЭВМ и запустите ее при значениях *C*, *D* и *K*, указанных в таблице:

Вещество	<i>C</i> , мг/л	<i>D</i> , мг/л	<i>K</i>
Свинец	10	0,03	1,12
Мышьяк	5	0,05	1,05
Фтор	8	0,05	1,01

Вы видите, что машина выводит на экран большое количество строк. Они быстро сменяют друг друга, поэтому трудно воспринять заключенную в них информацию. Давайте изменим условие задачи и программу, потребовав, чтобы ЭВМ сообщала, какой будет концентрация вредных примесей в реке не ежедневно, а через каждые 30 дней.

Приведем измененную программу:

```
1 ЗАПРОСИТЬ C,D,K
2 N=0
3 ПОКА C>D, ПОВТОРЯТЬ:
4 N=N+30
5 C=C/K^30
6 СООБЩИТЬ "Номер суток"
7 СООБЩИТЬ N
8 СООБЩИТЬ "Концентрация"
9 СООБЩИТЬ C
10 КОНЕЦ ЦИКЛА
```

Вот та же программа, записанная на языке Бейсик:

```
1 INPUT C,D,K
2 LET N=0
3 IF C<=D THEN 9
4 LET N=N+30
5 LET C=C/K^30
6 PRINT "Номер суток" N
7 PRINT "Концентрация" C
8 GOTO 3
9 STOP
```

Запустите измененную программу и посмотрите, какие результаты теперь выводятся на экран. Заметьте, что снова, как и в предыдущей лабораторной работе, натуральный эксперимент мы заменили вычислительным (к сожалению, руководители отдельных предприятий, сбрасывающих сточные воды без должной очистки, слишком пассивно наблюдают за подобными натурными "экспериментами").

## § 18. МЕТОД МОНТЕ-КАРЛО

У вас могло сложиться впечатление, что математические модели нужно строить только для решения задач, далеких от математики. На самом деле и в математике для решения задач часто требуются математические модели. Одна из таких задач — вычисление площадей. Конечно, для простейших фигур (прямоугольников, многоугольников, кругов) вычисление площади не составляет труда: надо в известные формулы подставить исходные данные. А как быть, если фигура имеет сложную форму? Итак,

**З а д а ч а.** Дана фигура сложной формы. Вычислить ее площадь.

Можно предложить разные модели для этой задачи. Например, в шестом классе вас учили использовать палетку: на фигуру накладывается клетчатая прозрачная бумага (палетка) и подсчитывается количество квадратиков, попавших в фигуру. В этой модели молчаливо предполагается, что, чем мельче клетки, тем точнее будет результат независимо от того, каким образом наложить палетку на фигуру.

Можно придумать и "физическую" модель: скопировать фигуру на картон, аккуратно вырезать ее, взвесить и поделить на вес единичного квадрата из этого же картона.

В одиннадцатом классе вы познакомитесь еще с одним способом нахождения площадей фигур — с помощью интегралов.

Однако по всем этим моделям довольно трудно составить алгоритмы для расчетов на ЭВМ.

Как видите, и для решения математических задач можно составлять различные математические модели. Математические модели жизненных задач — это "мосты" между жизнью и математикой. Математические модели, применяемые для решения математических задач, — это "мосты" между различными разделами математики. Например, метод координат позволяет создавать алгебраические модели геометрических объектов.

Математическая модель, которую мы сейчас построим, может показаться вам неожиданной, но она позволяет очень эффективно применять ЭВМ для решения задач на нахождение площадей, объемов и т. п.

Поместим данную фигуру в квадрат. Будем наугад (как говорят математики, случайным образом) бросать точки в этот квадрат. Естественно предполагать, что, чем больше площадь фигуры, тем чаще в нее будут попадать точки. Представьте себе квадратный дворик и в нем детскую площадку. Каждому ясно, что во время снегопада количество снежинок, попавших на детскую площадку, пропорционально ее площади. Таким образом, можно сделать допущение: при большом числе точек, наугад выбранных внутри квадрата, доля точек, содержащихся в данной фигуре, приближенно равна отношению площади этой фигуры к площади квадрата.

Такой метод приближенного нахождения площадей фигур носит название **метода Монте-Карло** (по названию города, где расположена знаменитая рулетка, которую можно рассматривать как "генератор" случайных чисел).

Несложно определить, что в нашей модели — исходные данные, а что — результат. Обозначим данную фигуру бук-

*Только случайность  
поможет вам найти  
площадь фигуры  
методом Монте-Карло*

вой  $F$ . Исходными являются сторона  $a$  квадрата, содержащего фигуру  $F$ , и количество точек  $N$ , которые мы будем случайным образом выбирать внутри квадрата. Результатом является площадь

$S$  фигуры  $F$ . Если через  $M$  обозначить число тех наугад выбранных точек, которые содержатся в фигуре  $F$ , то площадь  $S$  приблизительно равна  $a^2 M/N$ . Разумеется, к связям между исходными данными и результатом следует отнести и математические соотношения, позволяющие определить, попала ли выбранная точка в фигуру  $F$ .

Эти соотношения будут отличать модели, построенные для разных фигур. Значит, мы описали не одну модель, а скорее — некоторый способ получения моделей. Для каждой конкретной фигуры будет получаться своя модель.

Давайте рассмотрим математическую модель для приближенного нахождения площади круга радиуса  $R$ . Формула площади круга вам, конечно, известна:  $S=\pi R^2$ . Однако, если вы помните, эту формулу вам сообщили фактически без доказательства. Вот подходящий случай проверить ее с помощью ЭВМ!

Пусть для определенности  $R=1$ . На рисунке 31 изображен круг радиуса 1, заключенный в квадрат со стороной  $a=2$ . Таким образом, в качестве фигуры  $F$  выступает круг единичного радиуса. Выбрать точку — это значит задать ее координаты: числа  $x$  и  $y$ .

Каждому ясно: точка принадлежит квадрату, если  $-1 \leq x \leq 1$  и  $-1 \leq y \leq 1$ . Теорема Пифагора "подсказывает": если  $x^2 + y^2 \leq 1$ , то точка попадает в круг  $F$ , иначе она вне круга. Это и есть математическое соотношение, позволяющее для каждой точки определить, лежит ли она в  $F$ .

Математическая модель вычисления площади круга построена, и, казалось бы, можно приступать к составлению алгоритма для ВЫЧИСЛИТЕЛЯ. Однако, составляя алго-

ритм, мы быстро обнаружим, что запас допустимых действий ВЫЧИСЛИТЕЛЯ недостаточен: он не умеет выбирать числа случайным образом. Ну что ж, пополним запас допустимых действий ВЫЧИСЛИТЕЛЯ: по команде

Присвоить  $z$  значение  $RND(a)$

ВЫЧИСЛИТЕЛЬ присваивает переменной  $z$  случайное значение, заключенное между  $-a$  и  $a$ . Запишем алгоритм:

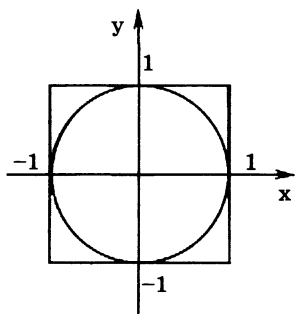


Рис. 31

Запросить количество точек  $N$ .  
 Присвоить номеру  $I$  выбранной точки значение 0.  
 Присвоить числу точек  $M$ , попавших в фигуру, значение 0.  
 Пока  $I \leq N$ , повторять:  
 Присвоить  $X$  значение  $RND(1)$ .  
 Присвоить  $Y$  значение  $RND(1)$ .  
 Если  $X^2 + Y^2 \leq 1$ , то:  
 Присвоить  $M$  значение  $M+1$ .  
 Конец ветвления.  
 Присвоить  $I$  значение  $I+1$ .  
 Присвоить  $S$  значение  $4M/I$ .  
 Сообщить "Число выбранных точек; площадь:".  
 Сообщить  $I, S$ .  
 Конец цикла.

Вычислительный эксперимент с алгоритмом вы проведете, выполняя лабораторную работу 10.

В языке Бейсик выбор случайного числа между  $-A$  и  $A$  осуществляется по команде  $LET Z=2*A*RND(1)-A$

### ? Вопрос

Какое допустимое действие **ВЫЧИСЛИТЕЛЯ** позволяет ему выбирать случайные числа?

### Задания для самостоятельного выполнения

1°. Составьте для **ВЫЧИСЛИТЕЛЯ** алгоритм нахождения площадей следующих фигур методом Монте-Карло (рис. 32—34).

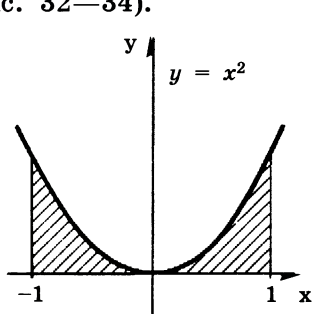


Рис. 32

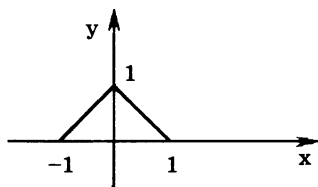


Рис. 33

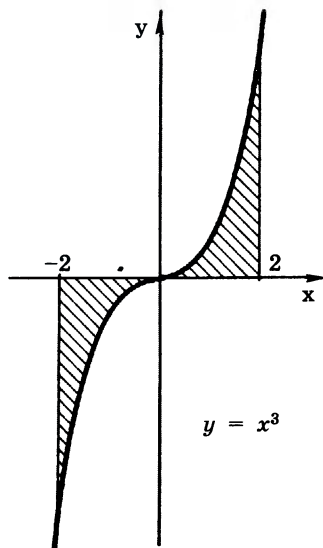


Рис. 34

2. Школьник решил обучить ВЫЧИСЛИТЕЛЯ играть с ним в игру "Угадай-ка" и составил алгоритм. Злоумышленник стер в алгоритме одну строку. Вот что получилось:

Запросить  $a$ .

Присвоить  $z$  значение  $RND(a)+a$ .

Присвоить  $N$  значение 0.

Пока  $|b-z| \geq 1$ , повторять:

Сообщить "Попробуйте угадать задуманное число".

Сообщить "Введите свое число".

Запросить  $b$ .

Если  $b-z \geq 1$ , то:

Сообщить "Задуманное число меньше".

Конец ветвления.

Если  $b-z \leq -1$ , то:

Сообщить "Задуманное число больше".

Конец ветвления.

Присвоить  $N$  значение  $N+1$ .

Конец цикла.

Сообщить "Вы угадали!".

Сообщить "Число попыток:".

Сообщить  $N$ .

Восстановите стертую строку. Какой диалог ведет ВЫЧИСЛИТЕЛЬ, выполняя исправленный алгоритм?

В заданиях 3—5 составьте математические модели и алгоритмы решения задач.

3°. Каждый день резидент приходит на встречу с агентом в случайный момент времени с 11 до 13 ч и ждет его 15 мин. Агент также приходит на встречу с ним каждый день с 11 до 13 ч случайным образом и тоже ждет 15 мин. Как часто за год встречаются резидент и агент?

4°. Ученик пишет квадратные уравнения  $x^2 + px + q = 0$ , выбирая коэффициенты  $p$  и  $q$  случайным образом из отрезка  $[-1; 1]$ . С какой частотой он будет писать уравнения, имеющие действительные корни?

5°. Завод, стоящий на берегу реки, сбрасывает в нее ежедневно случайным образом (из-за неисправности очистных сооружений) от 0 до 30 кг вредных веществ. За каждый килограмм сверх 15 завод обязан заплатить штраф 100 тыс. р. Прибыль завода от реализации продукции 700 тыс. р. в день. Как часто в течение пятилетки штраф превазойдет прибыль? Рентабелен ли такой завод?

6°. а) Постройте математическую модель игры в "Спортлото". Составьте алгоритм для ВЫЧИСЛИТЕЛЯ, помогающий вам играть в эту лотерею.

б) (К решению этой задачи дети до 16 лет не допускаются.) Постройте математическую модель игры в рулетку. Составьте алгоритм, имитирующий эту игру с помощью ВЫЧИСЛИТЕЛЯ.

7. Учитель хочет, чтобы **ВЫЧИСЛИТЕЛЬ** помог ему проверить знание учащимися таблицы умножения, задавая сомножители случайным образом. Составьте математическую модель и алгоритм опроса учащихся.

У к а з а н и е: используйте решение задачи 14,6 к § 17.



## Лабораторная работа 10

### РЕШЕНИЕ ЗАДАЧ С ИСПОЛЬЗОВАНИЕМ ЦИКЛИЧЕСКИХ И РАЗВЕТВЛЯЮЩИХСЯ АЛГОРИТМОВ

У вас уже накопилось много алгоритмов для **ЧЕРТЕЖНИКА** и **ВЫЧИСЛИТЕЛЯ**. Сегодня вы сможете посмотреть их в действии. Как обычно, начните с **ЧЕРТЕЖНИКА**. Запрограммируйте решение задачи 5 из § 18 и проследите, как **ЧЕРТЕЖНИК** выполняет программу. Если потребуются, отладьте ее. Надеемся, что работа с **ЧЕРТЕЖНИКОМ** отняла у вас немного времени и вы вспомнили, как исполняются алгоритмы, содержащие циклы и развилки.

Остальная часть занятия будет посвящена компьютерным экспериментам по нахождению площадей фигур методом Монте-Карло. Введите в ЭВМ программу вычисления площади круга из § 18 и запустите ее на исполнение. При этом значение  $N$  (число выбранных точек) надо брать достаточно большим — не менее 300. Когда авторы запустили эту программу, значение площади получилось равным 3,1287 (от точного значения  $\pi$  этот результат отделяют менее трех сотых). А какой результат получился у вас?

Не правда ли, необычный вопрос? Раньше такой вопрос не возникал, потому что результат работы программы всегда был один и тот же при одинаковых исходных данных независимо от того, кто и сколько раз эту программу запускал. Математическая модель, выбранная нами для решения задач на нахождение площадей, отличается от предыдущих тем, что в ней используются случайные числа (такие модели называют **вероятностными**). Поэтому и результаты могут получаться разными.

Некоторые из вас могли усомниться: а можно ли с помощью вероятностных моделей получать сколько-нибудь достоверные результаты? На подобные вопросы отвечает специальный раздел математики — теория вероятностей. Вы можете не волноваться, доказано, что метод Монте-Карло действительно применим для нахождения площадей.

На самом деле точность результатов зависит не только от того, является модель вероятностной или нет, — это зависит и от точности исходных данных, и от точности вычислений. Таким образом, неточные результаты могут получаться и при вычислениях по тем моделям, в которых случайность вроде бы и не присутствует.

Используя алгоритмы, составленные вами при решении задачи 1 из § 18, вычислите площади фигур, перечисленных в пунктах а) — в) этой задачи. Сравните ваши результаты с точными значениями площадей: а)  $2/3$ ; б) 8; в) 1.

Вас, конечно, заинтриговала задача 4 из § 18 о том, как часто квадратные уравнения имеют действительные корни. Запустите составленную вами программу и получите ответ (он не должен сильно отличаться от  $13/24$ ).



## КОНСПЕКТ ГЛАВЫ 5

Используя только развилки, нельзя записать инструкции, в которых один и тот же набор действий надо выполнять много раз подряд и заранее неизвестно количество повторений. Для этого нужна новая форма организации действий — цикл.

Циклом (повтором) называется такая форма организации действий, при которой одна и та же последовательность действий совершается несколько раз (или ни разу) до тех пор, пока выполняется некоторое условие.

Если действия  $P_1, P_2, \dots, P_n$  надо повторять, пока выполняется некоторое условие  $Q$ , то мы будем использовать следующую запись:

Пока  $Q$ , повторять:

$P_1$

$P_2$

$\dots$

$P_n$

Конец цикла.

Эта запись означает, что исполнитель сначала проверяет, выполняется ли условие  $Q$ . Если да, то совершаются действия  $P_1, P_2, \dots, P_n$  (последовательность этих действий называют телом цикла), после чего условие  $Q$  проверяется снова и т. д. Если же  $Q$  не выполняется, то исполнитель переходит к действию, записанному после строки "Конец цикла". В частности, если условие  $Q$  не выполнено с самого начала, то действия, составляющие тело цикла, не совершатся ни разу. Указатель "Конец цикла" позволяет избежать двусмысленности в записи цикла.

Приведем пример циклического алгоритма для ЧЕРТЕЖНИКА:

Пока впереди не край, повторять:

Повернуть налево.

Конец цикла.

Повернуть налево.

Повернуть налево.

Прыгнуть.

Выполнив его, ЧЕРТЕЖНИК отойдет на один шаг от края (в начале ЧЕРТЕЖНИК был у края, но не в углу).

При использовании циклической формы организации действий возможности ЭВМ проявляются очень наглядно: написав короткий циклический алгоритм, можно заставить ЭВМ автоматически выполнить сколь угодно большой объем работы. Одна из задач, решение которой фактически стало возможным только после появления ЭВМ, — нахождение площадей фигур методом Монте-Карло. Он состоит в следующем. Пусть требуется найти площадь фигуры  $F$  сложной формы. Поместим эту фигуру в прямоугольник с известными сторонами (тогда и площадь его известна). Будем случайным образом бросать точки в этот прямоугольник. Метод Монте-Карло основан на допущении, что при большом числе бросаний доля точек, попавших в  $F$ , приближенно равна отношению площади фигуры  $F$  к площади прямоугольника.

Для решения задач методом Монте-Карло нужно пополнить запас допустимых действий ВЫЧИСЛИТЕЛЯ командой

**Присвоить  $z$  значение  $RND(a)$**

По этой команде ВЫЧИСЛИТЕЛЬ присваивает переменной  $z$  случайное значение, заключенное между  $-a$  и  $a$ .

Вот алгоритм для нахождения площади полукруга радиуса 1 методом Монте-Карло. Полукруг помещен в прямоугольник со сторонами 2 и 1.

**Запросить количество точек  $N$ .**

**Присвоить номеру  $I$  выбранной точки значение 0.**

**Присвоить числу  $M$  значение 0.**

**Пока  $I \leq N$ , повторять:**

**Присвоить абсциссе  $X$  значение  $RND(1)$ .**

**Присвоить ординате  $Y$  значение  $(RND(1)+1)/2$ .**

**Если  $X^2 + Y^2 \leq 1$ , то:**

**Присвоить  $M$  значение  $M+1$ .**

**Конец ветвления.**

**Присвоить  $I$  значение  $I+1$ .**

**Присвоить  $S$  значение  $2M/I$ .**

**Сообщить "Число выбранных точек; площадь".**

**Сообщить  $I, S$ .**

**Конец цикла.**

Математические модели, основанные на использовании случайных чисел, называются вероятностными. Заметим, что при одних и тех же исходных данных в них могут получаться различные результаты. Значит, вероятностные модели всегда являются приближенными. Это не является недостатком таких моделей. Фактически точность результатов зависит от многих причин, в частности от точности исходных данных и от точности вычислений. Поэтому неточные результаты могут получаться и при вычислениях по тем моделям, в которых случайность вроде бы и ни при чем.

## Глава 6

# ВСПОМОГАТЕЛЬНЫЕ АЛГОРИТМЫ

---

Чтобы быстрее добраться из одного пункта в другой, человек изобрел поезда, автомобили, самолеты. Чтобы быстрее выполнить ту или иную работу, человек придумал разные хитроумные устройства. Чтобы быстрее решать задачи, человек создал ЭВМ. Борьба за экономию времени пронизывает всю историю человеческой деятельности. Для этого придумывались самые разнообразные вспомогательные средства. Есть такое средство и у программистов. Это вспомогательные алгоритмы. Конечно, циклов и развилочек вполне достаточно, чтобы записать любой алгоритм. Поэтому вспомогательными алгоритмами пользоваться необязательно (ведь и в Москву из Владивостока можно прийти пешком, не пользуясь транспортом). Но они экономят время и силы, уменьшают количество ошибок.

### §19. ПОНЯТИЕ ВСПОМОГАТЕЛЬНОГО АЛГОРИТМА

Представьте себе, что вы находитесь в летнем лагере труда и отдыха, вас назначили дежурным по столовой и вы хотите продумать порядок своих действий. Ясно, что некоторые обязанности дежурного придется в течение дня исполнять несколько раз. Например, такие:

Убрать со столов посуду.  
Вытереть столы.  
Расставить стулья.  
Уйти.

Эту последовательность действий предстоит выполнить три раза за дежурство: после завтрака, обеда и ужина. Назовем соответствующий алгоритм "Обязанности", а при написании расписания дня заменим эти четыре действия одной командой:

Выполнить алгоритм "Обязанности".

Тогда расписание дежурства по столовой примет такой вид:

Прийти в столовую в 7.00.  
Накрыть столы к завтраку.  
Дождаться окончания завтрака.  
Выполнить алгоритм "Обязанности".  
Прийти в столовую в 13.00.  
Накрыть столы к обеду.  
Дождаться окончания обеда.  
Выполнить алгоритм "Обязанности".  
Прийти в столовую в 18.00.  
Накрыть столы к ужину.  
Дождаться окончания ужина.  
Выполнить алгоритм "Обязанности".

Легко понять, что использование команды "Выполнить алгоритм..." позволило почти в два раза сократить длину нашей инструкции.

Подобным образом поступают часто, когда при составлении алгоритма возникает необходимость многократного использования одного и того же набора действий. Этот набор действий выделяют в качестве самостоятельного алгоритма и дают ему имя. С этого момента он становится **вспомогательным алгоритмом**, т. е. появляется возможность его использования в других алгоритмах. Набор команд, из которых состоит вспомогательный алгоритм, заменяют одной командой

*Каждому  
вспомогательному  
алгоритму —  
достойное имя*

**Выполнить алгоритм ...,**

указывая вместо многоточия имя алгоритма. Такие команды называются **командами вызова вспомогательного алгоритма**.

Составляя для исполнителя вспомогательные алгоритмы, мы как бы обучаем исполнителя новым действиям. Точно так же происходит и обучение в школе. Вначале школьников обучают сложению и вычитанию, затем, используя эти умения, обучают действиям умножения и деления. Умножение и деление в свою очередь используются в дальнейшем при решении более сложных задач.

Раз уж речь зашла о школе, вспомним материал седьмого класса — решение квадратных уравнений. Можно записать решение квадратного уравнения  $ax^2+bx+c=0$  (считаем, что  $a \neq 0$ ) в виде следующего алгоритма:

**Найти дискриминант  $d=b^2-4ac$ .**

**Если  $d < 0$ , то:**

**Сообщить "Уравнение корней не имеет".**

**Иначе:**

**Найти квадратный корень из  $d$  и обозначить его буквой  $p$ .**

Найти  $x_1$  по формуле  $x_1 = \frac{-b-p}{2a}$ .

Найти  $x_2$  по формуле  $x_2 = \frac{-b+p}{2a}$ .

Конец ветвления.

Дадим алгоритму решения квадратных уравнений имя "РЕКВУР" (впрочем, вы можете назвать этот алгоритм по-другому, более благозвучно).

И в седьмом классе вы, конечно, сталкивались с квадратными уравнениями в ходе решения различных задач. Но теперь решение квадратных уравнений стало для вас столь же элементарным, как перемножение чисел (возможно, что мы выдаем желаемое за действительное). Можно сказать, что "РЕКВУР" стал вашим вспомогательным алгоритмом.

Давайте проанализируем, как мы используем алгоритм "РЕКВУР". Скажем, нужно решить уравнение

$$2x^2 + 3x - 122 = 0.$$

Мы вспоминаем "РЕКВУР", подставляем вместо  $a$ ,  $b$ ,  $c$  числа 2, 3 и 122 и получаем результат:  $x_1$  и  $x_2$ . Чтобы решить другое квадратное уравнение, надо подставить вместо  $a$ ,  $b$ ,  $c$  другие числа.

Как видите, между вспомогательными алгоритмами "Обязанности" и "РЕКВУР" есть существенная разница. В команде вызова алгоритма "РЕКВУР" необходимо указывать значения исходных данных — коэффициентов квадратного уравнения, которое требуется решить. А в алгоритме "Обязанности" нет исходных данных. Поэтому ничего, кроме названия, в команде вызова указывать не надо.

При использовании вспомогательного алгоритма, как правило, никого не интересует, из каких действий он состоит. Важно только, каковы **исходные данные (аргументы)** этого алгоритма и что является **результатом** его работы. Точно так же, для того чтобы пользоваться стиральной машиной, необязательно вникать в ее устройство. Достаточно знать, что "аргументами" для нее являются стиральный порошок, чистая вода и грязные вещи, а "результатом" — грязная мыльная вода и чистые вещи. Что касается алгоритма "РЕКВУР", то его аргументами являются коэффициенты  $a$ ,  $b$  и  $c$ , а результатом либо корни  $x_1$ ,  $x_2$ , либо сообщение, что корней нет.

Записывая вспомогательный алгоритм, естественно указывать, кроме названия, аргументы и результаты. Например, запись вспомогательного алгоритма "РЕКВУР" можно начать так:

Алгоритм "РЕКВУР".

Аргументы:  $a$ ,  $b$ ,  $c$  (коэффициенты квадратного уравнения).

Результаты:  $x_1$ ,  $x_2$  или сообщение "Корней нет".

...

При вызове вспомогательного алгоритма надо указывать значения исходных данных. Например, вызов вспомогательного алгоритма "РЕКВУР" можно записать так:

Выполнить алгоритм "РЕКВУР" при  $a=...$ ,  $b=...$ ,  $c=...$ .

Или так:

Выполнить алгоритм "РЕКВУР", взяв в качестве аргументов ..., ... и ...

(вместо многоточий надо писать конкретные числа или выражения).

Давайте с помощью алгоритма "РЕКВУР" решим какую-нибудь задачу из вашего учебника алгебры. Заглянем в него. Вот, например, тема "Биквадратные уравнения". В ней объясняется, как решать уравнения вида

$$x^4 + px^2 + q = 0.$$

По сути дела, в вашем учебнике предлагается следующий алгоритм решения таких уравнений:

Запросить  $p$ ,  $q$ .

Выполнить алгоритм "РЕКВУР" при  $a=1$ ,  $b=p$ ,  $c=q$ .

Если  $p^2 - 4q \geq 0$  (т. е. если квадратное уравнение  $t^2+pt+q=0$  имеет корни), то:

Решить уравнение  $x^2 = x_1$ .

Сообщить  $x$ .

Решить уравнение  $x^2 = x_2$ .

Сообщить  $x$ .

Конец ветвления.

Здесь необходимо упомянуть об одном очень важном обстоятельстве. Представьте себе, что вы подарили своему товарищу вспомогательный алгоритм "РЕКВУР" и объяснили, как им пользоваться. Разумеется, вашему товарищу нет никакого дела до того, какие обозначения вы использовали в вашем алгоритме. Поэтому может случиться, что в его основном алгоритме будут использоваться те же обозначения, что и в "РЕКВУРЕ", например  $p$  (именно это произошло в только что написанном алгоритме).

Понятно, что надо как-то различать переменные, одинаково обозначенные в основном и вспомогательном алгоритмах. Поэтому договариваются считать, что все обозначения из вспомогательного алгоритма действительны только в пределах этого алгоритма. Сказанное, конечно же, не относится к результатам работы вспомогательного алгоритма,

потому что они должны использоваться и после завершения его работы.

Образно говоря, переменные, имеющие одно и то же название в основном и вспомогательном алгоритмах, просто "однофамильцы". При вызове вспомогательного алгоритма значения переменных основного алгоритма как бы "замораживаются", а после окончания работы вспомогательного алгоритма они "размораживаются".

Без такого "замораживания" то и дело происходили бы пренеприятнейшие вещи. Какие? Сейчас увидите!

Попробуем решить уравнение

$$x^4 - 8x^2 = 425.$$

Посмотрим, как выполнялся бы написанный выше алгоритм, если бы "замораживания" не происходило. Выполнив команду "Запросить  $p, q$ ", мы присвоим переменной  $p$  значение  $-8$ , а переменной  $q$  значение  $-425$ .

Теперь выполним алгоритм "РЕКВУР", взяв в качестве  $a, b, c$  числа  $1, -8, -425$  соответственно. Получим  $x_1 = 25$ ,  $x_2 = -17$ . Далее, для проверки условия надо подсчитать

*Замораживание  
переменных  
надежно  
предохраняет их  
от порчи  
вспомогательными  
алгоритмами*

значение  $p^2 - 4q$ . Но что это? Значение  $p$  уже не равно  $-8$ ! При выполнении алгоритма "РЕКВУР" ему было присвоено значение  $42$ . От такой "помощи" вспомогательного алгоритма становится не по себе. Это все равно, как если бы ваш магнитофон при воспроизведении записей одновременно заменял на

кассете, скажем, "Танец с саблями" А.И.Хачатуряна на "Танец маленьких лебедей" П.И.Чайковского. В действительности же после выполнения вспомогательного алгоритма значения  $p$  и  $q$  "размораживаются", т. е. снова становятся равными  $-8$  и  $-425$ , и алгоритм дает верный результат (какой?).

Итак, использование вспомогательных алгоритмов — еще одна форма организации действий. Вспомогательным называется алгоритм, снабженный таким заголовком, который позволяет вызывать этот алгоритм из других алгоритмов. Любой алгоритм можно сделать вспомогательным: надо лишь снабдить его соответствующим заголовком (т. е. указать название, аргументы и результаты).

Как видите, вспомогательные алгоритмы — мощное и удобное средство, облегчающее решение трудных задач. Можно сказать, что искусство составления алгоритмов как раз и заключается в умении конструировать сложный алгоритм из более простых вспомогательных алгоритмов, т. е. в

умении обучать исполнителя сложным вещам постепенно, идя от простого к сложному.

## ? Вопросы

1. Что такое вспомогательный алгоритм?
2. Как сделать алгоритм вспомогательным?
3. Какие сведения указываются в команде вызова вспомогательного алгоритма?
4. В чем заключается "замораживание" и "размораживание" переменных при использовании вспомогательного алгоритма?



## Задания для самостоятельного выполнения

1. Какие вспомогательные алгоритмы можно выделить в распорядке дня в школе?
2. Какие вспомогательные алгоритмы можно выделить в печатании текстов с помощью текстового редактора?
3. Имеется следующий вспомогательный алгоритм:

**Алгоритм "Воздействие".**

**Аргументы:** предметы  $x, y, z$ ; время  $t$ .

**Результат:**  $x$ .

**Поместить  $y$  на  $x$ .**

**Подождать  $t$  минут.**

**Удалить  $y$  с помощью  $z$ .**

Запишите команды вызова этого вспомогательного алгоритма, выполнив которые исполнитель (человек):

- а) выведет пятно с куска ткани при помощи пятновыводителя и воды;
- б) вымоет окно;
- в) съест ложку варенья;
- г) почистит зубы.

Придумайте еще какую-нибудь задачу, которую можно решить с помощью алгоритма "Воздействие".

4. Дан треугольник  $ABC$ . Определить, для решения какой задачи предназначен следующий алгоритм:

Выполнить алгоритм "Серединный перпендикуляр" при  $X=A$  и  $Y=B$ .

Выполнить алгоритм "Серединный перпендикуляр" при  $X=B$  и  $Y=C$ .

Поставить ножку циркуля в точку  $O$  пересечения полученных прямых.

Установить раствор циркуля равным длине отрезка  $OA$ .

Провести окружность.

Алгоритм "Серединный перпендикуляр". Аргументы: точки  $X, Y$ .

Результат: прямая.

Поставить ножку циркуля в точку Х.  
 Установить раствор циркуля равным длине отрезка ХУ.  
 Провести окружность.  
 Поставить ножку циркуля в точку У.  
 Провести окружность.  
 Провести прямую через точки пересечения окружностей.

У к а з а н и е: воспользуйтесь решением задачи 5 из § 9.

5. На экране компьютера с помощью графического редактора изображен треугольник. Составьте алгоритм нахождения центра вписанной окружности, используя в качестве вспомогательного алгоритм построения биссектрисы угла (см. задачу 9 из § 9).

## § 20. ВСПОМОГАТЕЛЬНЫЕ АЛГОРИТМЫ И ИСПОЛНИТЕЛИ АЛГОРИТМОВ

Давайте составим для ЧЕРТЕЖНИКА алгоритм изображения знакомой нам фигуры — "креста" (см. задачу 2, § 11), используя вспомогательный алгоритм (начальное положение ЧЕРТЕЖНИКА обозначено на рис. 35). Очевидно, заданная фигура состоит из четырех элементов, изображенных на рисунке 36. Действия, которые должен выполнить ЧЕРТЕЖНИК, чтобы нарисовать этот элемент (начальное положение как на рис. 36), запишем в виде вспомогательного алгоритма "Скобка":

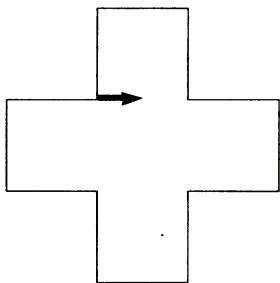


Рис. 35

Алгоритм "Скобка".  
 Повернуть налево.  
 Повернуть налево.  
 Сделать шаг.  
 Повернуть налево.  
 Сделать шаг.  
 Повернуть налево.  
 Сделать шаг.

Теперь можно составить основной алгоритм:

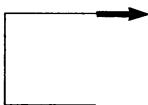


Рис. 36

Выполнить алгоритм "Скобка".  
 Повернуть налево.  
 Выполнить алгоритм "Скобка".  
 Повернуть налево.  
 Выполнить алгоритм "Скобка".  
 Повернуть налево.  
 Выполнить алгоритм "Скобка".

Можно сказать, что мы "научили" ЧЕРТЕЖНИКА совершать новое для него действие: рисовать скобку. И теперь это действие можно использовать для изображения не только данного, но и многих других рисунков.

Поработаем теперь с ВЫЧИСЛИТЕЛЕМ. Ранее мы составили для него алгоритм нахождения наибольшего из двух чисел. Давайте составим для ВЫЧИСЛИТЕЛЯ алгоритм нахождения наибольшего из трех чисел, скажем,  $a$ ,  $b$  и  $c$ . План наших действий может быть, например, таким: сначала найдем наибольшее из чисел  $a$  и  $b$ , обозначив результат буквой  $z$ , а затем найдем наибольшее из чисел  $z$  и  $c$ . Мы видим, что здесь дважды выполняется алгоритм нахождения наибольшего из двух чисел, каждый раз различных. Поэтому естественно записать вспомогательный алгоритм нахождения наибольшего из произвольных двух чисел, скажем,  $x$  и  $y$ , а затем его дважды использовать. Назовем этот алгоритм "Поиск максимума из двух чисел".

Алгоритм "Поиск максимума из двух чисел".

Аргументы: числа  $x$ ,  $y$ .

Результат:  $z$  (максимум из чисел  $x$  и  $y$ ).

Если  $x > y$ , то:

Присвоить  $z$  значение  $x$ .

Иначе:

Присвоить  $z$  значение  $y$ .

Конец ветвления.

Алгоритм поиска наибольшего из трех чисел можно теперь записать так:

Запросить  $a$ ,  $b$ ,  $c$ .

Выполнить алгоритм "Поиск максимума из двух чисел" при  $x=a$ ,  $y=b$ .

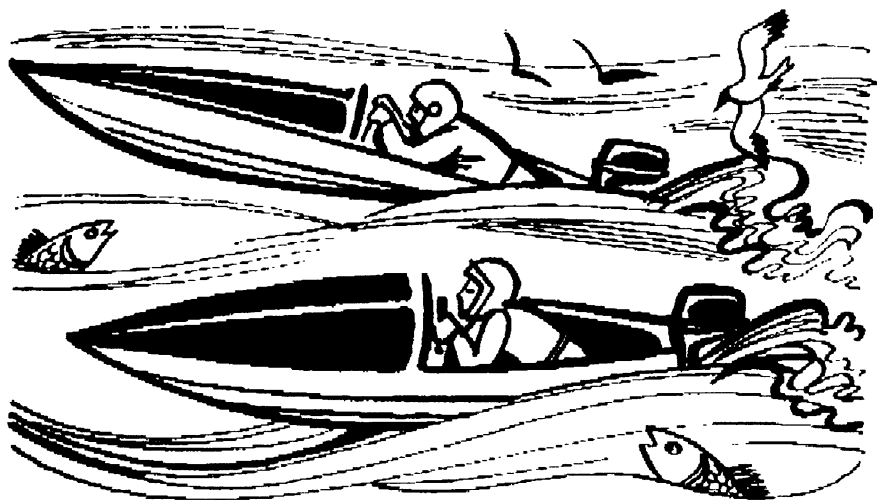
Выполнить алгоритм "Поиск максимума из двух чисел" при  $x=z$ ,  $y=c$ .

Сообщить  $z$ .

Как же ВЫЧИСЛИТЕЛЬ будет выполнять этот алгоритм? Сначала он запросит у нас три числа и обозначит их буквами  $a$ ,  $b$ ,  $c$ . После этого он выполнит вспомогательный алгоритм "Поиск...", взяв в качестве  $x$  число  $a$ , а в качестве  $y$  — число  $b$ . Найдя максимум  $z$ , он еще раз выполнит вспомогательный алгоритм "Поиск...", взяв на этот раз в качестве  $x$  число  $z$ , в качестве  $y$  число  $c$  и обозначив результат буквой  $z$ . Это число  $z$  ВЫЧИСЛИТЕЛЬ сообщит нам.

Чтобы подготовиться к лабораторной работе 11, разберем следующую задачу.

**Задача.** Две моторные лодки равномерно двигались по реке в направлении к озеру, в которое впадает река.



Поравнявшись, они начали двигаться равноускоренно. Какая из лодок раньше дойдет до озера?

Построение модели, как обычно, начнем с предположений. Поскольку вряд ли нам удастся получить очень точные сведения о лодках — их исходных положениях, скоростях, ускорениях..., модель построим попроще. Лодки будем считать точками, реку и движение лодок — прямолинейными. Исходными данными являются начальные скорости лодок (обозначим их  $p$  и  $q$ ), ускорения лодок ( $d$  и  $e$ ), расстояние до озера ( $S$ ). Результатом является сообщение о том, какая лодка раньше дойдет до озера или что лодки придут одновременно. Для определения этого надо найти, за какое время каждая лодка дойдет до озера, и сравнить получившиеся значения.

Время  $t$  находится из квадратного уравнения

$$vt + at^2/2 = S,$$

где  $v$  — начальная скорость,  $a$  — ускорение,  $S$  — пройденный путь. По условию задачи скорости и ускорения обеих лодок положительны.

Для нахождения корней этого квадратного уравнения воспользуемся вспомогательным алгоритмом "РЕКВУР". Правда, он был предназначен для человека и в нем использованы обозначения  $x_1$  и  $x_2$ . ВЫЧИСЛИТЕЛЬ таких обозначений "не понимает". Поэтому заменим их на  $x$  и  $y$ .

Поскольку скорости и ускорения положительны, лодки обязательно доплывут до озера. Поэтому наше квадрат-

ное уравнение имеет два корня (кстати, и дискриминант у него положителен; проверьте!). Какой же из корней выбрать?

Давайте рассуждать. Произведение корней отрицательно (проверьте, воспользовавшись теоремой Виета). Поэтому один из корней положительный, а другой — отрицательный. Отрицательный корень никакого физического смысла не имеет — ведь время движения лодок до озера не может быть меньше нуля. Значит, в качестве  $t$  надо взять положительный корень.

Вот теперь, когда мы окончательно разобрались (как нам кажется) в модели, напомним алгоритм решения задачи. В нем через  $T$  обозначено время движения первой лодки, а через  $Z$  — второй.

Сообщить "Введите расстояние до озера".

Запросить расстояние до озера  $S$ .

Сообщить "Введите начальные скорости лодок".

Запросить скорости лодок  $p$  и  $q$ .

Сообщить "Введите ускорения лодок".

Запросить ускорения лодок  $d$  и  $e$ .

Выполнить алгоритм "РЕКВУР", взяв в качестве аргументов  $d/2$ ,  $p$  и  $-S$ .

Если корень  $x > 0$ , то:

Присвоить времени  $T$  движения первой лодки значение  $x$ .

Иначе:

Присвоить времени  $T$  движения первой лодки значение  $y$ .

Конец ветвления.

Выполнить алгоритм "РЕКВУР", взяв в качестве аргументов  $e/2$ ,  $q$  и  $-S$ .

Если корень  $x > 0$ , то:

Присвоить времени  $Z$  движения второй лодки значение  $x$ .

Иначе:

Присвоить времени  $Z$  движения второй лодки значение  $y$ .

Конец ветвления.

Если  $T < Z$ , то:

Сообщить "Первой пришла лодка номер 1".

Конец ветвления.

Если  $T > Z$ , то:

Сообщить "Первой пришла лодка номер 2".

Конец ветвления.

Если  $T = Z$ , то:

Сообщить "Лодки пришли одновременно".

Конец ветвления.

В этом алгоритме мы не стали учитывать естественные ограничения на скорости, ускорения и расстояние до озера. Подумайте, каковы эти ограничения, и добавьте соответствующую проверку в алгоритм.

В языке Бейсик вспомогательные алгоритмы оформляются следующим образом. Перед заголовком ставится слово REM (от английского *remark* — пояснение). Например:

100 REM Поиск максимума из двух чисел X и Y  
После заголовка записывается сам вспомогательный алгоритм. В конце обязательно пишется слово RETURN (вернуться).

Вспомогательные алгоритмы лучше размещать после основного алгоритма. Это означает, что номера строк вспомогательного алгоритма должны быть больше номеров строк основного алгоритма. Прежде чем обратиться к вспомогательному алгоритму, надо присвоить аргументам этого алгоритма нужные значения, а затем написать команду GOSUB ..., где вместо многоточия пишется номер первой строки вспомогательного алгоритма.

Например, алгоритм нахождения максимума из трех чисел на языке Бейсик выглядит так:

```
10 INPUT A,B,C
20 LET X=A
30 LET Y=B
40 GOSUB 200
50 LET X=Z
60 LET Y=C
70 GOSUB 200
80 PRINT Z
90 STOP
200 REM Поиск максимума из двух чисел. Аргументы X,Y. Результат Z.
210 IF X>Y THEN LET Z=X ELSE LET Z=Y
220 RETURN
```

К сожалению, в языке Бейсик не предусмотрено "замораживание" значений переменных. Поэтому в нем нельзя использовать для обозначения переменных в тексте основной программы те буквы, которыми обозначены переменные в подпрограммах.



### Задания для самостоятельного выполнения

1. Что нарисует ЧЕРТЕЖНИК на бесконечном листе бумаги, выполняя следующий алгоритм:

- а) Выполнить алгоритм "Скобка".  
Выполнить алгоритм "Скобка".  
Выполнить алгоритм "Скобка".

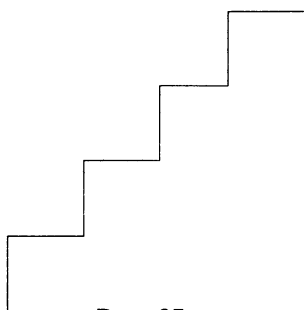


Рис. 37



Рис. 38

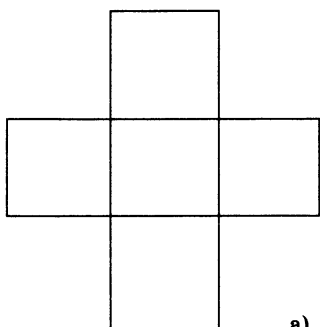
- б) Выполнить алгоритм "Скобка".  
 Повернуть налево.  
 Повернуть налево.  
 Выполнить алгоритм "Скобка".

2. Во вспомогательном алгоритме "Скобка" злоумышленник стер две команды. В результате, выполнив приведенный в данном параграфе алгоритм рисования "креста", ЧЕРТЕЖНИК вместо рисунка 35 изобразил рисунок 37. Какие команды стер злоумышленник?

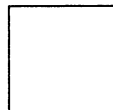
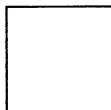
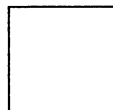
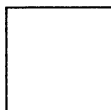
3. Составьте для ЧЕРТЕЖНИКА вспомогательный алгоритм поворота направо. Используя этот вспомогательный алгоритм, составьте алгоритм рисования скобки, начиная с нижнего правого конца (рис. 38).

4. Пользуясь решением задачи 6 из § 17, составьте для ЧЕРТЕЖНИКА алгоритм изображения на листе бумаги сетки из квадратов со стороной 1 см.

5. Составьте для ЧЕРТЕЖНИКА алгоритмы рисования фигур, изображенных на рисунке 39. Какой общий вспомогательный алгоритм естественно составить для рисования этих фигур?



а)



б)

Рис. 39

6. Чему будут равны переменные  $x$ ,  $y$ ,  $z$  после выполнения ВЫЧИСЛИТЕЛЕМ следующего алгоритма:

Присвоить  $x$  значение 1.  
Присвоить  $y$  значение 3.  
Присвоить  $z$  значение 3.  
Выполнить алгоритм "Поиск максимума из двух чисел" при  $x=x^y$ ,  $y=y^x$ .  
Выполнить алгоритм "Поиск максимума из двух чисел" при  $x=x^z$ ,  $y=z^y$ .

7. Пользуясь вспомогательным алгоритмом "Поиск максимума из двух чисел", составьте для ВЫЧИСЛИТЕЛЯ алгоритм нахождения максимального из четырех чисел.

8. Пользуясь алгоритмом нахождения максимального из четырех чисел как вспомогательным, составьте алгоритм решения модели для коммерческой задачи из § 7.

9. Составьте для ВЫЧИСЛИТЕЛЯ алгоритм решения уравнений вида

$$x^4 + px^2 + q = 0,$$

используя вспомогательный алгоритм решения квадратных уравнений.

10. В алгоритме решения задачи о двух лодках имеются два почти одинаковых блока действий. Найдите эти блоки и составьте соответствующий вспомогательный алгоритм.



## Лабораторная работа 11

### ПОДПРОГРАММЫ И ИХ ИСПОЛЬЗОВАНИЕ ПРИ РЕШЕНИИ ЗАДАЧ

В названии лабораторной работы вы, конечно, заметили новое слово: подпрограммы. И уже, наверно, догадались: оно означает вспомогательные алгоритмы, записанные на языке, понятном ЭВМ.

Вам хорошо известно, что запись программы отличается от записи алгоритма. Точно так же и оформление подпрограмм отличается от оформления вспомогательных алгоритмов. Сравните, например, вспомогательный алгоритм и подпрограмму, предназначенные для поиска максимума из двух чисел:

Вспомогательный алгоритм

Подпрограмма

Алгоритм "Поиск максимума из двух чисел".

20 ПОДПРОГРАММА [X,Y]

Аргументы:  $x$  и  $y$ .

Результат:  $z$ .

Если  $x < y$ , то:  
Присвоить  $z$  значение  $y$ .  
Иначе:  
Присвоить  $z$  значение  $x$ .  
Конец ветвления.

25 ЕСЛИ  $X < Y$ , ТО:  
30  $Z = Y$   
35 ИНАЧЕ:  
40  $Z = X$   
45 КОНЕЦ ВЕТВЛЕНИЯ  
50 КОНЕЦ ПОДПРО-  
ГРАММЫ [Z]

Вы видите, что при оформлении подпрограммы ее имя, а также слова "Аргументы" и "Результаты" надо опускать. Буквы, обозначающие аргументы, записываются в квадратных скобках после слова "ПОДПРОГРАММА". Подпрограмма завершается командой "КОНЕЦ ПОДПРОГРАММЫ". После этих слов в квадратных скобках записывают буквы, обозначающие результаты работы подпрограммы (если, конечно, таковые имеются).

А как же вызвать подпрограмму, раз имя ее ВЫЧИСЛИТЕЛЮ не сообщается? Роль имени играет номер первой строки подпрограммы. Поэтому, чтобы вызвать подпрограмму, пишут слова ВЫПОЛНИТЬ ПОДПРОГРАММУ, после них — не имя, а номер строки, с которой начинается подпрограмма. В квадратных скобках после номера перечисляют выражения, значения которых ЭВМ должна взять в качестве аргументов.

Например, вот как выглядит вызов подпрограммы "Поиск максимума ..." при  $x = a + bc$ , а  $y = a/b + c$ :

**5 ВЫПОЛНИТЬ ПОДПРОГРАММУ 20 [A+B\*C, A/B+C]**

Договоримся еще располагать подпрограммы после основной программы. Так, если последний номер основной программы 33, то подпрограмма должна начинаться с номера большего, чем 33.

Усвоив правила оформления подпрограмм, запишите в виде программы для ЧЕРТЕЖНИКА алгоритм рисования фигуры, изображенной на рисунке 37 (§ 20), и запустите программу, поставив ЧЕРТЕЖНИКА в центр листа.

Теперь поставьте его близко к краю листа. Вы видите, что ЧЕРТЕЖНИК не смог выполнить программу. Почему же это произошло? Может, программа написана неверно? Нет! Все дело в том, что наш алгоритм предназначался для работы на бесконечном листе бумаги, а вы заставили ЧЕРТЕЖНИКА работать на ограниченном листе (бесконечный лист, как вы понимаете, трудно изобразить на экране дисплея). Многие ошибки людей связаны с попытками применить в новых условиях старые методы, алгоритмы, инструкции.

Алгоритм изображения рисунка 37 на ограниченном листе бумаги довольно сложен. Попытайтесь его составить самостоятельно.

Теперь посмотрим, как вспомогательные алгоритмы выполняет **ВЫЧИСЛИТЕЛЬ**. Как вы помните, в § 20 мы специально приготовили для него задачу о двух лодках.

**Задача.** Две моторные лодки равномерно двигались по реке в направлении к озеру, в которое река впадает. Поравнявшись, они начали двигаться равноускоренно. Какая из лодок раньше дойдет до озера?

Запишите программу решения этой задачи и запустите ее при подходящих значениях скоростей, ускорений и расстояния до озера. (В программе вместо  $\sqrt{x}$  надо писать  $\text{SQR}(x)$ ).

Зафиксируйте начальные скорости обеих лодок и ускорение первой лодки. С каким наименьшим ускорением должна плыть вторая лодка, чтобы не отстать от первой? Сделайте так, чтобы **ВЫЧИСЛИТЕЛЬ** сообщил вам об этом.

Измените программу так, чтобы **ВЫЧИСЛИТЕЛЬ** сообщал, сколько времени затратила каждая лодка на путь до озера. Запустите программу. А теперь давайте проверим нашего исполнителя. Подставьте найденное время движения, скажем, первой лодки в левую часть уравнения

$$vt + at^2/2 = S$$

и найдите ее значение с помощью **ВЫЧИСЛИТЕЛЯ**. Вы увидите, что значение левой части не равно  $S$ ! Как же так? Ведь значение  $t$  является корнем как раз этого уравнения. Подумав, вы поймете, в чем тут дело: ответственность за несовпадение несут **ошибки округления**. ЭВМ округляет числа при любых вычислениях, неизбежно внося в результаты небольшие ошибки. Если же вычислений очень много, то ошибки могут "накапливаться", искажая результаты так, что с истиной не остается ничего общего. Впрочем, выполняя лабораторные работы, вы вряд ли с этим столкнетесь; и если ЭВМ выдала вам абсолютно неверный результат, то виноваты вы, а не безропотный компьютер.

Если осталось время, то давайте обратимся еще раз к нашей задаче о лодках. Мы считали, что они двигались равноускоренно. А что, если допустить и равнозамедленное движение? Тогда ускорения смогут принимать и отрицательные значения. Вот и тема для вычислительного эксперимента! Надо, конечно, уточнить построенную компьютерную модель, алгоритм и программу. Прежде всего уберите проверку положительности ускорений. Запустите программу, взяв расстояние  $S=500$  м и следующие скорости и ускорения:  $p=2$  м/с,  $q=4$  м/с,  $d=-0,01$  м/с<sup>2</sup>,  $e=-0,02$  м/с<sup>2</sup>. На экране появится сообщение "Уравнение корней не имеет". Откуда оно взялось? Это сработала подпрограмма "РЕКВУР". Раньше такое сообщение нам не

встречалось, поскольку при положительном ускорении соответствующее квадратное уравнение всегда имеет корни (мы это обсуждали в § 20). Если же ускорения отрицательны, то нельзя гарантировать, что обе лодки доплывут до озера. Другими словами, нельзя гарантировать, что каждое из наших квадратных уравнений имеет корни. Понятно, что в случае отсутствия корней **ВЫЧИСЛИТЕЛЬ** должен вежливо сообщить, что соответствующая лодка до озера не доплыла. Проверка этого должна быть предусмотрена в основной программе. Значит, выполнив "РЕКВУР", **ВЫЧИСЛИТЕЛЬ** должен запомнить, имело уравнение корни или нет. А сейчас он, сообщив об отсутствии корней, тут же забывает об этом. Да и не может **ВЫЧИСЛИТЕЛЬ** анализировать сообщения (даже свои собственные).

Как видите, порой недостаточно просто снабдить алгоритм заголовком, чтобы эффективно воспользоваться им как вспомогательным. Нередко результаты вспомогательного алгоритма не позволяют "состыковать" его с основным алгоритмом. Это довольно распространенная ситуация. Она заслуживает отдельного обсуждения. Понятно, что основной алгоритм "обращается" к вспомогательному не просто так, а ради получения результатов. А они далеко не всегда пригодны для дальнейшего использования исполнителем алгоритма. Например, как мы видели, результатом вспомогательного алгоритма может быть не только число, но и какое-либо сообщение. У других исполнителей бывают и еще более "неудобные" результаты (например, звуковые сигналы или рисунки).

Программисты давно придумали выход: в таких случаях обычно вводят специальную переменную и ее значениями заменяют (кодируют) "неудобные" результаты. Эта переменная называется **сигнальной**. Каждому из возможных результатов соответствует свое значение сигнальной переменной. Например, если алгоритм предназначен для проверки какого-либо условия, то его результатом будет одно из сообщений "да" или "нет". Чтобы сделать этот алгоритм пригодным для использования в других алгоритмах, можно ввести сигнальную переменную, принимающую одно из двух значений. Например, сообщение "да" кодируется числом 0, а сообщение "нет" — числом 1.

Вернемся к задаче о лодках. Обозначим сигнальную переменную буквой  $k$ . Она будет равна 0, если уравнение имеет корни, и 1 — в противном случае. Вот как запишется вспомогательный алгоритм "РЕКВУР" с использованием сигнальной переменной:

**Алгоритм "РЕКВУР".**

**Аргументы:**  $a, b, c$  (коэффициенты квадратного уравнения).

Результаты:  $x$ ,  $y$  (корни уравнения) и  $k$  (сигнальная переменная).

Присвоить дискриминанту  $d$  значение  $b^2 - 4ac$ .

Если  $d < 0$ , то:

Присвоить сигнальной переменной  $k$  значение 1.

Иначе:

Присвоить сигнальной переменной  $k$  значение 0.

Присвоить  $x$  значение  $(-b - \sqrt{d}) / (2a)$ .

Присвоить  $y$  значение  $(-b + \sqrt{d}) / (2a)$ .

Конец ветвления.

В основной программе после каждого вызова "РЕКВУРА" надо проверять, чему равно  $k$ . Если  $k$  оказалось равным 1, то ВЫЧИСЛИТЕЛЬ должен сообщить, что лодка не доплыла до озера (выбыла из соревнования), и остановиться.

Измените программу и запустите ее при тех же значениях исходных данных. Все в порядке? Тогда при тех же значениях скоростей и ускорений возьмите  $S=100$  м. ВЫЧИСЛИТЕЛЬ сообщит вам, что первой придет первая лодка. Это неправда! Разберитесь, в чем ошибка, и исправьте программу.

## § 21. ЗАДАЧА О ПРОИЗВОДСТВЕ ВАКЦИНЫ

Можем ли мы ждать милостей от природы? Не лучше ли взять их у нее самим, да побольше и побыстрее? Еще совсем недавно многие считали, что ждать некогда, бери, сколько хочешь, — природа все стерпит. Оказалось, однако, что это — заблуждение, часто оборачивающееся трагедией. Вырубаются леса, гибнут черноземы, исчезают целые виды животных (вспомните хотя бы судьбу зубров) — все это результат игнорирования важнейшего закона: нельзя брать у природы больше, чем она способна воспроизвести. Продemonстрируем действие этого закона на примере следующей задачи.

**Задача.** Для производства вакцины на заводе планируется выращивать культуру бактерий. Известно, что если масса бактерий —  $x$  г, то через день она увеличится на  $(a - bx)x$  г, где коэффициенты  $a$  и  $b$  зависят от вида бактерий. Завод ежедневно будет забирать для нужд производства  $m$  г бактерий. Для составления плана важно знать ответ на вопрос: как изменяется масса бактерий через 1, 2, 3, ..., 365 дней (до конца года)? Ответьте на этот вопрос.

По сути дела, все упрощающие предположения высказаны в условии задачи. Там же фактически перечислены исходные данные: коэффициенты  $a$  и  $b$ , масса бактерий, забираемых для нужд производства  $m$ , начальная масса бактерий  $x_0$ . Результатами являются значения массы бактерий через 1, 2, 3, 4, 5 ... дней. Мы будем обозначать эти зна-



чения  $x_1, x_2, \dots$ . Указать общую формулу для определения  $x_t$  через  $t$  сложно. Да она и не нужна. Ведь если известно значение  $x_t$ , то легко определить  $x_{t+1}$ :

$$x_{t+1} - x_t = (a - bx_t)x_t - m,$$

т. е.

$$x_{t+1} = x_t + (a - bx_t)x_t - m.$$

В частности, через сутки масса бактерий будет равна

$$x_1 = x_0 + (a - bx_0)x_0 - m.$$

Затем по  $x_1$  вычисляется  $x_2$ , по  $x_2$  вычисляется  $x_3$  и т. д. Правда, очередное значение  $x$  может стать нулем или даже отрицательным числом. Что же делать в таком случае? Надо, конечно, сигнализировать о ЧП, "останавливать производство" и тщательно анализировать причины создавшейся ситуации.

Эти соотношения и определяют связь между исходными данными и результатами.

Как видите, хотя в этой задаче нам и не удалось записать общую формулу членов последовательности, мы тем не менее можем вычислить любой ее член, зная предыдущие. Такой способ задания последовательностей называется **рекуррентным**. На вид несколько необычный, он часто оказывается более простым, чем вычисление по формуле общего члена последовательности.

Компьютерная модель построена. Составим алгоритм, оформив в виде вспомогательного алгоритма вычисление очередного значения  $x$  (рис. 40).

**Алгоритм "Вычисление следующего значения массы".**

**Аргументы:**  $y, a, b, m$ .

**Результат:**  $x$ .

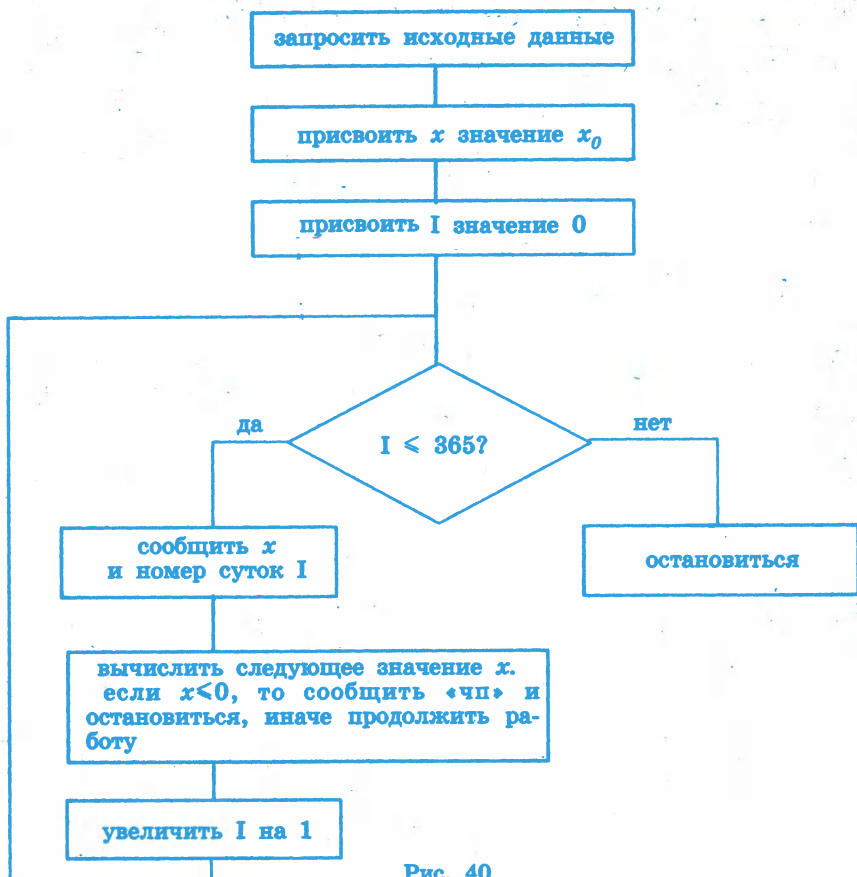


Рис. 40

$$x = y + (a - by)y - m.$$

Если  $x \leq 0$ , то:

Сообщить "ЧП. Бактерии погибли!".

Стоп.

Конец ветвления.

Теперь запишем основной алгоритм (начальную массу бактерий мы обозначили через  $p$ ):

Сообщить "Введите коэффициенты  $a$ ,  $b$  и массу  $m$ ".

Запросить  $a$ ,  $b$ ,  $m$ .

Сообщить "Введите начальное значение массы бактерий".

Запросить начальное значение массы бактерий  $p$ .

Присвоить  $x$  значение  $p$ .

Присвоить номеру года  $i$  значение 0.

Пока  $i \leq 365$ , повторять:

Сообщить "Значения  $x, i$ ".

Сообщить  $x, i$ .

Выполнить алгоритм "Вычисление следующего значения массы" при  $y=x$  и имеющихся значениях  $a, b, m$ .

Присвоить  $i$  значение  $i+1$ .

Конец цикла.



### Задания для самостоятельного выполнения

1°. ЧЕРТЕЖНИК находится в углу квадратного листа бумаги. Составьте алгоритм рисования "лесенки", соединяющей начальное положение ЧЕРТЕЖНИКА с противоположным углом листа (рис. 40).

2°. ЧЕРТЕЖНИК находится в некоторой точке квадратного листа бумаги. Используя алгоритмы решений задачи 3 из § 17 и задачи 1 данного параграфа как вспомогательные, составьте алгоритм изображения лесенки на рисунке 41.

3. Чему будут равны переменные  $x, y$  после выполнения ВЫЧИСЛИТЕЛЕМ следующих алгоритмов?

а) Присвоить  $x$  значение 2.

Присвоить  $y$  значение 1.

Пока  $x > y$ , повторять:

Выполнить алгоритм "РЕКВУР" при  $a=1, b=x+y, c=xy$ .

Если  $k=0$ , то:

Сообщить "Что-то не то! Прекращаю работу".

Стоп.

Конец ветвления.

Присвоить  $x$  значение  $-x$ .

Конец цикла.

б) Присвоить  $u$  значение 2.

Присвоить  $v$  значение 1.

Пока  $u > v$ , повторять:

Выполнить алгоритм "РЕКВУР" при  $a=1, b=uv, c=u+v$ .

Если  $k=0$ , то:

Присвоить  $u$  значение  $u+1$ .

Присвоить  $v$  значение  $v+1$ .

Иначе:

Присвоить  $u$  значение  $x$ .

Присвоить  $v$  значение  $y$ .

Конец ветвления.

Конец цикла.

в) Присвоить  $u$  значение 3.

Присвоить  $v$  значение 1.

Пока  $u > v$ , повторять:

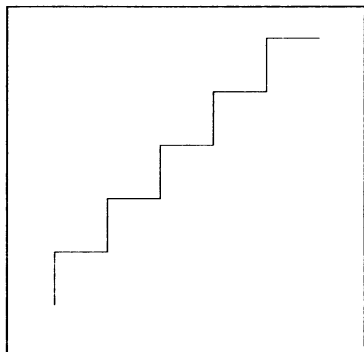


Рис. 41

Выполнить алгоритм "РЕКВУР" при  $a=1$ ,  $b=uv$ ,  $c=u+v$ .  
Выполнить алгоритм "Поиск максимума из двух чисел"  
при  $x=k$ ,  $y=0,5$ .  
Присвоить  $v$  значение  $v+z$ .  
Конец цикла.

4°. Пользуясь алгоритмом для нахождения остатка от деления одного числа на другое (см. задачу 11 из § 17) как вспомогательным, составьте для ВЫЧИСЛИТЕЛЯ следующие алгоритмы:

а) алгоритм, позволяющий определить, делится ли одно число на другое;

б) алгоритм, позволяющий определить, является ли число степенью двойки;

в) алгоритм, позволяющий найти наименьший делитель данного числа (отличный от 1);

г) алгоритм, позволяющий определить, является ли число простым (число называется простым, если оно отлично от единицы и делится только на себя и на единицу);

д) алгоритм, позволяющий определить, является ли число совершенным (число называется совершенным, если оно равно сумме всех его делителей, отличных от него самого).

5. Составьте алгоритм нахождения площадей фигур методом Монте-Карло (см. § 18), в котором проверка того, что точка попала внутрь фигуры, была бы вспомогательным алгоритмом.

6. Составьте алгоритм решения следующей задачи:

На острове живут зайцы и волки. Экологи установили такую закономерность: если в начале года количество зайцев равно  $x$ , а количество волков —  $y$ , то через год число зайцев будет равно  $x+(4-0,001y-0,0001x)x$ , а число волков будет равно  $y+(-0,03+0,003x)y$ . Сколько зайцев и волков будет на острове через год, два, три и т. д.? При каких начальных значениях  $x$  и  $y$  на острове исчезнут волки или зайцы?



## Лабораторная работа 12

### РЕШЕНИЕ ЗАДАЧ С ИСПОЛЬЗОВАНИЕМ ВЕТВЛЕНИЙ, ЦИКЛОВ И ВСПОМОГАТЕЛЬНЫХ АЛГОРИТМОВ

Эту лабораторную работу мы начнем с исполнения ЧЕРТЕЖНИКОМ программы по расчерчиванию листа бумаги на клетки со стороной 1 см (см. задачу 4 из § 20). Введите программу в ЭВМ, запустите ее и проверьте, хорошо ли ЧЕРТЕЖНИК расчертил предложенный вам на экране дисплея лист бумаги.

Введите в ЭВМ программу решения задачи о производстве вакцины и запустите ее при  $a = 1$ ,  $b = 0,0001$ ,  $m = 2000$  и  $x_0 = 12\ 000$ . Если программа правильная и **ВЫЧИСЛИТЕЛЬ** действительно напечатает на экране 365 результатов, то у вас, наверно, зарядит в глазах. И уж, конечно, вряд ли вы запомните все результаты вычислений. Подобная ситуация вам уже встречалась при решении экологической задачи. Тогда мы вышли из положения, ограничившись печатью лишь каждого тридцатого результата (концентраций вредных веществ через 30, 60, 90 дней и т. д.).

Сегодня мы поступим иначе. Сначала посмотрим, как изменялась масса бактерий первые десять дней, затем — вторые десять дней и т. д. Для этого изменим программу сначала так, чтобы  $i$  изменялось от 0 до 10, и запустим ее. Затем изменим программу так, чтобы  $i$  изменялось от 11 до 20, снова запустим ее и т. д. После каждой остановки программы у вас будет время запомнить результаты и даже записать их в тетрадь.

Запишите первые десять результатов и изобразите их на графике, откладывая по оси абсцисс количество дней, а по оси ординат массу бактерий.

Видно, что масса бактерий достаточно быстро убывает и становится близкой к 7236 г. А что будет дальше: не снизится ли количество бактерий до нуля? Продолжите получение результатов, пока не убедитесь в том, что масса бактерий "стабилизируется" на уровне, примерно равном 7236 г.

Используя аппарат высшей математики, можно строго доказать, что и в самом деле существует предельное значение, к которому стремится последовательность значений массы бактерий. Оно равно  $5000(1+\sqrt{0,2})$ . Попробуйте доказать это.

Таким образом, к концу года на заводе будет примерно 7236 г бактерий. Попробуем увеличить начальную массу бактерий. Быть может, тогда масса бактерий к концу года тоже увеличится? Проведите эксперимент, взяв начальную массу 13 000 г, 14 000 г, 17 000 г. Нарисуйте соответствующие графики зависимости массы бактерий от количества дней.

Увы! Наши надежды не оправдались: к концу года масса бактерий каждый раз упорно стремится к 7236 г. Если же вы возьмете еще большую начальную массу, скажем, 18 000 г, то уже через два дня... Запустите программу и посмотрите сами, что произойдет. Типичный случай, когда пословица "Кашу маслом не испортишь" неприемима.

Всякий результат, даже отрицательный, полезен. Наш компьютерный эксперимент наводит на мысль о том, что для получения к концу года все тех же 7236 г можно было в начале года взять совсем немного бактерий. Попробуйте сами уменьшить начальную массу бактерий (бактерии стоят недешево, и закупать лишние бактерии — бросать деньги на ветер). Какое наименьшее значение массы вам удалось получить? (Здесь имеет смысл снова применить метод деления пополам.)

Какие же выводы можно сделать? Существует такой интервал значений начальной массы, при которых к концу года масса бактерий стабилизируется на уровне 7236 г. Если же взять начальную массу за пределами этого интервала, то бактерии погибнут.

Если у вас осталось время, попробуйте провести еще один эксперимент: определить, какую наибольшую массу  $m$  можно ежедневно забирать, чтобы завод работал бесперебойно в течение года.

## § 22. ПОСЛЕДОВАТЕЛЬНОЕ ПОСТРОЕНИЕ АЛГОРИТМОВ

Вы уже убедились в том, что выделение вспомогательных алгоритмов — мощное средство, облегчающее решение сложных задач. Но использовать его можно не только так, как мы делали это до сих пор, выделяя вспомогательные алгоритмы из практически уже готовых алгоритмов. Гораздо более эффективным является другой метод, который называется **методом пошаговой детализации** или **последовательного построения алгоритмов**. Об этом методе и пойдет речь.

Давайте составим для **ЧЕРТЕЖНИКА** алгоритм рисования на бесконечном листе бумаги слова **РОБОТ** — рисунок 42 (высота каждой буквы — 4 см, ширина — 1 см, расстояние между буквами — 1 см).

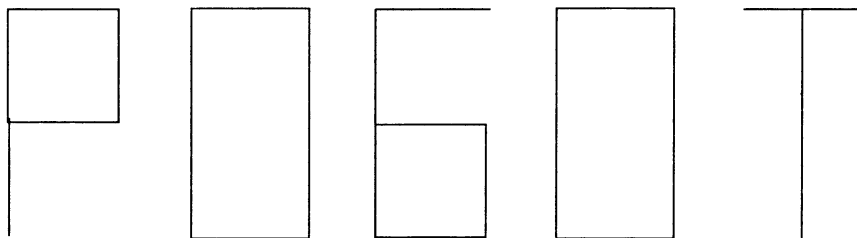


Рис. 42

Попробуем сначала записать весь алгоритм с ходу:

Сделать шаг.  
Сделать шаг.  
Сделать шаг.  
Сделать шаг.  
Повернуть налево.  
Повернуть налево.  
Повернуть налево.  
Сделать шаг.  
Повернуть налево.  
Повернуть налево.  
Повернуть налево.  
Сделать шаг.  
Сделать шаг.  
Повернуть налево.  
Повернуть налево.  
Повернуть налево.  
Сделать шаг.  
Повернуть налево.  
Прыгнуть.  
Прыгнуть.

...

Уф!.. А ведь мы всего лишь описали, как рисовать букву Р. Если продолжать в том же духе, то получится очень длинный и нерациональный алгоритм. К тому же он наверняка будет содержать ошибки, которые трудно обнаружить, глядя на эту запись.

Традиционный метод составления алгоритмов в данном случае, как видите, малоэффективен. Попробуем иначе подойти к решению задачи. Ясно, что ЧЕРТЕЖНИК должен последовательно нарисовать буквы Р, О, Б, О, Т. Таким образом, наша задача разбивается на пять этапов (рис. 43):

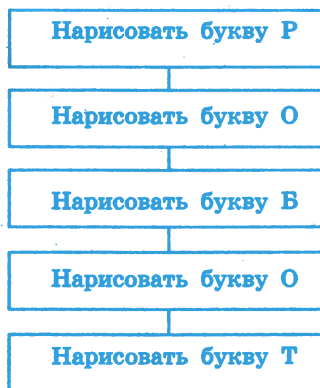


Рис. 43

Считая, что вспомогательные алгоритмы рисования букв уже составлены, запишем сразу основной алгоритм:

Выполнить алгоритм Р.  
Выполнить алгоритм О.  
Выполнить алгоритм Б.  
Выполнить алгоритм О.  
Выполнить алгоритм Т.

Теперь можно приступить к составлению вспомогательных алгоритмов. При этом нужно позаботиться о двух вещах. Во-первых, чтобы ЧЕРТЕЖНИК правильно нарисовал отдельные буквы, а во-вторых, чтобы, выполнив основной алгоритм, он сложил из них слово РОБОТ. Иначе говоря, надо заботиться не только о рисовании самих букв, но и об их "стыковке". Это напоминает строительство дома: надо заботиться не только о качестве кирпичей, но и о том, чтобы между ними не оставалось щелей. Таким образом, основной алгоритм предъявляет дополнительные требования к "стыковке" вспомогательных алгоритмов.

В нашем случае эти требования могут быть, например, такими. Будем считать, что ЧЕРТЕЖНИК начинает рисование каждой буквы с ее левой нижней точки, глядя вправо. Каждый из вспомогательных алгоритмов должен выводить ЧЕРТЕЖНИКА в исходную позицию для рисования следующей буквы.

Записав основной алгоритм и определив требования к вспомогательным алгоритмам, мы "спустились" на более низкий уровень в решении задачи, упростили себе дальнейшую работу. Однако алгоритмы рисования отдельных букв тоже не так уж просты (вспомните нашу попытку написать алгоритм рисования буквы Р). Значит, составление каждого из них также целесообразно разбить на несколько этапов, "спустившись" еще ниже. И так далее, до тех пор, пока задачи очередного уровня не окажутся совсем простыми. Итак, продолжим "спуск".

*Лучше  
две простые задачи,  
чем одна сложная —  
пользуйтесь методом  
пошаговой  
детализации!*

Легко обнаружить общий элемент у букв Р, О и Б. Назовем его "угол" (рис. 44). Вы без труда напишете вспомогательный алгоритм рисования "угла". Воспользовавшись этим алгоритмом, а также алгоритмом "Направо" (см. задачу 3 из § 20), можно записать каждый из пяти вспомогательных алгоритмов. Вот, например, вспомогательный алгоритм "Р":

Рис. 44

Выполнить вспомогательный алгоритм "Угол".  
 Выполнить вспомогательный алгоритм "Направо".  
 Сделать шаг.  
 Сделать шаг.  
 Выполнить вспомогательный алгоритм "Направо".  
 Сделать шаг.  
 Налево.  
 Прыгнуть.  
 Прыгнуть.  
 Налево.  
 Прыгнуть.  
 Прыгнуть.

Остальные вспомогательные алгоритмы напишите самостоятельно.

*Метод, при котором сложная задача разбивается на несколько более простых, получившиеся задачи сводятся к еще более простым и так далее, называется методом пошаговой детализации.* Этот метод универсален. Он всегда применяется в тех случаях, когда требуется спроектировать сложный объект, будь то большой алгоритм, прокатный стан, интегральная микросхема или пятилетний план. Да и вообще решение каждой более или менее сложной задачи проводится методом пошаговой детализации.

## ? Вопросы

1. В чем состоит метод пошаговой детализации алгоритмов?
2. Какого рода требования предъявляет основной алгоритм к вспомогательным?

## 📖 Задания для самостоятельного выполнения

1. Для рисования квадрата, изображенного на рисунке 45, был составлен следующий алгоритм:

Выполнить алгоритм "Уголок".

Выполнить алгоритм "Уголок".

Сформулируйте требования к вспомогательному алгоритму "Уголок" и запишите его.

2. Используя метод последовательной детализации, выделите вспомогательные алгоритмы, необходимые для того, чтобы ЧЕРТЕЖНИК нарисовал слово "РЕБУС" (рис. 46). Составьте соответствующий алгоритм.

3. ЧЕРТЕЖНИК помещен на квадратный лист бумаги, сторона которого длиннее

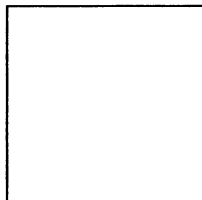


Рис. 45

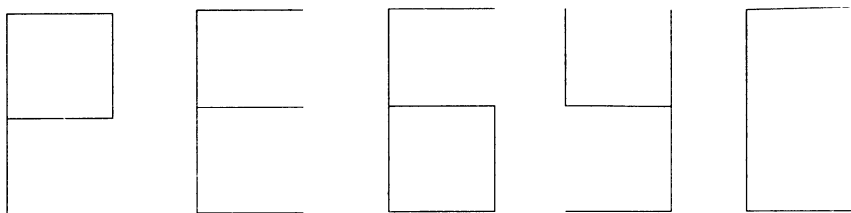


Рис. 46

3 см. Составьте для **ЧЕРТЕЖНИКА** алгоритм рисования квадрата со стороной 1 см.

4. **ЧЕРТЕЖНИК** помещен на прямоугольный лист бумаги. Составьте для **ЧЕРТЕЖНИКА** алгоритм рисования квадрата наибольшей возможной площади (воспользуйтесь алгоритмом решения задачи 2 к § 21).

5°. Составьте для **ВЫЧИСЛИТЕЛЯ** алгоритм разложения числа на простые множители (используйте ранее составленные алгоритмы в качестве вспомогательных).

6°. Составьте для **ВЫЧИСЛИТЕЛЯ** алгоритм решения уравнения  $ax^2+bx+c=0$  (коэффициенты  $a$ ,  $b$ ,  $c$  могут равняться нулю), используя в качестве вспомогательных алгоритмы нахождения корней квадратного и линейного уравнений.

7°. Составьте для **ВЫЧИСЛИТЕЛЯ** алгоритм, с помощью которого он из трех данных чисел выберет то, которое находится между двумя другими.

8. Составьте для себя расписание на неделю вперед методом пошаговой детализации.



### Лабораторная работа 13

#### РЕШЕНИЕ ЗАДАЧ С ПОМОЩЬЮ ПОШАГОВОЙ ДЕТАЛИЗАЦИИ

Вы уже довольно долго изучаете информатику и составили за это время, наверное, много алгоритмов. При этом каждый из вас составлял алгоритмы от начала до конца в одиночку (надеюсь, что вы не списывали домашних заданий, а выполняли их самостоятельно). Но время программистов-одиночек прошло безвозвратно. Для решения сложных задач требуются организованные усилия крупных вычислительных центров. Задачу, которую мы сегодня вам предложим, вы будете решать коллективно. Не потому, что она очень сложна, а просто для того, чтобы научиться совместной работе.

**Задача.** Нарисовать с помощью **ЧЕРТЕЖНИКА** одно из следующих слов: **СПОСОБ, ТРЕПЕТ, ФОСФОР, НАГАН, АППАРАТУРА.**

Программу рисования каждого слова естественно создавать методом пошаговой детализации (вспомните, например, как создавался алгоритм рисования слова **РОБОТ**).

Разделитесь на пять групп (по числу слов), каждая из которых будет рисовать свое слово: три группы по пять человек, одна группа из четырех и одна группа из шести человек. Если работников не хватает, ограничьтесь меньшим количеством слов.

Подпрограмму рисования одной буквы пишет и отлаживает один человек, и еще один человек (руководитель) создает основную программу рисования слова, из которой будут вызываться эти подпрограммы.

Выберите в каждой группе руководителя. Он должен:

1) выдать каждому подчиненному четкое задание (какую букву рисовать, каковы исходное и конечное положения **ЧЕРТЕЖНИКА**);

2) после создания подпрограмм указать подчиненным, с каких номеров должны начинаться их подпрограммы (подпрограммы должны нумероваться одна за другой);

3) написать основную программу рисования слова;

4) записать подпрограммы подчиненных на дискету (магнитную ленту), составить из них и основной программы единую программу;

5) отладить программу и получить результат.

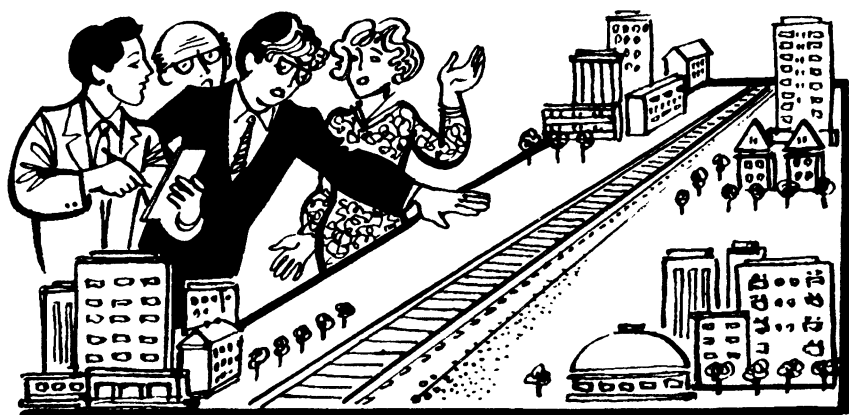
Учитель поможет вам записать программу на дискету (ленту) и потом вызвать ее в оперативную память машины.

В обязанности подчиненного входит написание и отладка подпрограммы в точном соответствии с заданием руководителя. Как видите, работа у руководителя сложнее и ответственнее, чем у подчиненных. Если ваш руководитель не справляется со своими обязанностями, переизберите его.

## **§ 23. ЗАДАЧА О ВЫБОРЕ МЕСТА ДЛЯ СТРОИТЕЛЬСТВА ЖЕЛЕЗНОДОРОЖНОЙ СТАНЦИИ**

Помните, в поэме **Н.А. Некрасова** описано, как собрались мужики окрестных деревень обсудить, кому на Руси жить хорошо? В те бесправные времена они ничего не могли сделать, чтобы облегчить свою жизнь: все социальные вопросы за них решали помещики. Теперь все иначе. Ни один социальный вопрос не решается без широкого обсуждения.

Вот одна из задач, с которыми часто приходится сталкиваться тем, кто по долгу службы печется о благе населения.



**Задача.** В одном районе расположены четыре населенных пункта. По территории района проходит железная дорога. По просьбе жителей района планируется построить железнодорожную станцию и проложить дороги от нее до каждого населенного пункта. Требуется определить наиболее удобное расположение железнодорожной станции.

Как всегда, начнем с построения компьютерной модели. Прежде всего допустим, что участок дороги, проходящий по территории района, прямолинеен и в любом месте участка можно построить станцию и соединить ее прямолинейными дорогами с каждым населенным пунктом (рис. 47). Нарисуем оси координат на карте района так, чтобы ось абсцисс проходила по интересующему нас участку железной дороги, а начало координат совпадало с его левым концом. Населенные пункты будем изображать кружочками, а названия позаимствуем у Некрасова (хотя, конечно, эти деревни давно переименованы).

○ ЗАПЛАТОВО

○ ДЫРЯВИНО

○ ГОРЕЛОВО

---

○ НЕУРОЖАЙКА

Рис. 47

Теперь надо уяснить смысл слов "наиболее удобное расположение станции". Это можно понимать по-разному. Если стремиться к экономии средств на строительство дорог, то станцию надо расположить так, чтобы сумма длин дорог, соединяющих станцию с населенными пунктами, была наименьшей. Если же стремиться к максимальной справедливости, то место для станции надо выбрать так, чтобы наибольшее из расстояний от нее до населенных пунктов было как можно меньше. Например, если в самом дальнем от станции населенном пункте заболел человек, то его надо доставить на станцию за самое короткое время. Мы за справедливость (как и поэт-демократ Н. А. Некрасов). Поэтому выберем второй принцип.

Определим исходные данные и результат для нашей модели. Исходными данными, конечно, являются координаты населенных пунктов и длина  $S$  отрезка, изображающего на карте участок железной дороги, пролегающий по району. Результат — абсцисса точки, где будет строиться станция.

Остается найти соотношения между результатом и исходными данными. Координаты наших четырех населенных пунктов обозначим  $(a, b)$ ,  $(c, d)$ ,  $(e, f)$ ,  $(g, h)$ .

Возьмем любую точку на отрезке, изображающем железную дорогу. Пусть  $t$  — ее абсцисса. Через  $z$  обозначим максимальное из расстояний между этой точкой и каждым из населенных пунктов. Иначе говоря,

$$z = \max\{\sqrt{(t-a)^2+b^2}; \sqrt{(t-c)^2+d^2}; \sqrt{(t-e)^2+f^2}; \sqrt{(t-g)^2+h^2}\}.$$

Чтобы определить, где построить станцию, надо узнать, при каком  $t$  из отрезка  $[0; S]$  переменная  $z$  принимает наименьшее значение. Как же найти это  $t$ ? Ведь мы не можем перебрать все числа от 0 до  $S$ : их бесконечно много. Выход один — искать приближенное значение  $t$ . Для этого разобьем отрезок  $[0; S]$  на равные части, длину каждой из них обозначим буквой  $r$ . Затем найдем значения  $z$  при  $t=0, r, 2r, \dots$  и так далее, пока  $t \leq S$ . Выберем из этих значений наименьшее и определим, при каком  $t$  оно достигается. Это значение  $t$  будет отличаться от искомого результата не больше чем на  $r$ . Поэтому мы возьмем его в качестве результата.

Вообще говоря, не для каждой функции полученное таким методом значение  $t$  действительно будет отстоять от точки минимума менее чем на  $r$  (попробуйте привести пример такой функции). Однако можно строго доказать, что в данном случае это верно.

Итак, для решения задачи нам потребуется из достаточно большого количества чисел найти минимальное. Чтобы лучше понять, как это делать, представьте себе, что

ваш класс в алфавитном порядке вызывают на медосмотр, а любознательный врач задался целью выявить самого легкого ученика. Скорее всего врач поступит так. Вызвав первого ученика, он запишет его вес и фамилию. Пригласив следующего, он сравнит его вес с записанным числом. Если этот ученик оказался легче, врач зачеркнет предыдущую запись и запишет сведения о новом ученике, в противном случае сохранится прежняя запись. Затем он вызовет следующего ученика и т. д. Когда пройдет весь класс, на листке у врача останутся фамилия и вес самого легкого ученика.

При решении нашей задачи мы будем поступать точно так же. В роли учеников будут выступать значения  $t$  (т. е. числа  $0, r, 2r$  и т. д.), а взвешивание заменим вычислением числа  $z$ . Остается понять, что будет играть роль листка, на который врач записывал фамилию и вес ученика. Ведь ВЫЧИСЛИТЕЛЬ "не умеет" пользоваться бумагой и ручкой. ВЫЧИСЛИТЕЛЬ может "записать" (запомнить) число единственным способом — присвоить его какой-нибудь переменной. В нашем случае потребуются две переменные: придется "запоминать" числа  $t$  и  $z$ . Для запоминания  $t$  приспособим переменную  $V$ , а для запоминания  $z$  — переменную  $M$ . По окончании выполнения алгоритма значение переменной  $V$  будет указывать искомое место расположения станции, а  $M$  — расстояние от этого места до самого удаленного из населенных пунктов.

Теперь мы можем полностью представить себе, что должен делать ВЫЧИСЛИТЕЛЬ, решая эту задачу. Сначала он положит  $t$  равным  $0$  и подсчитает  $z$ ; это  $z$  он примет за начальное значение  $M$ , а  $0$  — за начальное значение  $V$ . Затем ВЫЧИСЛИТЕЛЬ подсчитает  $z$  при  $t=r$  и сравнит его с  $M$ . Если  $z$  окажется меньше  $M$ , он присвоит  $M$  это значение  $z$ , а  $V$  — значение  $t$ , в противном случае оставит  $M$  и  $V$  неизменными и т. д., пока не дойдет до значения  $t=S$ .

Ясно, что поиск  $z$  при данном  $t$  лучше всего оформить в виде вспомогательного алгоритма. Запишем этот алгоритм.

**Алгоритм "Нахождение максимального из расстояний до населенных пунктов".**

**Аргументы:**  $a, b, c, d, e, f, g, h$  (координаты населенных пунктов) и  $t$  (координата точки на отрезке  $[0; S]$ ).

**Результат:**  $z$  (максимальное из расстояний до населенных пунктов).

**Выполнить алгоритм "Поиск максимума из двух чисел"** при  $x=\sqrt{(a-c)^2+b^2}$ ,  $y=\sqrt{(c-t)^2+d^2}$ .

**Выполнить алгоритм "Поиск максимума из двух чисел"** при  $x=z$ ,  $y=\sqrt{(e-t)^2+f^2}$ .

Выполнить алгоритм "Поиск максимума из двух чисел" при  $x=z$ ,  $y=\sqrt{(g-t)^2+h^2}$ .

Основной алгоритм запишется так:

Запросить координаты населенных пунктов  $a, b, c, d, e, f, g, h$ .

Запросить длину  $S$  участка железнодорожного пути и длину  $r$  отрезка разбиения.

Присвоить координате  $t$  значение 0.

Выполнить алгоритм "Нахождение максимального из расстояний до населенных пунктов", взяв в качестве аргументов  $a, b, c, d, e, f, g, h$  и  $t$ .

Присвоить  $M$  начальное значение, равное  $z$ .

Присвоить координате  $V$  искомой точки начальное значение, равное  $t$ .

Пока  $t < S$ , повторять:

Присвоить  $t$  значение  $t+r$ .

Выполнить алгоритм "Нахождение максимального из расстояний до населенных пунктов", взяв в качестве аргументов  $a, b, c, d, e, f, g, h$  и  $t$ .

Если  $z < M$ , то:

Присвоить  $M$  очередное значение, равное  $z$ .

Присвоить  $V$  очередное значение, равное  $t$ .

Конец ветвления.

Конец цикла.

Сообщить "Координата искомой точки и максимальное расстояние от нее до населенных пунктов".

Сообщить  $V, M$ .

Этот алгоритм позволяет найти решение задачи приближенно. Однако можно отыскать и точное решение этой задачи. Найдите его самостоятельно!



### Задания для самостоятельного выполнения

1°. Составьте компьютерную модель и алгоритм решения задачи о железнодорожной станции:

а) руководствуясь первым принципом выбора наиболее удобного расположения станции (минимум суммы расстояний);

б) руководствуясь компромиссным принципом: если  $z$  — расстояние от станции до наиболее удаленного пункта, а  $w$  — сумма расстояний до всех пунктов, то требуется сделать как можно меньшим среднее арифметическое из  $z$  и  $w$ .

ЗАДАЧА О ВЫБОРЕ МЕСТА ДЛЯ  
СТРОИТЕЛЬСТВА ЖЕЛЕЗНОДОРОЖНОЙ СТАНЦИИ

Начнем с того, что поручим ВЫЧИСЛИТЕЛЮ выбрать наилучшее расположение железнодорожной станции. Введите программу и запустите ее при  $a=-3$ ,  $b=8$ ,  $c=5$ ,  $d=7$ ,  $e=-2$ ,  $f=-3$ ,  $g=12$ ,  $h=-4$ ,  $S=10$ ,  $r=0,1$ . Запишите полученный результат, чтобы не забыть.

После этого введите программу нахождения выбора места расположения станции, при котором стоимость строительства дорог будет наименьшей (см. задачу 1,а из § 23). Запустите ее при тех же значениях координат населенных пунктов.

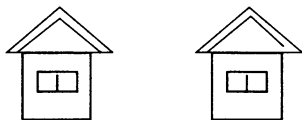
Давайте обсудим получившиеся результаты. Сравним стоимости строительства дорог в каждом из этих случаев. Подсчитайте с помощью ЭВМ, сколько придется затратить средств на строительство дорог, считая, что на 1 км дороги тратится 1,5 млн. рублей. Не правда ли, получилась внушительная сумма! Однако представьте себе, что "скорая помощь" должна доставить больного из наиболее удаленного населенного пункта на станцию. Средняя скорость машины — 80 км/ч. Определите, насколько быстрее прибудет "скорая помощь" в первом случае, чем во втором. При этом во втором случае вам потребуется найти расстояние от станции до наиболее удаленного населенного пункта. Составьте соответствующую программу. Проанализируйте полученные результаты.

В решении задачи, мы надеемся, вам все ясно. Но столь значительное расхождение результатов в зависимости от понимания слов "наиболее удобное расположение" кажется необъяснимым и способно вызвать недоумение. Чтобы разобраться, в чем тут дело, построим другую компьютерную модель, несколько упростив при этом задачу. Мы будем рассматривать лишь два населенных пункта, расположенные по разные стороны от железной дороги. Остальные предположения остаются прежними, только при построении модели мы будем ориентироваться не на ВЫЧИСЛИТЕЛЯ, а на графический редактор.

Определим исходные данные. Населенные пункты на экране ЭВМ будем изображать точками, а участок железной дороги — отрезком горизонтальной прямой. На вашем экране это должно выглядеть примерно так, как изображено на рисунке 48.

Результат — точка на отрезке прямой, обозначающая место, где надо строить станцию.

## ○ ЗАПЛАТОВО



Б

А

○ НЕУРОЖАЙКА

Рис. 48

Если руководствоваться первым толкованием слов "наибольшее удобство", то эта точка будет пересечением прямой, проведенной через "населенные пункты", с прямой, изображающей железную дорогу (ведь отрезок прямой всегда короче ломаной, соединяющей концы этого отрезка!). Тем самым мы определили связь между исходными данными и результатом.

Алгоритм решения задачи в этом случае состоит в том, чтобы провести отрезок, соединяющий точки, изображающие населенные пункты. Как это сделать, вы, конечно, помните: надо поместить курсор в точку, где отрезок должен начинаться, и нажать на функциональную клавишу "Отрезок", затем переместить курсор в точку, где отрезок должен кончиться, и снова нажать на клавишу "Отрезок".

Исполните этот алгоритм. Получившаяся на экране вашего компьютера точка пересечения отрезков — ответ к задаче. (Некоторые графические редакторы выводят координаты точки, в которой стоит курсор; с помощью такого редактора можно получить ответ задачи, не сильно отличающийся от ответа, который вы бы нашли с помощью ВЫЧИСЛИТЕЛЯ, — достаточно изобразить исходные данные в подходящем масштабе.)

Если населенные пункты удалены от железной дороги неодинаково, то и расстояние до станции от одного из них больше, чем от другого.

Давайте перемещать положение станции по железной дороге в сторону более удаленного населенного пункта. Тогда расстояние до него от станции будет уменьшаться, в то время как от второго пункта до станции расстояние будет увеличиваться. И до тех пор, пока эти расстояния не станут равными, жители более удаленного пункта будут считать нарушенным принцип социальной справедливости. Чтобы станция была одинаково удалена от обоих населен-

ных пунктов, нужно взять точку пересечения железной дороги и серединного перпендикуляра к отрезку, соединяющему населенные пункты (попробуйте обосновать это, используя свойства серединного перпендикуляра к отрезку). Составьте и исполните алгоритм построения соответствующей точки.

Проведите компьютерный эксперимент для этой модели задачи. В частности, расположите населенные пункты на одной вертикальной прямой. В этом случае, очевидно, серединный перпендикуляр будет параллелен железной дороге и ни о какой точке пересечения говорить не приходится. Что означает наша неудача?

Наверно, вы уже догадались: построенная нами модель соответствует исходной задаче не полностью (или, как иногда говорят, не адекватна исходной задаче). В самом деле, искать расположение станции за пределами отрезка, ограниченного проекциями населенных пунктов на железную дорогу, неразумно — ведь тогда путь до станции от каждого населенного пункта будет только длиннее.

Уточним нашу модель, выразив связь между исходными данными и результатом следующим образом:

если серединный перпендикуляр к отрезку, соединяющему населенные пункты, пересекает участок железной дороги, ограниченный проекциями этих населенных пунктов, то точка пересечения искомая; иначе искомой является проекция того населенного пункта, который отстоит от железной дороги на большее расстояние.

Составьте алгоритм, соответствующий этой модели задачи, и исполните его. У вас получится картинка, похожая на рисунок 49.

Сравните точки, полученные по первому и второму алгоритмам (на рисунке 49 они обозначены буквами А и Б). Как видите, точки А и Б достаточно далеки друг от друга.

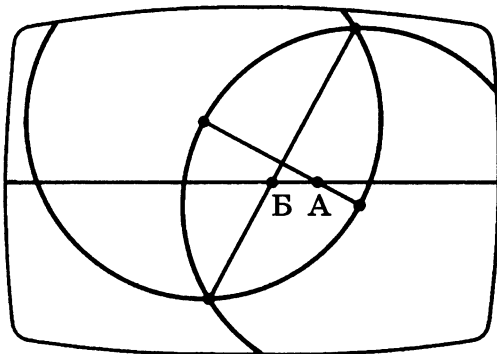


Рис. 49

Если у вас осталось время, решите на ЭВМ задачу 1,6 из § 23. Сравните полученный результат с ранее полученными результатами решения задачи из объяснительного текста § 23 и задачи 1,а.



## КОНСПЕКТ ГЛАВЫ 6

**Вспомогательным** называется алгоритм, снабженный заголовком, позволяющим вызывать этот алгоритм из других алгоритмов. Любой алгоритм можно сделать вспомогательным, снабдив его соответствующим заголовком (т. е. указав название, исходные данные и результаты). Примеры вспомогательных алгоритмов:

**Алгоритм "Скобка".**

**Повернуть налево.**

**Повернуть налево.**

**Сделать шаг.**

**Повернуть налево.**

**Сделать шаг.**

**Повернуть налево.**

**Сделать шаг.**

**Алгоритм "Поиск максимума из двух чисел".**

**Аргументы:**  $x$  и  $y$  — числа, среди которых ищется максимальное.

**Результат:**  $z$  — максимум из чисел  $x$  и  $y$ .

**Если  $x < y$ , то:**

**Присвоить  $z$  значение  $y$ .**

**Иначе:**

**Присвоить  $z$  значение  $x$ .**

**Конец ветвления.**

Вспомогательные алгоритмы создаются тогда, когда возникает необходимость многократного использования одного и того же набора действий в одном или в разных алгоритмах, а также для решения сложных задач, когда задача разбивается на ряд более простых задач (в этом суть метода пошаговой детализации).

Если при составлении какого-либо алгоритма потребовалась "помощь" уже имеющегося вспомогательного алгоритма, достаточно воспользоваться командой вызова вспомогательного алгоритма. В ней указывается название вспомогательного алгоритма, а также значения исходных данных (если они предусмотрены в заголовке вспомогательного алгоритма). Примеры:

**Выполнить алгоритм "Скобка".**

**Выполнить алгоритм "Максимум",** взяв в качестве  $x$  число 25, а в качестве  $y$  — значение выражения  $a^2 + b^2$ .

Составляя для исполнителя вспомогательные алгоритмы, мы как бы обучаем исполнителя новым действиям. В результате этого подробные объяснения того, что нужно делать, можно заменить одной командой.

При использовании вспомогательного алгоритма достаточно знать, что является его исходными данными и результатом. Иногда в основном алгоритме используются те же обозначения переменных, что и во вспомогательном. Поэтому считают, что все обозначения из вспомогательного алгоритма, кроме результатов, действительны только в пределах этого алгоритма. При вызове вспомогательного алгоритма значения переменных основного алгоритма как бы "замораживаются", а после окончания работы вспомогательного алгоритма "размораживаются".

Результаты, ради которых основной алгоритм "обращается" к вспомогательному, не всегда пригодны для дальнейшего использования исполнителем алгоритма (сообщения, например). В таких случаях вводят специальную — **сигнальную** — переменную и ее значениями кодируют результаты. Каждому из возможных результатов соответствует свое значение сигнальной переменной. Например, сообщение "да" может кодироваться числом 0, а сообщение "нет" — числом 1.

Хотя циклов и разилок достаточно, чтобы записать любой алгоритм, вспомогательные алгоритмы — мощное средство, облегчающее решение трудных задач. Можно сказать, что искусство составления алгоритмов заключается в умении конструировать сложный алгоритм из более простых алгоритмов, т. е. в умении обучать исполнителя, идя от простого к сложному.

Вспомогательные алгоритмы можно не только выделять из уже готовых алгоритмов. Гораздо более эффективно используются преимущества вспомогательных алгоритмов в методе пошаговой детализации (последовательного построения) алгоритмов. При этом сложная задача разбивается на несколько более простых. Предполагая, что вспомогательные алгоритмы решения этих задач уже построены, записывается основной алгоритм, состоящий главным образом из команд вызова вспомогательных алгоритмов. Затем определяются требования к этим алгоритмам. Требования диктуются как задачами, решаемыми каждым из вспомогательных алгоритмов, так и необходимостью их стыковки в основном алгоритме. Некоторые из задач, на которые разбита исходная задача, могут оказаться еще слишком сложными. Тогда они в свою очередь разбиваются на более простые задачи и т. д. В результате некоторые вспомогательные алгоритмы могут стать основными по отношению к вспомогательным алгоритмам более низкого уровня. Про-

цесс пошаговой детализации заканчивается, когда задачи очередного уровня окажутся совсем простыми.

Метод пошаговой детализации алгоритмов универсален. Он применим для решения задач из самых разных областей жизни.

**Подпрограммы** — это вспомогательные алгоритмы, записанные на языке, понятном ЭВМ. Оформление подпрограмм отличается от оформления вспомогательных алгоритмов. Вот пример вспомогательного алгоритма и соответствующей подпрограммы для **ВЫЧИСЛИТЕЛЯ** (аналогично оформляются подпрограммы для **ЧЕРТЕЖНИКА**):

**Вспомогательный алгоритм**

**Алгоритм "Поиск максимума из двух чисел".**

**Аргументы:**  $x$  и  $y$ .

**Результат:**  $z$ .

**Если**  $x < y$ , **то:**

**Присвоить**  $z$  значение  $y$ .

**Иначе:**

**Присвоить**  $z$  значение  $x$ .

**Конец ветвления.**

**Подпрограмма**

**20 ПОДПРОГРАММА [X,Y]**

**25 ЕСЛИ X<Y, ТО:**

**30 Z=Y**

**35 ИНАЧЕ:**

**40 Z=X**

**45 КОНЕЦ ВЕТВЛЕНИЯ**

**50 КОНЕЦ ПОДПРОГРАММЫ [Z]**

Таким образом, при оформлении подпрограммы ее имя, а также слова "Аргументы" и "Результаты" надо опускать. Буквы, обозначающие аргументы, записываются в квадратных скобках после слова "ПОДПРОГРАММА". Подпрограмма завершается командой "КОНЕЦ ПОДПРОГРАММЫ". После этих слов в квадратных скобках записывают буквы, обозначающие результаты работы подпрограммы (если, конечно, таковые имеются).

Подпрограммы обычно договариваются располагать после основной программы.

При вызове подпрограммы роль имени играет номер первой строки подпрограммы. Команда вызова подпрограммы начинается со слов **ВЫПОЛНИТЬ ПОДПРОГРАММУ**, после них пишется номер строки, с которой начинается подпрограмма. Затем в квадратных скобках перечисляют выражения, значения которых ЭВМ должна взять в качестве исходных данных.

Вот, например, как выглядит вызов подпрограммы "Поиск максимума ..." при  $x=a+bc$ , а  $y=a/b+c$ :

**5 ВЫПОЛНИТЬ ПОДПРОГРАММУ 20 [A+B\*C,A/B+C]**

# ОРГАНИЗАЦИЯ ДАННЫХ

---

До сих пор, изучая алгоритмы, мы говорили лишь об организации действий, составляющих алгоритм, и почти не касались организации объектов, над которыми эти действия производятся. На самом деле, прежде чем исполнитель начнет действовать, мы должны предоставить ему все, что может понадобиться при выполнении алгоритма, обеспечить фронт работ.

*Данные хороши  
тогда,  
когда они хорошо  
организованы*

Исполнение многих алгоритмов было бы просто невозможно, если бы соответствующие объекты не были каким-либо образом организованы: упорядочены, классифицированы, занумерованы и т. д. Итак, нужно уметь организовывать не только действия, но и те объекты, над которыми эти действия производятся. В дальнейшем эти объекты мы будем называть данными.

### § 24. ТАБЛИЧНЫЙ СПОСОБ ОРГАНИЗАЦИИ ДАННЫХ

В этом параграфе мы расскажем об одном из самых распространенных способов организации данных — табличном. Без преувеличения можно сказать, что с таблицами вы сталкиваетесь ежедневно. Таблица итогов шахматного турнира и классный журнал, таблица умножения на обложке вашей тетради и план кинозала — все это примеры табличной организации данных.

Допустим, что 16 апреля прошлого года 9"а" класс писал сочинение по литературе и нам нужно узнать, какую оценку получил ученик этого класса Иванов И. Берем классный журнал 9"а" класса за прошлый год, открываем его на странице "Литература", находим строку с оценками Иванова И., столбец за 16 апреля. На их пересечении и записана искомая оценка.

После уроков вы пришли в кинотеатр. Как занять место в зале согласно купленному билету? Сначала, двига-

ясь по проходу, ищем свой ряд, а затем идем вдоль него до нужного места.

Нетрудно заметить сходство этих примеров. Хотя школьные оценки на странице классного журнала и места в зрительном зале имеют мало общего, расположены они в принципе одинаково: *в несколько рядов (строк) одинаковой длины*. Такой способ расположения данных и называется **табличным**. Может случиться и так, что в таблице всего одна строка. Такую таблицу называют **линейной**. Таблицу, состоящую из нескольких строк, называют **прямоугольной**.

Чтобы можно было оперировать с элементами линейных и прямоугольных таблиц, надо каким-то образом обозначать таблицы и их элементы. Обычно для обозначения таблиц используют латинские буквы. Договоримся нумеровать строки таблиц сверху вниз, а столбцы слева направо. Элемент, расположенный на  $i$ -м месте линейной таблицы  $A$ , обозначают  $A(i)$ , а элемент, расположенный на пересечении  $i$ -й строки и  $j$ -го столбца прямоугольной таблицы  $X$ , обозначают  $X(i,j)$ . Например, в таблице  $A$

*В таблице  
каждый элемент —  
на своем месте!*

7	2	3	34
4	5	0	-3
1	3	8	1,2

$A(1,1)=7$ ,  $A(2,3)=0$ ,  $A(3,2)=3$ ,  $A(3,4)=1,2$ .

### ? Вопросы

1. Что такое табличная форма организации данных?
2. Какие таблицы называются:
  - а) линейными;
  - б) прямоугольными?
3. Как обозначаются таблицы и их элементы?

### Задания для самостоятельного выполнения

1. Определить, как выглядит таблица  $A$ , состоящая из 4 строк и 5 столбцов, если для любых  $i$  и  $j$ :

а)  $A(i,j)=ij$ ;

б)  $A(i,j)=i$ ;

в)  $A(i,j)=1$ ;

г)  $A(i,j)=\max\{i,j\}$ ;

д)  $A(i,j)=\min\{i,j\}$ ;

е)  $A(i,j)=|i-j|$ ;

ж)  $A(i,j)$  равняется 1, если  $i \leq j$ , и 0, если  $i > j$ .

2. Дана таблица  $B$ . Запишите, как выражается ее элемент  $B(i, j)$  через  $i$  и  $j$ , если таблица имеет вид:

а) 
$$\begin{bmatrix} 2 & 4 & \dots & 2^n \\ 2 & 4 & \dots & 2^n \\ 2 & 4 & \dots & 2^n \\ 2 & 4 & \dots & 2^n \end{bmatrix}$$

(в этой таблице 4 строки и  $n$  столбцов);

б) 
$$\begin{bmatrix} 1 & 2 & 3 & 4 & \dots & n \\ 0 & 2 & 3 & 4 & \dots & n \\ 0 & 0 & 3 & 4 & \dots & n \\ 0 & 0 & 0 & 4 & \dots & n \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & 0 & \dots & n \end{bmatrix}$$

(в этой таблице  $n$  строк и  $n$  столбцов);

в)  $n \ n-1 \ \dots \ 2 \ 1$

(в этой линейной таблице  $n$  столбцов).

3. Приведите пример таблицы  $A$ , состоящей из  $n$  строк и  $m$  столбцов, в которой элементы  $A(i, j)$  и  $A(i, j+1)$  равны для любых  $i \leq n$  и  $j \leq m-1$ .

4. В следующей таблице приведены итоги шахматного турнира:

0	1/2	1	0	1	1/2	0	0
1/2	0	1/2	0	1	0	0	1
0	1/2	0	1	0	1	1	0
1	1	0	0	1/2	1	0	1
0	0	1	1/2	0	1/2	1/2	1/2
1/2	1	0	0	1/2	0	1	1
1	1	0	1	1/2	0	0	0
1	0	1	0	1/2	0	0	0

Сколько очков набрал победитель турнира?

## § 25. ТАБЛИЦЫ И ИСПОЛНИТЕЛИ

Хорошо знакомый вам исполнитель **ЧЕРТЕЖНИК** в своей работе совсем не пользуется таблицами. Поэтому мы продемонстрируем табличный способ организации данных на примере нового для вас исполнителя. Назовем его **РОБОТ-МАНИПУЛЯТОР** или просто **РОБОТ**. На складе, где он работает, имеется большой прямоугольный стеллаж, в ячейках которого находятся ящики с однотипными деталями (микросхемами, транзисторами и т. д.). Робот может перемещаться вдоль стеллажа, не выходя при этом за его пределы. Работа у него несложная: по запросу оператора он ищет нужные детали и, поместив их в свой грузовой отсек, доставляет на рабочее место.

Укажем список допустимых действий РОБОТА-МАНИПУЛЯТОРА.

1. Запросить наименование детали.
2. Переместиться к левой нижней ячейке стеллажа.
3. Переместиться к началу ряда.
4. Переместиться к соседней сверху (снизу, справа, слева) ячейке.
5. а) Переложить деталь из ячейки в грузовой отсек.  
б) Переложить деталь из грузового отсека в ячейку.  
(Ячейка, о которой идет речь в пунктах а) и б), — та, напротив которой находится РОБОТ.)
6. Проверить одно из следующих условий:  
а) можно шагнуть вверх (вниз, вправо, влево) или нельзя;  
б) есть деталь в ячейке, напротив которой находится РОБОТ, или нет;  
в) совпадает наименование детали в ячейке, напротив которой находится РОБОТ, с наименованием запрошенной детали или не совпадает;  
г) пуст грузовой отсек или не пуст.

Теперь мы можем приступить к составлению алгоритмов для нашего РОБОТА-МАНИПУЛЯТОРА. Научим его находить на стеллаже все детали заданного вида. Ясно, что робот должен просматривать все ячейки стеллажа в поисках нужных деталей. Как же проще всего организовать просмотр ячеек? Например, так: сначала РОБОТ перемещается к первой слева ячейке нижнего ряда стеллажа, а затем просматривает ряды стеллажа один за другим, пока не дойдет до верхнего ряда; каждый ряд рассматривает, двигаясь вдоль него слева направо, а после просмотра возвращается к крайней слева ячейке соответствующего ряда.

Давайте начнем с написания вспомогательного алгоритма "Просмотр ряда". Выполняя этот алгоритм, РОБОТ перемещается вдоль ряда слева направо, проверяя, совпадают ли наименования встречаемых по дороге деталей с наименованием запрошенной детали. В случае совпадения он перекладывает деталь из ячейки в грузовой отсек.

**Алгоритм "Просмотр ряда".**

Пока можно идти вправо, повторять:

Если наименование детали, лежащей в ячейке, совпадает с наименованием запрошенной детали, то:

Переложить деталь из ячейки в грузовой отсек.

Конец ветвления.

Шаг вправо.

Конец цикла.

Если наименование детали, лежащей в ячейке, совпадает с наименованием запрошенной детали, то:

Переложить деталь из ячейки в грузовой отсек.

Конец ветвления.

Как же теперь составить основной алгоритм? По-видимому, здесь необходимо применить циклический способ организации действий. РОБОТ должен переместиться к первой слева ячейке нижнего ряда стеллажа, просмотреть первый ряд, вернуться к его началу, подняться на одну ячейку вверх, просмотреть второй ряд, вернуться к его началу, подняться на одну ячейку вверх и т. д. Значит, в цикле должны повторяться действия:

Выполнить алгоритм "Просмотр ряда".  
Переместиться к первой слева ячейке ряда.  
Сделать шаг вверх.

РОБОТ должен повторять эти три действия до тех пор, пока не дойдет до верхнего ряда стеллажа. Вот как выглядит искомый алгоритм:

Запросить наименование детали.  
Переместиться к первой слева ячейке нижнего ряда.  
Пока можно идти вверх, повторять:  
Выполнить алгоритм "Просмотр ряда".  
Переместиться к первой слева ячейке ряда.  
Шаг вверх.  
Конец цикла.  
Выполнить алгоритм "Просмотр ряда".

Табличная форма организации данных полезна и при решении вычислительных задач, которые, как вы уже знаете, возникают повсюду. Поговорим, к примеру, о погоде. Допустим, не доверяя Гидрометцентру, вы решили сами заняться изучением погоды. Для этого вам, конечно, потребуется информация о ежемесячном количестве осадков, выпавших в вашем районе. За несколько лет наберется так много данных, что их надо как-то упорядочить. Лучше всего занести данные в таблицу. Каждая строка этой таблицы содержит 12 чисел — результаты измерений за один год. Число строк равно количеству лет, когда производились измерения. Располагая этой таблицей, можно решать разнообразные задачи: находить среднегодовое количество осадков, самый влажный месяц в году или за несколько лет и т. д. Можно попытаться даже прогнозировать погоду.

Прежде чем составлять алгоритмы решения этих задач, надо позаботиться, чтобы исполнитель мог работать с числовыми таблицами. Для этого мы не будем выбирать нового исполнителя, а снова расширим запас допустимых действий ВЫЧИСЛИТЕЛЯ. Прежде всего, ВЫЧИСЛИТЕЛЮ

*Не храните данные  
бессмысленной  
грудой —  
записывайте их  
в таблицы!*

нужно уметь запрашивать и сообщать таблицу. Поэтому будем считать допустимыми действия "Запросить таблицу" и "Сообщить таблицу". После слова "таблицу" надо указать обозначение таблицы, а также (например, в скобках) число строк и столбцов в ней:

**Запросить прямоугольную таблицу  $A$  из 21 строк и 12 столбцов.**

**Запросить линейную таблицу  $B$  из 52 элементов.**

**Сообщить линейную таблицу  $C(11)$ .**

**Сообщить таблицу  $D(17,5)$ .**

Будем считать также, что в команде присваивания, кроме обычных переменных, могут использоваться переменные, являющиеся элементами таблиц. При этом ВЫЧИСЛИТЕЛЬ использует обычные обозначения для этих элементов (см. предыдущий параграф). Номера строк и столбцов могут быть как числами, так и алгебраическими выражениями:  $A(1,2)$ ,  $B(I+J)$ ,  $C(I+1,2-J)$ . Вот примеры команд присваивания:

**Присвоить  $A(2,1)$  значение  $A(I,J)-B(K)^2$ .**

**Присвоить  $X$  значение  $C(2)-\sin(B(K)+X)$ .**

Для демонстрации работы ВЫЧИСЛИТЕЛЯ с таблицами посмотрим, какой вид примет следующая таблица  $A$

$$\begin{bmatrix} 6 & 4 & 3 \\ 3 & 2 & 1 \\ 1 & 7 & 6 \end{bmatrix}$$

после выполнения им, например, такой последовательности действий:

**Запросить таблицу  $A(3,3)$ .**

**Присвоить  $A(1,1)$  значение 2.**

**Присвоить  $A(2,1)$  значение  $A(2,3)$ .**

**Присвоить  $A(3,2)$  значение  $A(1,1)+A(1,2)$ .**

**Присвоить  $w$  значение  $A(1,1)+A(1,2)A(1,3)$ .**

**Сообщить таблицу  $A(3,3)$ .**

Чтобы проследить за выполнением этого алгоритма, надо прежде всего вспомнить, как выполняется действие присваивания: вычисляется значение выражения, стоящего в правой части, а затем этому значению дается имя переменной, стоящей в левой части. При этом старое значение переменной забывается. Например, в четвертом действии нашего алгоритма  $A(1,1)$  обозначает уже не число 6, как вначале, а число 2.

При выполнении этого алгоритма таблица  $A$  будет меняться следующим образом:

$$\begin{bmatrix} 6 & 4 & 3 \\ 3 & 2 & 1 \\ 1 & 7 & 6 \end{bmatrix} \rightarrow \begin{bmatrix} 2 & 4 & 3 \\ 3 & 2 & 1 \\ 1 & 7 & 6 \end{bmatrix} \rightarrow \begin{bmatrix} 2 & 4 & 3 \\ 1 & 2 & 1 \\ 1 & 7 & 6 \end{bmatrix} \rightarrow \begin{bmatrix} 2 & 4 & 3 \\ 1 & 2 & 1 \\ 1 & 6 & 6 \end{bmatrix}.$$

Переменная  $w$  примет значение 14.

Допустим теперь, что у нас имеется линейная таблица из двенадцати элементов, в которую занесены результаты ежемесячных измерений количества осадков, выпавших в 1990 г. Составим для ВЫЧИСЛИТЕЛЯ алгоритм, с помощью которого он определит число засушливых месяцев, когда количество осадков было ниже, скажем, 10 мм.

Обозначим таблицу наблюдений буквой  $A$ , а искомое число месяцев буквой  $n$ . ВЫЧИСЛИТЕЛЬ должен просматривать элементы таблицы  $A$  с первого по двенадцатый и каждый раз, встретив число, меньшее 10, добавлять к значению  $n$  единицу (сначала  $n$  равняется 0). Не правда ли, очень похоже на поиск нужных деталей на стеллаже? Роль грузового отсека играет переменная  $n$ . Если бы ВЫЧИСЛИТЕЛЬ мог определять, рассматривает он крайнюю клетку таблицы или нет, мы могли бы почти дословно переписать алгоритм "Просмотр ряда". Как же ВЫЧИСЛИТЕЛЬ "узнает", что таблица закончилась? Для этого он должен "помнить" номер рассматриваемого элемента таблицы и сравнивать его с числом 12. Обозначим этот номер буквой  $i$ .

Запишем алгоритм решения нашей "метеорологической" задачи:

Присвоить переменной  $n$  (число засушливых месяцев) начальное значение 0.

Присвоить  $i$  (номеру месяца) значение 1.

Пока  $i \leq 12$ , повторять:

Если  $A(i) < 10$ , то:

Присвоить  $n$  значение  $n+1$ .

Конец ветвления.

Присвоить  $i$  значение  $i+1$ .

Конец цикла.

Сообщить  $n$ .

Как видите, переменная  $i$  выступает в роли счетчика, подсчитывающего, сколько раз выполняется набор действий, входящих в тело цикла. Подобные циклы со счетчиком часто применяются при работе с таблицами. Будем использовать для них специальную форму записи:

Для каждого  $i$  от  $L$  до  $R$ :

$P_1$

$P_2$

$\vdots$

$P_n$

Конец цикла по  $i$ .

Здесь  $i$  — счетчик;  $L$  — нижняя, а  $R$  — верхняя граница изменения  $i$ ;  $P_1, P_2, \dots, P_n$  — действия, которые выполняются в цикле. Разумеется, можно использовать другие

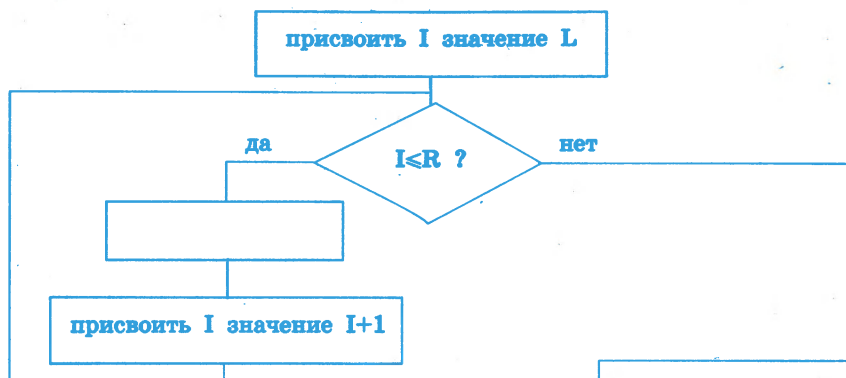


Рис. 50

обозначения для счетчика и границ цикла. Кроме того, вместо  $L$  и  $R$  можно писать алгебраические выражения или просто числа.

Выполняя этот цикл (назовем его циклом "Для каждого"), ВЫЧИСЛИТЕЛЬ сначала присвоит  $i$  значение  $L$ . Затем он проверит, больше ли  $i$ , чем  $R$ ; если да, то цикл заканчивается (не начавшись), если нет, то ВЫЧИСЛИТЕЛЬ выполнит действия  $P_1$ ,  $P_2$  и т. д., число  $i$  увеличится на 1, и для этого нового значения  $i$  повторяется тело цикла до тех пор, пока  $i$  не превзойдет  $R$ .

Приведем блок-схему цикла "Для каждого" (рис. 50):

Запишем алгоритм нахождения числа засушливых месяцев в году, используя цикл в форме "Для каждого":

Присвоить переменной  $n$  (число засушливых месяцев) начальное значение 0.

Для каждого  $i$  от 1 до 12:

Если  $A(i) < 10$ , то:

Присвоить  $n$  значение  $n+1$ .

Конец ветвления.

Конец цикла по  $i$ .

Сообщить  $n$ .

Допустим теперь, что известна таблица наблюдений за несколько лет и нужно подсчитать количество  $n$  засушливых месяцев за этот период. В таблице наблюдений (обозначим ее через  $B$ ) число строк равно числу лет, а в каждой строке по 12 элементов. Естественно оформить в качестве вспомогательного алгоритма подсчет числа засушливых месяцев за год. Аргументами в нем будут номер года (строки в таблице наблюдений)  $p$  и число  $x$  засушливых месяцев за предыдущие годы рассматриваемого периода. Результатом является новое значение переменной  $n$ .

Алгоритм "Один год".

Аргументы: номер года  $p$  и число  $x$  засушливых месяцев за предыдущие годы.

Результат:  $n$  (число засушливых месяцев за  $p$  лет).

Присвоить  $n$  начальное значение, равное  $x$ .

Для каждого  $q$  от 1 до 12:

Если  $B(p, q) < 10$ , то:

Присвоить  $n$  значение  $n+1$ .

Конец ветвления.

Конец цикла по  $q$ .

Основной алгоритм теперь запишется так:

Сообщить "Введите число лет".

Запросить  $k$ .

Сообщить "Введите таблицу наблюдений".

Запросить таблицу  $B(k, 12)$ .

Присвоить переменной  $n$  (число засушливых месяцев) начальное значение 0.

Для каждого  $i$  от 1 до  $m$ :

Выполнить алгоритм "Один год" при  $x=n$ ,  $p=i$ .

Конец цикла по  $i$ .

Сообщить "Число засушливых месяцев".

Сообщить  $n$ .

## ? Вопросы

1. Какими допустимыми действиями обладает РОБОТ-МАНИПУЛЯТОР?

2. Какие допустимые действия позволяют ВЫЧИСЛИТЕЛЮ оперировать с таблицами?

3. Как оформляется цикл "Для каждого"? Как он изображается при помощи блок-схемы?

## Задания для самостоятельного выполнения

1. Составьте для РОБОТА-МАНИПУЛЯТОРА алгоритмы, выполнив которые он:

- а) отойдет от края стеллажа;
- б) положит в грузовой отсек деталь из крайней правой ячейки верхнего ряда стеллажа;
- в) соберет все диоды из первого справа столбца стеллажа;
- г) соберет все резисторы из третьего сверху ряда стеллажа;
- д) оставит в нижнем ряду стеллажа только детали выбранного типа;
- е) оставит на стеллаже только детали выбранного типа;
- ж) расположит детали из нижнего ряда стеллажа подряд (без пропусков), начиная с левой ячейки ряда;

з) расположит все детали, находящиеся на стеллаже, подряд, начиная с левой нижней ячейки стеллажа;

и)\* положит все резисторы, имеющиеся на стеллаже, подряд, начиная с левой нижней ячейки стеллажа;

к)\* определит, каких деталей на стеллаже меньше — диодов или резисторов, положив дефицитную деталь в левую нижнюю ячейку стеллажа (или оставив эту ячейку пустой, если дефицитная деталь на стеллаже отсутствует);

л)\* рассортирует детали на стеллаже, положив сначала подряд все диоды, затем все резисторы и т. д.

2. Чему будет равно значение переменной  $M$  после выполнения следующего алгоритма? Сколько раз будет исполняться тело цикла?

Присвоить  $M$  значение 0.

Для каждого  $I$  от 1 до 8:

Присвоить  $M$  значение  $M+I$ .

Присвоить  $I$  значение  $I+1$ .

Конец цикла по  $I$ .

3. Дана таблица  $A$ :

$$\begin{bmatrix} 3 & -2 & 5 & 7 \\ 6 & 0 & -2 & 0 \\ 8 & 1 & 2 & 0 \end{bmatrix}$$

Как изменится таблица  $A$  после выполнения следующих алгоритмов:

а) Присвоить  $A(1,1)$  значение 2.

Присвоить  $A(3,2)$  значение  $A(1,1)+A(3,2)$ .

Если  $A(3,2)>3$ , то:

Присвоить  $A(3,4)$  значение  $2A(3,1)$ .

Иначе:

Присвоить  $A(1,4)$  значение  $A(3,2)+3$ .

Конец ветвления.

Для каждого  $I$  от 1 до 3:

Присвоить  $A(I,I)$  значение  $I^2$ .

Конец цикла по  $I$ .

б) Присвоить  $A(2,2)$  значение 2.

Присвоить  $A(1,2)$  значение  $A(2,2)-A(1,2)$ .

Если  $A(1,2)>3$ , то:

Присвоить  $A(3,3)$  значение  $A(3,1)/A(1,1)$ .

Иначе:

Присвоить  $A(1,4)$  значение  $3A(3,2)$ .

Конец ветвления.

Для каждого  $I$  от 1 до 3:

Присвоить  $A(I,4-I)$  значение  $A(4-I,I)$ .

Конец цикла по  $I$ .

4°. В алгоритме вычисления суммы кубов элементов таблицы  $A$ , имеющей 4 строки и 3 столбца, злоумышленник поменял местами две строки. Вот что у него получилось:

Присвоить  $S$  значение 0.

Для каждого  $I$  от 1 до 4:

Для каждого  $J$  от 1 до 3:

Присвоить  $S$  значение  $S+A(I,J)^3$ .

Конец цикла по  $I$ .

Конец цикла по  $J$ .

Какие строки поменял местами злоумышленник?

5. Для нахождения максимального числа в прямоугольной таблице злоумышленником был составлен следующий алгоритм:

Запросить таблицу  $A(5,5)$ .

Присвоить  $M$  значение 0.

Для каждого  $I$  от 1 до 5:

Для каждого  $J$  от 1 до 5:

Если  $M < A(I,J)$ , то:

Присвоить  $M$  значение  $A(I,J)$ .

Конец ветвления.

Конец цикла по  $J$ .

Конец цикла по  $I$ .

Сообщить "Максимум равен".

Сообщить  $M$ .

Как и было задумано злоумышленником, ЭВМ, действуя по этой программе, далеко не всегда достигает цели. Требуется:

а) привести примеры таблиц, в которых ЭВМ найдет максимальное число;

б) привести примеры таблиц, в которых ЭВМ не найдет максимального числа;

в) исправить программу.

6. Работая с редактором текстов, школьник напечатал начало известного стихотворения Крученых:

дыл

бул

щир

Подошедший злоумышленник поставил курсор на начало слова "дыл" и затем выполнил следующий алгоритм:

Для каждого  $I$  от 1 до 3:

Для каждого  $J$  от 1 до 3:

Если буква, стоящая в  $I$ -м слове на  $J$ -м месте равна "л", то:

Заменить эту букву на "м".

Конец ветвления.

Конец цикла по  $J$ .

Конец цикла по  $I$ .

Какие слова оказались напечатанными на экране после выполнения этого алгоритма?

7°. Составьте для ВЫЧИСЛИТЕЛЯ алгоритм проверки того, что в данной квадратной таблице равны элементы, расположенные симметрично относительно диагонали, проведенной из левого верхнего угла в правый нижний угол:

$$\begin{bmatrix} * & & & \\ & * & & \\ & & * & \\ & & & * \end{bmatrix}$$

8°. Составьте для ВЫЧИСЛИТЕЛЯ алгоритмы, выполнив которые он заполнит и сообщит таблицы из задач 2,а—в (§ 24).

9°. Дана прямоугольная таблица. Составьте алгоритмы, по которым ВЫЧИСЛИТЕЛЬ:

- а) заменит в таблице все отрицательные элементы нулями, а положительные оставит неизменными;
- б) поменяет в таблице первые две строки местами;
- в) поменяет в таблице первые два столбца местами.

10. Дана квадратная таблица  $A$ , состоящая из  $n^2$  элементов. Напишите алгоритм для вычисления:

- а) суммы квадратов элементов, стоящих на диагонали, идущей из левого верхнего угла в правый нижний;
- б) суммы квадратных корней тех же элементов.

11. В прямоугольной таблице с двумя строками, по 10 чисел в каждой, записаны координаты 10 точек (в первой строке абсциссы, во второй — ординаты). Требуется:

- а) найти максимальное расстояние между точками;
- б) найти минимальное расстояние между точками;
- в) составить таблицу всех попарных расстояний между точками.

12. В прямоугольной таблице записаны координаты 10 векторов. Определить:

- а) какие из этих векторов перпендикулярны данному вектору;
- б) какие из этих векторов параллельны данному вектору;
- в) какие из этих векторов имеют ту же длину, что и данный вектор.

13°. Пусть дана таблица наблюдений  $B$  (см. текст § 25). Составьте алгоритм для определения:

- а) самого засушливого месяца за один год;
- б) самого влажного месяца за один год;
- в) общего количества осадков, выпавших за один год;
- г) среднего количества осадков за месяц в течение одного года;

д) числа месяцев в течение одного года, когда количество осадков превышало 40 мм.

14°. Используя алгоритмы решения задачи 13 в качестве вспомогательных, составьте алгоритмы определения:

а) самого засушливого и самого влажного месяца за все время наблюдений;

б) общего количества осадков, выпавших за время наблюдений;

в) наиболее засушливого (влажного) года.

15. Дана таблица результатов шахматного турнира. Составьте для ВЫЧИСЛИТЕЛЯ алгоритм определения победителя этого турнира.

16. Составьте математическую модель и алгоритм решения следующей задачи:

**Задача.** На заводе имеется несколько цехов. В памяти ВЫЧИСЛИТЕЛЯ постоянно имеются сведения о наличии сырья в каждом из них. Требуется определить номер цеха, в котором осталось меньше всего сырья.



### Лабораторная работа 15

#### РЕШЕНИЕ ЗАДАЧ С ИСПОЛЬЗОВАНИЕМ ТАБЛИЧНОЙ ФОРМЫ ОРГАНИЗАЦИИ ДАННЫХ

Приступим к работе с исполнителями РОБОТОМ-МАНИПУЛЯТОРОМ и ВЫЧИСЛИТЕЛЕМ.

Перед вами на экране изображен прямоугольный стеллаж. В ячейках стеллажа расположены детали, обозначенные буквами (например, так, как на рисунке 51). Сам РОБОТ изображен курсором.

А	В										А				
			Р												
					С						Г				
			К								К				
											Д				
■		А				С									

Вижу деталь →
Запрошенная деталь →

**Грузовой отсек**

--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

Рис. 51

Вот список команд, которые соответствуют допустимым действиям РОБОТА-МАНИПУЛЯТОРА:

1. ЗАПРОСИТЬ ДЕТАЛЬ
2. ШАГ ВПРАВО
3. ШАГ ВЛЕВО
4. ШАГ ВВЕРХ
5. ШАГ ВНИЗ
6. К НАЧАЛУ СТЕЛЛАЖА
7. К НАЧАЛУ РЯДА
8. ВЗЯТЬ ДЕТАЛЬ
9. ПОЛОЖИТЬ ДЕТАЛЬ

Кроме того, РОБОТ умеет проверять следующие условия:

1. ВЫБРАННАЯ ДЕТАЛЬ
2. ЕСТЬ ДЕТАЛЬ
3. МОЖНО ВПРАВО
4. МОЖНО ВЛЕВО
5. МОЖНО ВВЕРХ
6. МОЖНО ВНИЗ
7. ГРУЗОВОЙ ОТСЕК ПУСТ
8. НЕ ВЫБРАННАЯ ДЕТАЛЬ
9. НЕТ ДЕТАЛИ
10. НЕЛЬЗЯ ВПРАВО
11. НЕЛЬЗЯ ВЛЕВО
12. НЕЛЬЗЯ ВВЕРХ
13. НЕЛЬЗЯ ВНИЗ
14. ГРУЗОВОЙ ОТСЕК НЕ ПУСТ

Вот как записывается программа сбора всех нужных деталей в ряду (см. § 25):

1. ЗАПРОСИТЬ ДЕТАЛЬ
2. ПОКА МОЖНО ВПРАВО, ПОВТОРЯТЬ:
3. ЕСЛИ ВЫБРАННАЯ ДЕТАЛЬ, ТО:
4. ВЗЯТЬ ДЕТАЛЬ
5. КОНЕЦ ВЕТВЛЕНИЯ
6. ШАГ ВПРАВО
7. КОНЕЦ ЦИКЛА
8. ЕСЛИ ВЫБРАННАЯ ДЕТАЛЬ, ТО:
9. ВЗЯТЬ ДЕТАЛЬ
10. КОНЕЦ ВЕТВЛЕНИЯ

Как и для ЧЕРТЕЖНИКА, программу надо начинать командой "НАЧАТЬ РАБОТУ". По этой команде РОБОТ предоставит вам возможность заполнить ячейки стеллажа деталями по вашему усмотрению. Поставив деталь в ячейку (нажав на клавишу с соответствующей цифрой), переместите РОБОТА к другой ячейке с помощью клавиш со стрелками. Закончив размещение деталей на стеллаже, нажмите на клавишу "ПЕРЕВОД СТРОКИ". После этого

РОБОТ сможет перемещаться по стеллажу, только выполняя ваши команды.

Посмотрите, как РОБОТ выполняет команды, перечисленные выше. Для этого, как обычно, команды надо вводить без номера.

Теперь давайте заставим РОБОТА выполнить алгоритм поиска на стеллаже нужной детали. Сначала, конечно, надо превратить этот алгоритм в программу. Наберите полученную программу и запустите ее. Когда, выполняя команду "ЗАПРОСИТЬ ДЕТАЛЬ", РОБОТ спросит вас, какие детали вам нужны, введите соответствующую цифру. Проследите, в каком порядке РОБОТ обходит ячейки стеллажа, как исчезают со стеллажа нужные вам детали, перемещаясь в грузовой отсек. Если вы разобрались, как организируются данные для РОБОТА, то переходите к работе с ВЫЧИСЛИТЕЛЕМ (в противном случае еще раз проследите, как РОБОТ исполняет вашу программу).

Вы и дальше можете представлять себе таблицу в виде прямоугольного стеллажа. Есть, однако, существенная разница между тем, как работают с таблицами РОБОТ и ВЫЧИСЛИТЕЛЬ. РОБОТУ, чтобы извлечь деталь из нужной ячейки стеллажа, надо, вообще говоря, пройти длинный путь, тогда как ВЫЧИСЛИТЕЛЮ достаточно знать номера строки и столбца, и он сразу получит нужную информацию.

Приступим к обработке метеорологических наблюдений. Давайте поручим ВЫЧИСЛИТЕЛЮ исполнить составленный вами алгоритм поиска самого влажного месяца (задача 9, а из § 25). Конечно, сначала надо запастись таблицей наблюдений. Сотрудники Екатеринбургской метеостанции сообщили нам таблицу наблюдений за 20 лет. Вот две строчки из этой таблицы:

Год	1	2	3	4	5	6	7	8	9	10	11	12
1983	34,5	51,3	20,5	26,9	45,5	71,5	152,9	96,6	74,8	14,5	21,0	22,3
1984	8,0	1,2	3,8	11,9	66,3	60,0	50,6	145,2	79,9	74,9	56,6	9,4

Для отладки программы вам этой таблицы будет достаточно.

Чтобы ВЫЧИСЛИТЕЛЬ был способен работать с таблицей, ему нужно в самом начале сообщить размеры таблицы. Это делают командой

**ТАБЛИЦА ...**,

где вместо многоточия пишут имя таблицы и ее размеры в круглых скобках. Например, для данной таблицы команда выглядит так:

**ТАБЛИЦА В(2,12)**

По этой команде **ВЫЧИСЛИТЕЛЬ** резервирует в памяти место, где будет размещаться таблица *В*. Составьте теперь программу и запустите ее. Ну и как?

Вы уже знаете, что не надо огорчаться, если программа с первого раза не прошла, а надо заняться ее отладкой. Можно воспользоваться режимом "отладка". Если же это не помогло вам найти ошибку, попробуйте обнаружить ее другим способом. Таких способов очень много. Например, если в результате работы программы некоторая переменная принимает не то значение, которое должно быть, то полезно после каждой команды, меняющей значение этой переменной, заставить **ВЫЧИСЛИТЕЛЯ** сообщать новое значение.

После окончания отладки примените вашу программу для отыскания наиболее влажного месяца за 20 лет наблюдений. Данные этих наблюдений хранятся на диске. Чтобы вызвать их в оперативную память ЭВМ, надо использовать команду

**ВЫЗВАТЬ ТАБЛИЦУ *В*(20,12).**

Теперь начало вашей программы должно выглядеть так:

**1 ТАБЛИЦА *В*(20,12)**

**2 ВЫЗВАТЬ ТАБЛИЦУ *В*(20,12)**

## § 26. ЗАДАЧИ ПЛАНИРОВАНИЯ

Представьте, что вы стали директором завода и, изучив спрос, решили организовать участок для производства двух видов товаров повышенного спроса — мясорубки и скороварки. Для краткости обозначим эти товары буквами *А* и *Б*. Допустим, что вам удалось заключить договоры с другими предприятиями на поставку ресурсов (металла, элек-



троэнергии и т. п.) и выделить определенное число рабочих. Изучение рыночной конъюнктуры позволило определить минимальные объемы производства для каждого изделия. Всякий хороший директор стремится к тому, чтобы прибыль была наибольшей. Будем считать это и вашей задачей.

**Задача.** На участке работает 20 человек; каждый из них в среднем за год работает 1800 ч. Выделенные ресурсы: 32 т металла, 54 тыс. кВт·ч электроэнергии. План по реализации: не менее 2 тыс. изделий А и не менее 3 тыс. изделий Б. На выпуск одной тысячи изделий А затрачивается 3 т металла, 3 тыс. кВт·ч электроэнергии и 3 тыс. ч рабочего времени. На выпуск одной тысячи изделий Б затрачивается 1 т металла, 6 тыс. кВт·ч электроэнергии и 3 тыс. ч рабочего времени. От реализации 1 тыс. изделий А завод получает прибыль 500 тыс. р., от реализации 1 тыс. изделий Б — 700 тыс. р. Выпуск какого количества изделий А и Б (в тыс. штук) надо запланировать, чтобы прибыль от их реализации была наибольшей?

Как вы помните, начать нужно с компьютерной модели. Сделаем упрощающие предположения. Часть из них мы высказали уже при формулировке задачи: мы опустили целый ряд ограничений на ресурсы, необходимые для производства (кроме людских ресурсов, расхода металла и электроэнергии), а также взяли усредненное значение затрат рабочего времени.

Далее, пусть  $x$  — планируемое количество изделий А, а  $y$  — планируемое количество изделий Б (в тыс. штук). Для простоты будем считать числа  $x$  и  $y$  натуральными.

Все исходные данные указаны в условии задачи. Результатом, очевидно, являются максимальная возможная прибыль от реализации и соответствующие ей оптимальные значения  $x$  и  $y$ .

Запишем теперь соотношения, связывающие исходные данные и результаты.

План по реализации запишется следующими неравенствами:

$$x \geq 2; \quad (1)$$

$$y \geq 3. \quad (2)$$

Поскольку на изготовление одной тысячи изделий А расходуется 3 т металла, а на изготовление одной тысячи изделий Б расходуется 1 т металла, общий расход металла составит  $3x+y$  т. По условию он не должен превышать 32 т, т. е. должно быть справедливо неравенство

$$x+y \leq 32. \quad (3)$$

Аналогично записываются остальные ограничения. Ограничение на расход электроэнергии:

$$3x+6y \leq 54. \quad (4)$$

Ограничение на ресурсы рабочего времени:

$$3x+3y \leq 36. \quad (5)$$

Прибыль от реализации  $x$  тысяч изделий А и  $y$  тысяч изделий Б равна

$$500x+700y.$$

Теперь наша задача может быть сформулирована так: найти наибольшее значение выражения  $500x+700y$  ( $x, y$  — натуральные числа), где  $x$  и  $y$  должны удовлетворять неравенствам (1) — (5).

Приступим к составлению алгоритма.

Требуемые значения  $x$  и  $y$  мы будем искать, перебирая все значения, которые они могут принимать. Поскольку перебрать можно лишь конечное число значений, давайте заранее определим границы для  $x$  и  $y$ .

Из неравенств (3)—(5) получаем (вспомните правила действий с неравенствами):

$$\begin{aligned} y &\leq 32 - 3x; \\ y &\leq (54-3x)/6; \\ y &\leq (36-3x)/3. \end{aligned}$$

Далее, поскольку  $x \geq 2$ , получаем:

$$\begin{aligned} 32-3x &\leq 26; \\ (54-3x)/6 &\leq 8; \\ (36-3x)/3 &\leq 10. \end{aligned}$$

Отсюда  $y \leq 8$ . Рассуждая аналогично, из условия  $y \geq 3$  получаем  $x \leq 29/3$ , откуда  $x \leq 9$  (мы рассматриваем только целые значения  $x$  и  $y$ !).

Таким образом, для любых значений  $x$  от 1 до 9 и  $y$  от 1 до 8 можно подсчитать значение прибыли по формуле  $500x+700y$ , если, конечно,  $x$  и  $y$  удовлетворяют неравенствам (1)—(5). Если же  $x$  и  $y$  не удовлетворяют неравенствам (1)—(5), то значение прибыли будем считать равным нулю, поскольку в этом случае участок просто не в состоянии выпустить  $x$  тысяч изделий А и  $y$  тысяч изделий Б.

Полученные 72 значения прибыли удобно расположить в форме таблицы, состоящей из 9 строк и 8 столбцов. На пересечении строки с номером  $x$  и столбца с номером  $y$  записывается прибыль, полученная от реализации  $x$  тысяч изделий А и  $y$  тысяч изделий Б. Обозначим эту таблицу буквой V.

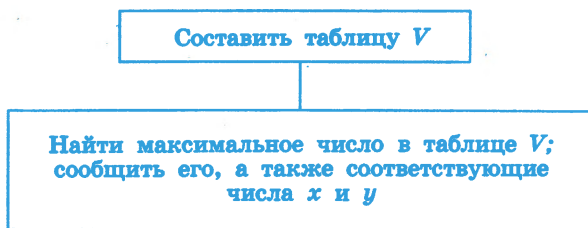


Рис. 52

Итак, элементы таблицы  $V$  будем определять по формуле

$$V(x,y) = \begin{cases} 500x+700y, & \text{если } x \text{ и } y \text{ удовлетворяют} \\ & \text{неравенствам (1)—(5);} \\ 0, & \text{в противном случае.} \end{cases} \quad (6)$$

Таким образом, задача свелась к нахождению максимального элемента в таблице  $V$ . Подобную задачу мы разбирали на лабораторной работе 14. Единственное существенное отличие состоит в том, что таблица результатов наблюдений нам была дана заранее, а элементы таблицы  $V$  нужно предварительно сосчитать по приведенному выше правилу (6).

Проведем пошаговую детализацию алгоритма. Он будет состоять из двух крупных блоков (рис. 52):

В лабораторной работе 14 вы, по сути дела, составили программу нахождения максимального элемента в таблице. Вы можете пользоваться своим алгоритмом или, если хотите, приведенным ниже.

Сначала напишем вспомогательный алгоритм поиска максимума в строке (его аргументом является номер строки  $I$ , а результатом — максимальное число  $R$  в строке таблицы и его номер  $Q$  в строке):

**Алгоритм "Максимальный элемент в строке".**

**Аргумент:**  $I$  (номер строки).

**Результаты:** максимальное число  $R$  и его номер  $Q$  в строке.

**Присвоить  $R$  начальное значение 0.**

**Для каждого  $J$  от 1 до 8:**

**Если  $V(I,J) > R$ , то:**

**Присвоить  $R$  очередное значение  $V(I,J)$ .**

**Присвоить  $Q$  очередное значение  $J$ .**

**Конец ветвления.**

**Конец цикла по  $J$ .**

Алгоритм поиска максимума в таблице выглядит так:

**Присвоить  $M$  начальное значение 0.**

Для каждого  $P$  от 1 до 9:

Выполнить алгоритм "Максимальный элемент в строке", взяв  $I$  равным  $P$ .

Если  $R > M$ , то:

Присвоить  $M$  очередное значение, равное  $R$ .

Присвоить  $X$  очередное значение, равное  $P$ .

Присвоить  $Y$  очередное значение, равное  $Q$ .

Конец ветвления.

Конец цикла по  $P$ .

Сообщить "Максимум прибыли, количество изделий  $A$ , количество изделий  $B$  (тыс. штук)".

Сообщить  $M$ ,  $X$ ,  $Y$ .

Теперь составим алгоритм, реализующий первый блок нашей блок-схемы. Ясно, что этот алгоритм состоит в вычислении для всех  $x$  от 1 до 9 и всех  $y$  от 1 до 8 значения  $V(x, y)$ . Вычисление  $V(x, y)$  можно изобразить блок-схемой на рисунке 53.

В дальнейшей детализации нуждается лишь проверка условия. Соответствующий вспомогательный алгоритм назовем "Условие", его аргументы —  $x$  и  $y$ , а результат — сигнальная переменная  $C$ : она равна 0, если условие выполняется, и равна 1 в противном случае. Этот алгоритм выполняется так: сначала  $C$  присваивается значение 0; затем если хотя бы одно из пяти неравенств (1)—(5) нарушается, то  $C$  становится равным 1, в противном случае остается равным 0.

Алгоритм "Условие".

Аргументы:  $X, Y$  (количества изделий  $A$  и  $B$ ).

Результат:  $C$  (сигнальная переменная, равная 0, если  $X$  и  $Y$  удовлетворяют неравенствам (1)—(5), и равная 1 в противном случае).

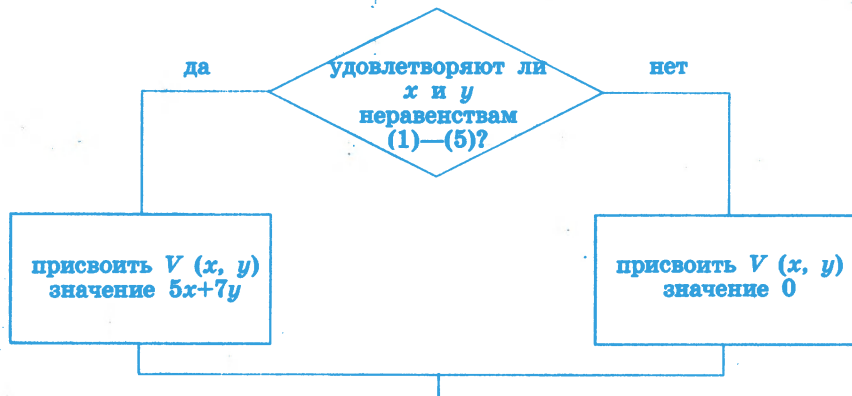


Рис. 53

Присвоить  $C$  начальное значение 0.

Если  $X < 2$ , то:

Присвоить  $C$  значение 1.

Конец ветвления.

Если  $Y < 3$ , то:

Присвоить  $C$  значение 1.

Конец ветвления.

Если  $3X + Y > 32$ , то:

Присвоить  $C$  значение 1.

Конец ветвления.

Если  $3X + 6Y > 54$ , то:

Присвоить  $C$  значение 1.

Конец ветвления.

Если  $3X + 3Y > 36$ , то:

Присвоить  $C$  значение 1.

Конец ветвления.

Как видите, алгоритм получился довольно длинный и однообразный. У каждого уважающего себя программиста должно возникнуть желание сделать его покороче. Да и нет никакой гарантии, что плановый отдел не предъявит завтра еще какие-нибудь новые условия. Что же тогда, весь алгоритм переписывать? Разумеется, надо всегда стремиться к тому, чтобы текст вспомогательного алгоритма не зависел от данных, которые требуются в основном алгоритме. Решить обе поставленные задачи нам снова поможет табличная организация данных. Только сначала превратим неравенства (1)–(5) в неравенства одного смысла. Для этого достаточно заменить неравенства  $x \geq 2$  и  $y \geq 3$  на неравенства  $-x \leq -2$  и  $-y \leq -3$ . Составим из коэффициентов при  $x$  и  $y$  и правых частей неравенств таблицу  $T$ :

$$\begin{bmatrix} -1 & 0 & -2 \\ 0 & -1 & -3 \\ 3 & 1 & 32 \\ 3 & 6 & 54 \\ 3 & 3 & 36 \end{bmatrix}$$

Теперь вспомогательный алгоритм "Условие" можно записать так:

Алгоритм "Условие".

Аргументы:  $X, Y$ .

Результат:  $C$ .

Присвоить  $C$  значение 0.

Для каждого  $I$  от 1 до 5:

Если  $T(I, 1) \cdot X + T(I, 2) \cdot Y > T(I, 3)$ , то:

Присвоить  $C$  значение 1.

Конец ветвления.

Конец цикла по  $I$ .

Как видите, алгоритм получился гораздо короче и универсальнее: при изменении исходных данных не надо переписывать весь алгоритм, а достаточно поменять таблицу  $T$  и, если число условий изменится, поменять число 5 в цикле "Для каждого".

Теперь мы можем записать основной алгоритм решения задачи:

Для каждого  $I$  от 1 до 9:  
Для каждого  $J$  от 1 до 8:  
Выполнить алгоритм "Условие" при  $X=I$ ,  $Y=J$ .  
Если  $C=0$ , то:  
Присвоить  $V(I, J)$  значение прибыли  $500I+700J$ .  
Иначе:  
Присвоить  $V(I, J)$  значение 0.  
Конец ветвления.  
Конец цикла по  $J$ .  
Конец цикла по  $I$ .  
Присвоить  $M$  начальное значение 0.  
Для каждого  $P$  от 1 до 9:  
Выполнить алгоритм "Максимальный элемент в строке", взяв  $I$  равным  $P$ .  
Если  $R>M$ , то:  
Присвоить  $M$  очередное значение, равное  $R$ .  
Присвоить  $X$  очередное значение, равное  $P$ .  
Присвоить  $Y$  очередное значение, равное  $Q$ .  
Конец ветвления.  
Конец цикла по  $P$ .  
Сообщить "Максимум прибыли, количество изделий А, количество изделий Б (тыс. штук)".  
Сообщить  $M$ ,  $X$ ,  $Y$ .

Соответствующая программа на языке Бейсик:

```
1 DIM V(9,8)
2 FOR I=1 TO 9
3   FOR J=1 TO 8
4     LET X=I: LET Y=J: GOSUB 22
5     IF C=0 THEN LET V(I,J)=500*I+700*J ELSE LET
      V(I,J)=0
6   NEXT J
7 NEXT I
8 LET M=0
9 FOR P=1 TO 9
10  LET I=P: GOSUB 16
11  IF R>M THEN LET M=R: LET X=P: LET Y=Q
12 NEXT P
13 PRINT "Максимум прибыли, планируемый вы-
      пуск изделий А и Б (тыс. штук)"
```

```

14 PRINT M,X,Y
15 STOP

16 REM Максимальный элемент в строке
17 LET R=0
18 FOR J=1 TO 8
19 IF V(I,J)>R THEN LET R=V(I,J): LET Q=J
20 NEXT J
21 RETURN

22 REM Условие
23 LET C=0
24 IF X<2 THEN LET C=1
25 IF Y<3 THEN LET C=1
26 IF 3*X+Y>32 THEN LET C=1
27 IF 3*X+6*Y>54 THEN LET C=1
28 IF 3*X+3*Y>36 THEN LET C=1
29 RETURN

```



### Задания для самостоятельного выполнения

1°. Составьте алгоритм нахождения максимального значения выражения  $5x^2 - 6y^2$ , если натуральные числа  $x$ ,  $y$  удовлетворяют неравенствам:  $x-2y<4$ ,  $2xy>7$ ,  $x+y<100$ .

2°. Кооператив из 20 человек выпускает изделия А и Б, те же, что и участок завода, о котором рассказано в этом параграфе. Кооператив намерен получить прибыль не менее 6,5 млн. р. в год. Ему выделены 54 тыс. квт.ч электроэнергии. Какое минимальное количество металла потребуется кооперативу, чтобы обеспечить нужную прибыль? Составьте компьютерную модель и алгоритм решения этой задачи.

3°. Для полива трех полей колхоз использует насосную станцию. На первое поле требуется подать не менее 200 кубометров воды в сутки, на второе — не менее 300, на третье не менее 350. В распоряжении колхоза 1200 кубометров воды в сутки. Стоимость подачи  $q$  кубометров воды на первое поле  $1570q^{1/8}$  р., на второе поле  $1720q^{1/8}$  р., на третье  $1930q^{1/8}$  р. Сколько кубометров воды надо подать на каждое поле, чтобы затраты были наименьшими? Составьте компьютерную модель и алгоритм решения этой задачи.



### Лабораторная работа 16

#### РЕШЕНИЕ ЗАДАЧ ПЛАНИРОВАНИЯ

Итак, вы директор завода. Вас, конечно, интересует ответ к задаче, разобранный в § 26. Запустите программу и запишите результаты. Они вам еще понадобятся.

Давайте проанализируем, что же получилось. Сначала выясним, все ли ресурсы участка завода израсходует, выполняя этот план. Подсчитайте расход металла, электроэнергии и рабочего времени. Вы видите: ресурсы электроэнергии и рабочего времени будут использованы полностью, в то время как запас металла — лишь частично.

Перед вами как директором завода стоит задача увеличения прибыли. Какие пути для этого вы изберете? Прежде всего надо понять, какие факторы влияют на прибыль. Это, например, металлоемкость, энергоемкость изделий и производительность труда.

Можно, конечно, искусственно повысить металлоемкость изделий А и Б, одновременно увеличив их цену. По этому пути в недавнем прошлом шли многие руководители предприятий. В результате из продажи исчезали небольшие кастрюли, а появлялись гиганты, больше пригодные для плавания в них, чем для приготовления пищи. Подобные методы наносили огромный ущерб нашей экономике. Мы уверены, что вы изберете другой путь.

Наиболее приемлемый путь — повышение производительности труда. Как вы знаете, производительность труда повышается за счет сокращения рабочего времени, необходимого для производства продукции. Посмотрим, во сколько раз надо уменьшить это время, чтобы использовать все ресурсы (материальные и трудовые) и получить максимальную прибыль. Таким образом, мы теперь должны определить, сколько изделий А и Б надо выпустить, чтобы получить максимальную прибыль, израсходовав всю имеющуюся электроэнергию и весь запас металла, не обращая внимания на ресурс рабочего времени. Затем посмотрим, сколько рабочего времени понадобилось бы на производство этой продукции раньше, при прежней производительности труда. Отношение этого числа к имеющемуся ресурсу рабочего времени (36 000 ч) даст нам ответ.

Казалось бы, задача изменилась сильно, однако программа для ее решения легко получается из программы, созданной для решения предыдущей задачи. Прежде всего заметим, что изменилась область допустимых значений для  $x$  и  $y$ . В самом деле, число ограничений уменьшилось на единицу: нет ограничения (5). Найдите новые границы для  $x$  и  $y$  и внесите соответствующие изменения в программу. Далее, надо изменить подпрограмму, убрав из нее проверку неравенства (5). Наконец, в основной программе надо добавить команду для вычисления ответа к задаче. Измените программу (руководствуясь нашими указаниями) и получите результат. Какую прибыль теперь получает участок вашего завода?

## § 27. ПАКЕТЫ ПРИКЛАДНЫХ ПРОГРАММ

В начале изучения курса информатики вы поработали с электронными таблицами, информационно-поисковыми системами, текстовым и графическим редакторами. Понравилось? Удобно и быстро, не правда ли? Причем этими программами можно пользоваться, не зная их строения. Достаточно только разобраться в соответствующей инструкции. Важно и то, что с помощью каждой такой программы можно решить не только одну конкретную задачу, но и целый класс задач, иногда очень непохожих друг на друга.

Специальные программы, предназначенные для решения классов задач из той или иной конкретной области, называются **прикладными программами**. Несколько взаимосвязанных прикладных программ называют **пакетом прикладных программ (ППП)**.

Пакеты прикладных программ освобождают специалиста, желающего воспользоваться помощью компьютера, от длительной и нелегкой работы по изучению языков программирования и от самостоятельного составления программ. Легко представить беспомощность пассажиров, которым предложили перед полетом изготовить самолет. В похожем положении оказалась бы огромная армия специалистов без такого мощного оружия, как пакеты прикладных программ.

*Экономьте время —  
пользуйтесь  
готовыми пакетами  
прикладных программ*

Затраты на создание сложных пакетов порой в несколько раз превышают стоимость самих ЭВМ. Сегодня уже недостаточно просто сконструировать ЭВМ. Чтобы она могла приносить пользу, должно быть создано и программное обеспечение — пакеты прикладных программ для решения наиболее популярных классов задач. С некоторыми из этих классов задач вы уже познакомились: расчеты физических экспериментов (§ 14, 15), задачи, решаемые методом Монте-Карло (§ 18), экономические задачи (§ 10, 23, 26), задачи обработки больших объемов данных (§ 3), текстов (§ 4) и изображений (§ 5), задачи прогнозирования (§ 21, 25) и т. д.

Пакетов прикладных задач разработано очень много: для расчетов траекторий спутников и параметров сложнейших технологических процессов, для решения проблем медицинской диагностики и управления каскадами гидроэлектростанций. Знакомые вам ИПС, графический и текстовый редакторы, электронная таблица относятся к простейшим прикладным программам.

В этом параграфе речь пойдет еще об одной прикладной программе. Она называется "Оптим" и предназначена

для решения задач планирования. Вспомните: одну из таких задач вы решали в § 26. В ней шла речь о производственном участке, выпускающем изделия двух видов А и Б. Требовалось так спланировать выпуск этих изделий ( $x$  тыс. изделий А и  $y$  тыс. изделий Б), чтобы получить максимальную прибыль. Вы решали эту задачу, перебирая всевозможные натуральные значения  $p$  и  $y$ , удовлетворяющие пяти неравенствам. Реально при составлении планов приходится учитывать гораздо больше ограничений и для большего числа переменных. Простым перебором не обойтись: даже если использовать сверхбыстрые ЭВМ, оптимальный план пришлось бы искать слишком долго. Математики придумали специальные алгоритмы, позволяющие решать задачи планирования во много раз быстрее. Один из таких алгоритмов (он называется "Симплекс-метод") реализован в "Оптиме".

На лабораторной работе вы с помощью "Оптимы" решите следующую задачу. В ней идет речь о том же производственном участке, что и в § 26.

**Задача.** Начальник участка изучает возможность расширить ассортимент товаров — добавить к выпускаемым изделиям А и Б еще два вида изделий: В и Г. Предварительное изучение спроса показало, что можно реализовать не более 5 тыс. изделий В, получив при этом прибыль в размере 1200 р. с каждого изделия. Можно также реализовать не более 6 тыс. изделий Г, получив прибыль 1000 р. с изделия. На тысячу изделий В расход металла составляет 0,5 т, электроэнергии — 4 тыс. кВт·ч, рабочего времени — 5 тыс. ч. Для выпуска тысячи изделий Г требуется 1,5 т металла, 4 тыс. кВт·ч электроэнергии, 6 тыс. ч рабочего времени. Расширение ассортимента изделий потребует приобретения дополнительного оборудования на сумму 800 тыс. р. (она будет возмещена из прибыли участка). Целесообразно ли расширение ассортимента выпускаемых товаров (т. е. можно ли спланировать выпуск товаров А, Б, В, Г так, чтобы получить прибыль большую, чем при выпуске только товаров А и Б)?

Математическая модель этой задачи составляется так же, как в § 26. Вся разница — в количестве переменных (теперь их четыре) и количестве ограничений (их девять). Чтобы не использовать много различных букв, будем обозначать количества товаров А, Б, В и Г (в тыс. штук) через  $x(1)$ ,  $x(2)$ ,  $x(3)$  и  $x(4)$ .

Ограничения запишутся так:

$$x(1) \geq 2$$

(надо произвести не менее 2 тыс. изделий А);

$$x(2) \geq 3$$

(надо произвести не менее 3 тыс. изделий Б);

$$3x(1)+x(2)+0,5x(3)+1,5x(4)\leq 32$$

(нельзя израсходовать больше 32 т металла);

$$3x(1)+6x(2)+4x(3)+4x(4)\leq 54$$

(нельзя израсходовать больше 54 тыс. кВт·ч электроэнергии);

$$3x(1)+3x(2)+5x(3)+6x(4)\leq 36$$

(на участке работают всего 20 человек);

$$x(3)\leq 5$$

(нельзя реализовать больше 5 тыс. изделий В);

$$x(4)\leq 4$$

(нельзя реализовать больше 4 тыс. изделий Г);

$$x(3)\geq 0,$$

$$x(4)\geq 0$$

(нельзя выпустить отрицательное число изделий).

Прибыль подсчитывается по формуле

$$500x(1)+700x(2)+1200x(3)+1000x(4)-800.$$

Наша цель — составить такой производственный план (т. е. найти такие значения  $x(1)$ ,  $x(2)$ ,  $x(3)$  и  $x(4)$ ), который обеспечит максимальную прибыль. Таким образом, задача нахождения наилучшего производственного плана свелась к задаче определения максимального значения функции при некоторых ограничениях. Такие задачи называют **задачами оптимизации**, а функции, максимум (или минимум) которых надо найти, — **целевыми функциями**.

Решая задачу на ЭВМ с помощью "Оптимы" (как и с помощью электронных таблиц), достаточно построить математическую модель; ни алгоритм, ни программу составлять не потребуется.

## Вопросы

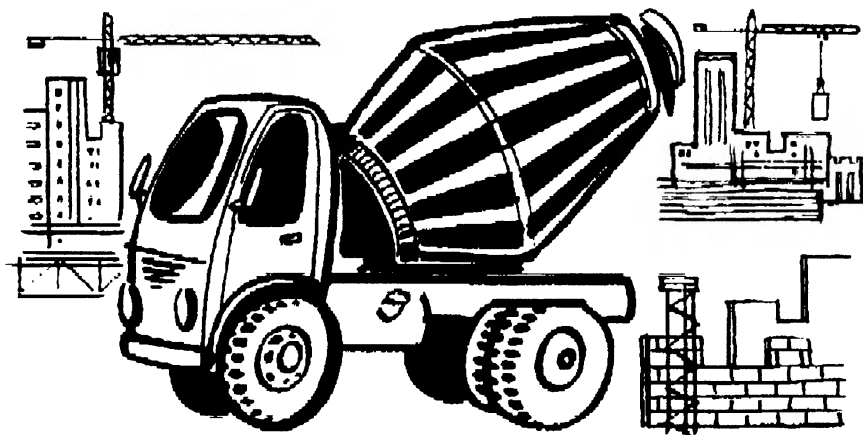
1. Какие программы называются прикладными?
2. Для чего нужны пакеты прикладных программ?
3. Что такое задача оптимизации?
4. Какая функция называется целевой?



## Задание для самостоятельного выполнения

Составьте математическую модель следующей транспортной задачи, записав систему ограничений и целевую функцию.

**З а д а ч а.** Бетон, производимый на заводах А и В, нужно развезти по трем стройплощадкам:  $N1_1$ ,  $N2_2$  и  $N3_3$ . Известны потребности стройплощадок в бетоне, запасы бетона на каждом заводе и затраты на перевозку 1 т бетона



от каждого завода до каждой стройплощадки. Требуется составить такой план перевозок, который обеспечивал бы наименьшие затраты.



## Лабораторная работа 17

### РАБОТА С ПРИКЛАДНОЙ ПРОГРАММОЙ "ОПТИМА"

На экране дисплея — незаполненная таблица, а выше нее заголовок "Симплекс-метод" (рис. 49). Это значит, что в ваш персональный компьютер загружена "Оптимa" и она ждет ваших указаний.

Симплекс-метод  
Таблица коэффициентов ограничений

№ ограничения	Коэффициенты при переменных						Знак	Свободный член
	x(1)	x(2)	x(3)	x(4)	x(5)	x(6)		
(1)								
(2)								
(3)								
(4)								
(5)								
(6)								
(7)								
(8)								
(9)								
F:								
[F1]:Минимизировать F    [F2]:Максимизировать F [F3]:Показать систему ограничений [F4]:Включить/выключить рабочую строку    [F5]:Убрать строку								

Начать надо с заполнения таблицы. В строки, помеченные номерами, запишите коэффициенты линейных неравенств или равенств — ведь среди ограничений могут быть и равенства. Чтобы перемещать курсор по таблице, пользуйтесь клавишами со стрелками и клавишей "ПЕРЕХОД".

Ограничения вида  $x \leq 0$  в таблицу можно не записывать: "Оптим" заранее считает, что все переменные неотрицательны.

Исправить ошибки вы можете с помощью клавиши "УДАЛИТЬ СИМВОЛ". Если же строка вообще никуда не годится, сотрите ее, нажав на клавишу <F5>.

Точно так же заполняется строка "F", в которой записываются коэффициенты целевой функции.

Последнее, что надо сообщить "Оптиме", — цель поиска. Что ей искать — максимум целевой функции или ее минимум? Нажав на нужную функциональную клавишу, вы получите ответ: слева появятся оптимальные значения переменных, справа — значение целевой функции. Тут же "Оптим" вам напомнит, какое значение найдено — максимальное или минимальное.

Наверно, вам захочется посмотреть на набранные вами ограничения в привычном, натуральном виде. Нажмите на клавишу <F3> — и система ограничений вместе с целевой функцией перед вами. Еще одно нажатие на <F3> — и перед вами снова таблица коэффициентов.

Теперь, разобравшись в этой несложной инструкции, приступайте к делу. Для начала найдите оптимальный план к задаче из § 26. Сравните время, затраченное вами на получение результата при помощи прикладной программы "Оптим", с тем, которое ушло на составление алгоритма, написание и отладку программы при выполнении лабораторной работы 16. Как видите, разница внушительная.

Пора приниматься за изучение возможностей расширения ассортимента выпускаемых товаров. Новая задача (см. § 27) отличается от только что решенной количеством переменных, числом ограничений и целевой функцией. Произведите эти изменения и получите ответ. Так стоит ли расширять ассортимент?

Жизнь, однако, не стоит на месте. Каждый день приносит что-нибудь новенькое. Представьте себе: от ваших смежников пришла телеграмма, что из 32 т металла они решили 24 т передать другому предприятию, а на вашу долю оставить только 8 т. Тут уж не до прибыли, лишь бы госзаказ выполнить! Замените 32 на 8 и посмотрите, какой ответ вам даст компьютер. "Система ограничений несовместна", — напечатает он. Это означает, что нет таких значений переменных, которые удовлетворяют всем ограни-

чениям. Похоже, что по вине смежников вам не удастся выполнить госзаказ — металла не хватит. А какое минимальное количество металла необходимо? Надеемся, что доводы вашей ЭВМ окажутся убедительными для смежников и вы получите металл в достаточном количестве.

Решив задачу 1, вы составили математическую модель транспортной задачи. Найдите оптимальный план перевозок при следующих исходных данных:

потребности стройплощадок в бетоне: №1 — 200 т, №2 — 280 т, №3 — 220 т;

запасы заводов: А — 320 т, В — 380 т;

таблица затрат на перевозку 1 т бетона:

Завод	Затраты на перевозку по стройплощадкам		
	№1	№2	№3
А	200 р.	400 р.	600 р.
В	500 р.	500 р.	300 р.

Не правда ли, это очень просто и удобно — пользоваться "Оптимой" для решения задач планирования?

## § 28. НЕМНОГО СТАТИСТИКИ

Каждый день вы ходите в школу и обратно. Сколько шагов вы делаете, преодолевая это расстояние? Если в течение нескольких дней вы из любопытства проведете подсчеты, то наверняка у вас получатся близкие друг другу, но все же разные числа. Никому, конечно, и в голову не придет, что меняется расстояние между школой и домом. Ясно, что на количество шагов влияют различные внешние факторы. Скажем, в школу вы шли быстро, чтобы не опоздать, — ваш шаг был шире, а по дороге домой вы шли не спеша, с одноклассницей — ваш шаг был короче. Можно сказать, что количество шагов от школы до дома — величина случайная. Проведя двадцать наблюдений, вы получите двадцать значений случайной величины.

Например, пусть некий школьник И\*\*\* получил такую последовательность чисел: 372, 376, 374, 375, 373, 364, 380, 374, 377, 375, 376, 373, 375, 374, 373, 371, 375, 373, 374, 376. Какое же количество шагов естественно взять в качестве расстояния от школы до дома? Каждому ясно — среднее арифметическое. Нетрудно подсчитать, что получится 374. Разумеется, если хочется узнать расстояние поточнее, надо пройти его больше двадцати раз.

*Усредняйте значения — это уберет вас от ошибочных выводов!*

Как же определить, насколько точно мы узнали расстояние, проведя то или иное количество наблюдений? Математики для этой цели ввели специальную величину и

*Дисперсия — главный свидетель разброса данных*

назвали ее дисперсией (от *dispersion* — "разброс"). Обозначим значения случайной величины  $a_1, \dots, a_n$ , а среднее арифметическое этих значений — буквой  $M$ .

Дисперсия — это среднее арифметическое квадратов разностей между значениями случайной величины и ее средним значением. В наших обозначениях

$$D = \frac{(a_1 - M)^2 + \dots + (a_n - M)^2}{n}.$$

Из этой формулы видно, что, чем меньше дисперсия, тем меньше отличаются результаты наблюдений от своего среднего значения и тем ближе среднее значение к истинному. В частности, если  $D=0$ , то все числа  $a_i$  совпадают между собой (и со своим средним значением).

Конечно, среднее значение числа шагов от дома до школы характеризует не только расстояние, но и длину шага человека: у разных людей и длина шага разная. Куда больше можно узнать о человеке, его характере, темпераменте и некоторых наклонностях, зная всю последовательность наблюдений.

Например, рассматривая приведенную выше последовательность, полученную И\*\*\*, можно предположить, что значение 364 получилось в тот день, когда он опаздывал в школу. Вообще же характер у И\*\*\* довольно ровный, темперамент скорее флегматический — лишь один раз (получив, наверное, двойку) он шел заметно медленнее, чем обычно, сделав 380 шагов. Подумайте, что еще можно сказать об И\*\*\*.

Допустим теперь, что И\*\*\* сагитировал нескольких своих товарищей провести тот же эксперимент. Через 10 дней каждый из них, в том числе и сам И\*\*\*, представил по 20 результатов наблюдений, не указав свою фамилию. Можно ли узнать, какие из результатов принадлежат И\*\*\*, а какие нет? Да, можно!

Математики установили, что для этого, как правило, достаточно сравнить дисперсии и средние значения. Дисперсия и среднее значение почти так же индивидуальны, как отпечатки пальцев. Если наблюдения делал один и тот же человек, то дисперсии и средние значения будут близки, если разные люди, то далеки.

Осталось выяснить: какие значения считать близкими,

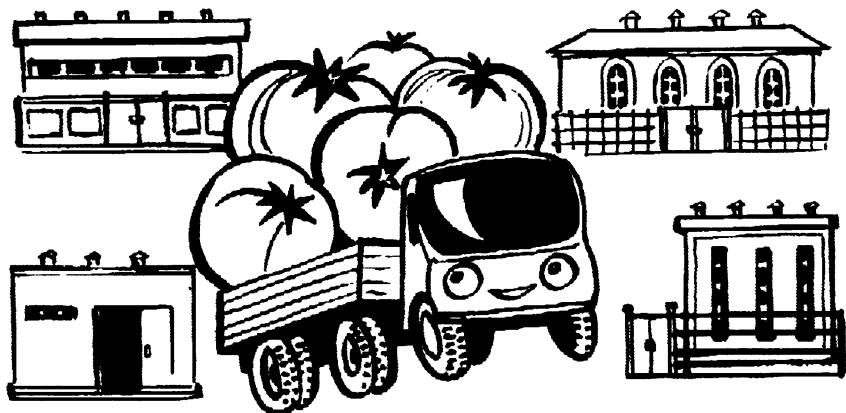
а какие далекими? На этот вопрос дает ответ специальный раздел математики — статистика. Оказывается, достоверность ответа зависит от числа наблюдений. Если число наблюдений от 20 до 50, то дисперсии можно считать далекими, когда отношение большей дисперсии к меньшей больше 2. Чтобы говорить о близости средних значений двух последовательностей результатов, надо найти модуль разности средних и разделить его на квадратный корень из суммы дисперсий. Если полученное число больше 0,6, то средние значения считают далекими. В том случае, когда близки и дисперсии и средние значения, можно сделать вывод, что наблюдения почти наверняка проводились одним и тем же человеком.

У любознательного ученика может возникнуть вопрос: откуда эти числа (2 и 0,6) взялись? Отвечаем: из специальных таблиц, которые были составлены математиками. Их можно найти в любом справочнике по математической статистике.

Метод сравнения средних значений и дисперсий используется в самых разных областях человеческой деятельности. В медицине — для установления диагноза, в литературоведении — для определения автора произведений, в криминалистике — для розыска преступников.

На лабораторной работе вы как раз и займетесь расследованием одного преступления.

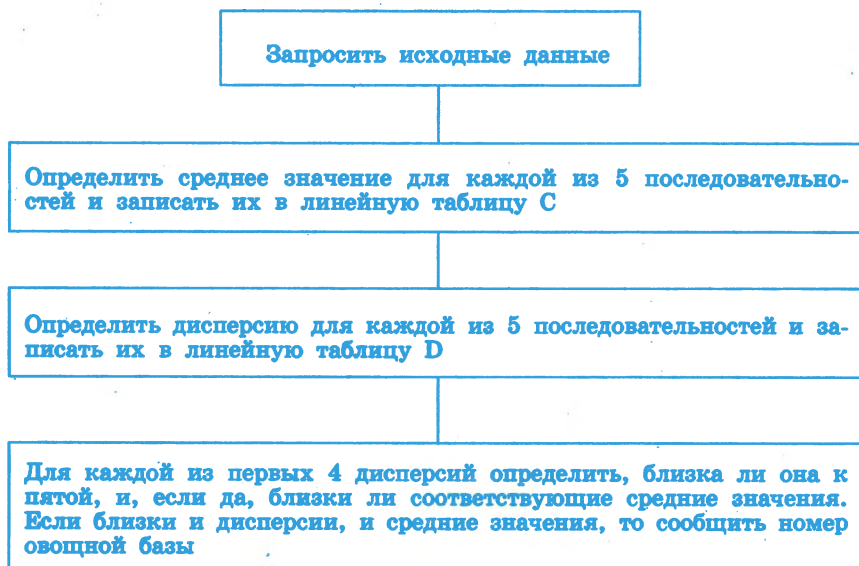
**З а д а ч а.** Органами милиции задержан грузовик с помидорами, похищенными на овощной базе. В городе всего четыре овощные базы, каждая из них получает помидоры из своего сельскохозяйственного района. Определить, с какой базы были вывезены помидоры. Расследование осложняется тем, что помидоры на всех базах одного сорта.



Воспользуемся методом сравнения средних значений и дисперсий. В каждом сельскохозяйственном районе свои условия произрастания помидоров, поэтому помидоры разных районов отличаются, скажем, удельным весом. Выберем по двадцать помидоров (реально, конечно, их выбирается гораздо больше) на каждой овощной базе и из грузовика. У нас получится четыре последовательности, по двадцать значений в каждой (для каждой овощной базы), и еще одна (для грузовика), с которой мы и будем сравнивать первые четыре. Это наши исходные данные. Результатом является номер овощной базы, где совершенно хищение.

Чтобы добиться этого результата, нужно, как рассказано выше, вычислить средние значения и дисперсии всех пяти последовательностей и провести сравнение. Напомним, что сначала сравниваются дисперсии, а затем для последовательностей, имеющих близкие дисперсии, сравниваются средние значения. Уже из сказанного ясно, что алгоритм получится достаточно сложным, так что мы будем строить его методом последовательной детализации.

Прежде всего организуем исходные данные. Последовательности для овощных баз и грузовика удобно записать в виде прямоугольной таблицы, имеющей 5 строк и 20 столбцов. Обозначим ее буквой А. Первые четыре строки таблицы А соответствуют четырем базам, а последняя строка — грузовику. Тогда самая общая блок-схема выглядит так:



Эта блок-схема показывает нам, что надо выделить вспомогательный алгоритм нахождения среднего значения и дисперсии 20 чисел, записанных в строке таблицы  $A$ , а также алгоритм упорядочения двух чисел.

Алгоритм "Среднее значение".

Аргумент:  $k$  (номер строки таблицы  $A$ ).

Результат:  $m$  (среднее значение).

Присвоить  $S$  начальное значение 0.

Для каждого  $j$  от 1 до 20:

Присвоить  $S$  значение  $S+A(k,j)$ .

Конец цикла по  $j$ .

Присвоить  $m$  среднее значение  $S/20$ .

Алгоритм "Дисперсия".

Аргумент:  $k$  (номер строки таблицы  $A$ ).

Результат:  $d$  (дисперсия).

Присвоить  $S$  начальное значение 0.

Для каждого  $j$  от 1 до 20:

Присвоить  $S$  значение  $S+(A(k,j)-C(k))^2$ .

Конец цикла по  $j$ .

Присвоить  $d$  среднее значение  $S/20$ .

Алгоритм "Упорядочение двух чисел".

Аргументы:  $x, y$  (числа, которые надо упорядочить)

Результаты:  $L, R$  ( $L$  — меньшее из чисел  $x$  и  $y$ ,  $R$  — большее из них).

Если  $x < y$ , то:

Присвоить  $L$  значение  $x$ .

Присвоить  $R$  значение  $y$ .

Иначе:

Присвоить  $L$  значение  $y$ .

Присвоить  $R$  значение  $x$ .

Конец ветвления.

Основной алгоритм решения задачи выглядит так (части алгоритма, соответствующие разным блокам блок-схемы, отделены друг от друга):

Запросить таблицу  $A(5,20)$ .

Для каждого  $i$  от 1 до 5:

Выполнить алгоритм "Среднее значение" при  $k=i$ .

Присвоить  $C(i)$  значение  $m$ .

Конец цикла по  $i$ .

Для каждого  $i$  от 1 до 5:

Выполнить алгоритм "Дисперсия" при  $k=i$ .

Присвоить  $D(i)$  значение  $d$ .

Конец цикла по  $i$ .

Для каждого  $i$  от 1 до 4:

Выполнить алгоритм "Упорядочение двух чисел" при  $x=D(i)$ ,  $y=D(5)$ .

Если  $R/L < 2$ , то:

Если  $|C(i) - C(5)| \sqrt{(D(i) + D(5))} < 0,6$ , то:

Сообщить "Номер базы".

Сообщить  $i$ .

Конец ветвления.

Конец ветвления.

Конец цикла по  $i$ .



### Задания для самостоятельного выполнения

1. Проведите следующий эксперимент. Двадцать раз подбросьте монету и при выпадении решки записывайте 1, а при выпадении орла — 0. Получится последовательность из 20 нулей и единиц. Каковы среднее значение и дисперсия для этой последовательности? Повторите эксперимент. Получились ли новые среднее значение и дисперсия близкими к предыдущим?

2. Составьте математическую модель и алгоритм решения следующей задачи:

Известны данные, показывающие продолжительность горения (в часах) электрических ламп, изготовленных на двух заводах.

Лампы первого завода:

1600, 1510, 1610, 1520, 1650, 1530, 1680, 1570, 1600, 1700, 1720, 1680, 1800, 1780, 1690, 1710, 1630, 1720, 1750, 1810.

Лампы второго завода:

1580, 1460, 1640, 1550, 1600, 1620, 1700, 1640, 1750, 1820, 1660, 1740, 1760, 1730, 1590, 1610, 1650, 1700, 1580, 1670.

Можно ли утверждать, что на заводах поддерживаются одинаковые технологические условия производства?



### Лабораторная работа 18

#### КРИМИНАЛИСТИЧЕСКАЯ ЗАДАЧА

В § 28 была разобрана криминалистическая задача, и вы написали алгоритм ее решения. Как вы помните, исходные данные о помидорах (они подготовлены экспертом, включенным в группу по расследованию хищения), организованы в виде таблицы, имеющей 5 строк (первые четыре строки содержат удельные веса двадцати наугад выбранных помидоров с четырех овощных баз, а пятая — с грузовика) и 20 столбцов. Эта таблица обозначена буквой  $A$  и запи-

сана на дискете, помещенной в дисковод вашей ЭВМ, или на кассете, вставленной в ваш магнитофон.

Поэтому первую команду вашей программы надо заметить такой командой:

Вызвать таблицу  $A(5,20)$ .

Измените программу, запустите ее и получите результат.

Но что это? ЭВМ выдала номера двух баз! Такое заключение эксперт постесняется предъявить следователю. Как же это могло произойти? Может быть, хищение произведено с двух баз? Но зачем преступнику подвергать себя двойному риску? А вы как считаете, в чем тут дело?

Тот, кто внимательно прочитал § 28, уже догадался, что двадцати помидоров для однозначного ответа, по видимому, маловато. Попробуем увеличить количество помидоров до тридцати. К счастью, предусмотрительный эксперт записал на вашу дискету данные о тридцати помидорах с каждой базы и с грузовика. Конечно, надо изменить программу, поменяв в ней 20 на 30. Надеемся, что теперь-то вам удастся определить место хищения.



## КОНСПЕКТ ГЛАВЫ 7

Составляя алгоритмы, важно заботиться не только об организации действий, но и об организации объектов, над которыми производятся действия. Эти объекты называются данными. Один из самых распространенных способов организации данных — табличный.

**Табличным** называется расположение данных в нескольких строках одинаковой длины. Если в таблице всего одна строка, то такую таблицу называют линейной. Если таблица состоит не менее чем из двух строк, то ее называют прямоугольной.

Для обозначения таблиц используют латинские буквы. Строки таблиц нумеруются сверху вниз, а столбцы — слева направо. Элемент, расположенный на  $i$ -м месте линейной таблицы  $A$ , обозначают  $A(i)$ , а элемент, расположенный на пересечении  $i$ -й строки и  $j$ -го столбца прямоугольной таблицы  $X$ , обозначают  $X(i,j)$ . Например, в таблице  $A$ :

7	2	6	34
4	5	0	-3
1	6	8	1,2

$A(1,1)=7$ ,  $A(2,3)=0$ ,  $A(3,2)=6$ ,  $A(3,4)=1,2$ .

Табличный способ расположения данных — основной для нового исполнителя — РОБОТА-МАНИПУЛЯТОРА. Данные, с которыми работает РОБОТ-МАНИПУЛЯТОР, — это микросхемы, транзисторы и другие радиодетали. Они

разложены в ячейках прямоугольного стеллажа. Каждая ячейка рассчитана на хранение только одной детали. РОБОТ снабжен грузовым отсеком для хранения деталей, взятых со стеллажа. Он может перемещаться вдоль стеллажа, переходя от ячейки к ячейке и не выходя при этом за пределы стеллажа.

Укажем список допустимых действий РОБОТА-МАНИПУЛЯТОРА:

1. Запросить наименование детали.
2. Переместиться к левой нижней ячейке стеллажа.
3. Переместиться к началу ряда.
4. Переместиться к соседней сверху (снизу, справа, слева) ячейке.
5. а) Переложить деталь из ячейки в грузовой отсек.  
б) Переложить деталь из грузового отсека в ячейку.  
(Ячейка, о которой идет речь в пунктах а) и б), — та, напротив которой находится РОБОТ.)
6. Проверить одно из следующих условий:
  - а) можно шагнуть вверх (вниз, вправо, влево) или нельзя;
  - б) есть деталь в ячейке, напротив которой находится РОБОТ, или нет;
  - в) совпадает наименование детали в ячейке, напротив которой находится РОБОТ, с наименованием запрошенной детали или не совпадает;
  - г) пуст грузовой отсек или не пуст.

Вот пример алгоритма для РОБОТА-МАНИПУЛЯТОРА. Выполняя этот алгоритм, РОБОТ запросит наименование детали, а затем соберет в грузовой отсек все такие детали из нижнего ряда стеллажа:

Запросить наименование детали.

Переместиться к началу стеллажа.

Пока можно идти вправо, повторять:

Если наименование детали, лежащей в ячейке, совпадает с наименованием запрошенной детали, то:

Переложить деталь из ячейки в грузовой отсек.

Конец ветвления.

Шаг вправо.

Конец цикла.

Если наименование детали, лежащей в ячейке, совпадает с наименованием запрошенной детали, то:

Переложить деталь из ячейки в грузовой отсек.

Конец ветвления.

Табличная форма организации данных полезна и при решении вычислительных задач. Для работы с таблицами ВЫЧИСЛИТЕЛЬ имеет следующие допустимые действия:

1. Запросить таблицу.
2. Сообщить таблицу.

После слов "таблицу" указывают обозначение таблицы, а также (например, в скобках) число строк и столбцов в ней:

Запросить прямоугольную таблицу  $A$  из 12 строк и 2 столбцов.

Запросить линейную таблицу  $B$  из 15 элементов.

Кроме того, в команде присваивания могут использоваться, кроме обычных переменных, переменные, являющиеся элементами таблиц. Номера строк и столбцов могут быть как числами, так и алгебраическими выражениями:  $A(1,2)$ ,  $B(I+J)$ ,  $C(I+1,2J)$ . Вот пример команды присваивания:

Присвоить  $X$  значение  $C(2)\sin(B(K)+X)$ .

При работе с таблицами полезна еще одна форма записи цикла — цикл "Для каждого":

Для каждого  $i$  от  $L$  до  $R$ :

$P_1$

$P_2$

$\dots$

$P_n$

Конец цикла по  $i$ .

Выполняя этот цикл, ВЫЧИСЛИТЕЛЬ сначала присваивает  $i$  значение  $L$ . Затем он проверяет, больше ли  $i$ , чем  $R$ ; если да, то цикл заканчивается (не начавшись), если нет, то ВЫЧИСЛИТЕЛЬ выполнит действия  $P_1, P_2, \dots, P_n$ , число  $i$  увеличивается на 1, и для этого нового значения  $i$  повторяется тело цикла до тех пор, пока  $i$  не превзойдет  $R$ . Разумеется, можно использовать другие обозначения для счетчика и границ цикла. Кроме того, вместо  $L$  и  $R$  можно писать алгебраические выражения или просто числа. Переменная  $i$  выступает в роли счетчика.

Чтобы ВЫЧИСЛИТЕЛЬ, имитируемый с помощью ЭВМ, был способен работать с таблицей, в начале программы нужно сообщить размеры таблицы. Это делают командой

ТАБЛИЦА...,

где вместо многоточия пишут имя таблицы и ее размеры в круглых скобках. По этой команде ЭВМ резервирует в памяти место, где будет размещаться таблица.

# ЯЗЫК ПРОГРАММИРОВАНИЯ БЕЙСИК

---

Из предыдущих глав вы узнали, что такое компьютерная модель и алгоритм. Познакомились вы и с некоторыми исполнителями алгоритмов (ЧЕРТЕЖНИК, РОБОТ-МАНИПУЛЯТОР, ВЫЧИСЛИТЕЛЬ). Но предназначены они не столько для решения практических задач, сколько для изучения основ алгоритмизации. В этой и последующих главах мы расскажем вам о тех средствах, которые позволяют решать с помощью ЭВМ широкий круг задач, возникающих на производстве, в науке и технике.

Для того чтобы ЭВМ "поняла" алгоритм решения задачи и выполнила его, нужно записать этот алгоритм на специальном языке. Такие языки (их называют **языками программирования** или **алгоритмическими языками**) являются одним из основных средств общения человека и компьютера. Напомним, что алгоритм, записанный на языке программирования, называется **программой**.

В настоящее время существуют сотни различных языков программирования. Каждая ЭВМ, как правило, полиглот: она "знает" несколько языков. Компьютер, с которым общаетесь вы, "знает" язык программирования Бейсик. Этот язык был создан в 1965 г., и в настоящее время значительное число программ для микроЭВМ написано на нем. Бейсик — один из наиболее простых языков

*Если ваша ЭВМ  
не понимает  
язык Бейсик,  
значит это —  
не ЭВМ*

программирования. Правда, некоторые затруднения может вызывать то, что все слова в Бейсике английские. Впрочем, английский язык положен разработчиками в основу большинства языков программирования. Если же вы не изучали английский язык, не огорчайтесь: чтобы "объясняться" на Бейсике, вам придется запомнить не более двух десятков английских слов. Как и многие естественные языки (русский, английский и т. д.), Бейсик имеет много "диалектов", различающихся правилами записи некоторых команд, но по сути одинаковых. И вообще в отличие от естественных языков большинство языков программирова-

ния "похожи" друг на друга, и, изучив один, несложно изучить и другой. Главное — знать, как вообще изучить язык программирования. В этой главе на примере языка Бейсик мы покажем, как это сделать.

На предыдущих занятиях вы составили и записали уже много алгоритмов. Вспомните: любой алгоритм состоит из допустимых действий, организованных с помощью алгоритмических конструкций ветвлений, циклов, вспомогательных алгоритмов. И каждая из этих конструкций оформлялась для всех исполнителей одинаково. Разумеется, во всех алгоритмических языках предусмотрены все эти алгоритмические конструкции. Только оформляются они в разных языках по-разному. Изучить алгоритмический язык — значит узнать, как в нем называются те или иные допустимые действия и как оформляются алгоритмические конструкции. Те из вас, кто не имел возможности работать с машинной реализацией **ВЫЧИСЛИТЕЛЯ**, **ЧЕРТЕЖНИКА** и **РОБОТА-МАНИПУЛЯТОРА**, уже познакомились с некоторыми элементами языка Бейсик. Ну что ж, теперь они расширят свои знания.

## § 29. ОСНОВНЫЕ КОМАНДЫ ЯЗЫКА БЕЙСИК

Прежде всего объясним основные правила записи программ на Бейсике. В начале каждой строки программы ставится номер. В памяти машины строки программы всегда располагаются в порядке возрастания номеров. При записи программы на языке Бейсик необязательно записывать команды столбиком — можно несколько команд записывать в одной строке, разделяя их двоеточием. Если же вы хотите, чтобы ЭВМ сразу исполнила вашу команду, не запоминая ее, введите эту команду без номера.

Система команд языка Бейсик очень похожа на систему команд для **ВЫЧИСЛИТЕЛЯ**: она позволяет главным образом решать вычислительные задачи в так называемом диалоговом режиме. Это означает, что в ходе работы компьютер запрашивает исходные данные и сообщает результаты вычислений. Как вы помните, **ВЫЧИСЛИТЕЛЬ** работает так же.

Хотя допустимые действия **ВЫЧИСЛИТЕЛЯ** может выполнить и ЭВМ, настроенная на язык Бейсик, **ВЫЧИСЛИТЕЛЬ** предоставляет гораздо большие возможности для организации действий. Вы это увидите, познакомившись с организацией ветвлений и подпрограмм в языке Бейсик.

Действию "Присвоить X значение F" соответствует команда

**LET X=F.**

Слово **LET** означает "пусть". Эта команда называется ко-

мандой присваивания. Выполнив ее, ЭВМ вычислит значение выражения F и присвоит полученное значение переменной X. Версии Бейсика на многих ЭВМ позволяют опускать слово LET в команде присваивания. Мы тоже не будем далее писать слово LET.

Переменную в Бейсике можно обозначать и латинской буквой, и буквой с цифрой, а также двумя буквами. Например, A, C5, G0, AV и т. д. Для записи операций используются уже знакомые вам знаки:

- + сложение;
- вычитание;
- \* умножение;
- / деление;
- ^ возведение в степень.

Вот пример программы, содержащей команды присваивания:

```
11 A1=2
25 B1=6/2
36 C=(A1+B1)*A1
```

Выясните, какое значение примет переменная C после выполнения этой программы.

Обозначения в Бейсике для функций, изучаемых в школе ( $\sin x$ ,  $\cos x$ ,  $\ln x$  и т. д.), отличаются от обычных. Приведем эти обозначения:

Функция	Обозначение в Бейсике
$\sin x$	SIN(X)
$\cos x$	COS(X)
$\operatorname{tg} x$	TAN(X)
$\operatorname{arctg} x$	ATN(X)
$\ln x$	LOG(X)
$e^x$	EXP(X)
$\sqrt{x}$	SQR(X)
$ x $	ABS(X)

Обратите внимание: в Бейсике аргумент функции всегда записывается в скобках. Аргументом функции может служить как число, так и произвольное выражение.

Действию "Запросить A,B" на языке Бейсик соответствует команда

### INPUT A,B

(INPUT — ввести). Здесь A и B — переменные, значения которых ЭВМ запросит (разумеется, ЭВМ может запросить значения любого количества переменных). При выполнении этой команды ЭВМ печатает на экране дисплея знак вопроса, останавливается и ждет, пока в нее не будет введено два числа. Числа набираются на клавиатуре ЭВМ одно за другим и разделяются запятой. Не забудьте после набора чисел нажать на клавишу "ПЕРЕВОД СТРОКИ". Первое из

введенных чисел будет присвоено переменной А, второе — переменной В.

Если же ввести в ЭВМ строку

**INPUT "ВВЕДИТЕ ТРИ ЧИСЛА";А,В,С ,**

то на экране сначала появится надпись "ВВЕДИТЕ ТРИ ЧИСЛА?", после чего ЭВМ будет ждать ввода трех чисел. Как видите, с помощью команды INPUT можно выдавать на экран поясняющие сообщения.

Действию "Сообщить" на Бейсике соответствует команда

**PRINT...**

В переводе с английского PRINT означает "печатать". Вместо многоточия после слова PRINT может стоять текст (последовательность букв и других символов, изображенных на клавиатуре), заключенный в кавычки, а также переменные или выражения, значения которых мы хотим увидеть на экране ЭВМ. Например, по команде

**PRINT "КОРНИ УРАВНЕНИЯ" X1,X2**

сначала на экране дисплея появится сообщение КОРНИ УРАВНЕНИЯ, а затем значения переменных X1 и X2.

В языке Бейсик есть специальные команды, по которым ЭВМ прекращает работу по программе. Это команды STOP (стоп) и END (конец). Команду END располагают обычно в конце программы.

Вот пример программы, в которой использованы все основные команды Бейсика. Вы, конечно, сами легко поймете, для чего она предназначена.

```
10 PRINT "СООБЩИТЕ ЗНАЧЕНИЕ СКОРОСТИ"  
20 INPUT V  
30 INPUT "СООБЩИТЕ ВРЕМЯ";T  
50 S=V*T  
60 PRINT "РАССТОЯНИЕ S="S  
70 END
```

## **? Вопросы**

1. Для чего нужны номера строк в программе на языке Бейсик?

2. Как в языке Бейсик отделяются друг от друга команды, стоящие в одной строке?

3. По какой команде языка Бейсик ЭВМ запрашивает данные?

4. По какой команде языка Бейсик ЭВМ печатает результаты?

5. Какие команды используются для прекращения работы ЭВМ по программе?



## Задания для самостоятельного выполнения

1. Допустим, вы ввели в ЭВМ команды в следующем порядке:

```
40 PRINT X,Y
25 A=2:B=3
32 X=A+B
31 A=A+B
34 Y=A*B
```

В каком порядке эти команды расположатся в памяти ЭВМ? Какие числа напечатает ЭВМ, выполнив эту программу?

2. Запишите на языке Бейсик следующие выражения:

а)  $\frac{5\sin x}{z^2+1}$       б)  $\frac{z^2-3z+17}{\operatorname{ctg} y}$       в)  $\log_2 5 + (12,5 - e^2)$ .

3. Что напечатает ЭВМ, выполняя следующие команды, если  $X=4$ ,  $A=2$ ,  $S=60$ ?

а) PRINT "X="X  
б) PRINT 6\*X-A^3  
в) PRINT "РАССТОЯНИЕ " S " КМ"

4. По какой из приведенных ниже команд ЭВМ напечатает значение переменной X?

а) PRINT "X"      в) PRINT X      д) PRINT X=5  
б) INPUT X      г) INPUT "X"

5. В программе нахождения длин диагоналей параллелограмма по смежным сторонам и углу между ними злоумышленник стер все знаки препинания. Восстановите их.

```
10 INPUT ВВЕДИТЕ ДЛИНЫ СТОРОН A B
20 INPUT ВВЕДИТЕ ВЕЛИЧИНУ УГЛА В РАДИАНАХ Z
30 L=SQR(A^2+B^2-2*A*B*COS(Z))
   R=SQR(A^2+B^2+2*A*B*COS(Z))
40 PRINT ДЛИНЫ ДИАГОНАЛЕЙ РАВНЫ L R
```

6. Составьте программу вычисления площади треугольника по трем сторонам  $a$ ,  $b$ ,  $c$ . (Указание: используйте формулу Герона.)

## § 30. ВЕТВЛЕНИЯ В ЯЗЫКЕ БЕЙСИК

В этом параграфе мы покажем, как оформляются на языке Бейсик разветвляющиеся программы.

Простейшей командой, изменяющей естественный порядок выполнения программы, является команда GOTO (GO TO — "перейти к..."). Эта команда имеет вид:

**GOTO N,**

где N — номер строки программы. По этой команде ЭВМ переходит к выполнению команды с номером N.

Вы знакомы с двумя способами записи ветвлений — неполной и полной. Ветвление в неполной форме записывалось так:

**Если  $Q$ , то:**

**$P$**

**Конец ветвления**

( $Q$  — условие, а  $P$  — последовательность команд, которую надо выполнять в случае, когда условие верно). Можно поразному перевести на язык Бейсик эту форму записи. Если  $P$  состоит из одного действия, то перевод такой:

**IF  $Q$  THEN  $P$**

(слово IF означает "если", а слово THEN — "то"). Например:

```
10 IF  $X > 0$  THEN  $Y = \text{LOG}(X)$   
20 ...
```

Если значение переменной  $X > 0$ , то ЭВМ вычислит  $\ln X$  и присвоит результат переменной  $Y$ , а затем будет выполняться команда с номером 20. Если же значение  $X \leq 0$ , то сразу будет выполняться команда с номером 20.

Допустим теперь, что последовательность  $P$  состоит из нескольких действий. Тогда сокращенная форма ветвления записывается сложнее. Пусть в  $P$  содержится 6 действий  $P_1, \dots, P_6$ , ветвление должно начинаться со 120-й строки, а следующая после ветвления команда имеет номер 150. Это означает, что в случае невыполнения условия  $Q$  надо перейти к команде 150. Поэтому запись будет такой:

```
120 IF NOT( $Q$ ) THEN GOTO 150  
121  $P_1$   
122  $P_2$   
.....  
126  $P_6$   
150 ...
```

Здесь NOT( $Q$ ) означает отрицание условия  $Q$  (NOT — "не"). Например, NOT( $X < Y$ ) означает  $X \geq Y$ , а NOT( $X = Y$ ) означает  $X \neq Y$ .

Отметим еще, что в строках вида

**IF ... THEN GOTO  $K$  ,**

где  $K$  — номер строки, одно из слов, GOTO или THEN, можно не писать, т. е. три команды

```
IF ... THEN  $K$   
IF ... GOTO  $K$   
IF ... THEN GOTO  $K$ 
```

означают одно и то же.

Например, алгоритм нахождения максимума из двух чисел:

Запросить  $A, B$ .  
Присвоить  $M$  значение  $A$ .  
Если  $M < B$ , то:  
Присвоить  $M$  значение  $B$ .  
Конец ветвления.  
Сообщить  $M$

на языке Бейсик можно записать так:

```
10 INPUT A,B
20 M=A
30 IF M<B THEN M=B
40 PRINT M
50 END
```

Алгоритм можно перевести на язык Бейсик и иначе:

```
10 INPUT A,B
20 M=A
30 IF NOT(M<B) THEN 50
40 M=B
50 PRINT M
60 END
```

Покажем теперь, как записать на языке Бейсик команду ветвления в полной форме:

Если  $Q$ , то:  
 $P$   
Иначе:  
 $T$   
Конец ветвления.

Аналогично предыдущему, если  $P$  и  $T$  состоят из одного действия, удобно переводить так (ELSE — иначе):

IF  $Q$  THEN  $P$  ELSE  $T$

Если же  $P$  или  $T$  состоят из нескольких действий, то удобнее переводить следующим образом (для примера пусть  $P$  состоит из трех,  $T$  — из четырех команд, ветвление начинается с 530-й строки, следующая за ветвлением строка имеет номер 600):

```
530 IF Q THEN 536
531 T1
532 T2
533 T3
534 T4
535 GOTO 600
536 P1
537 P2
538 P3
600 ...
```

Проверьте, что, исполняя этот фрагмент, ЭВМ выполнит команды  $P_1, P_2, P_3$ , если условие  $Q$  выполняется, и команды  $T_1, T_2, T_3, T_4$ , если  $Q$  не имеет места. А затем после выполнения соответствующей серии команд ЭВМ перейдет к выполнению команды, записанной в строке 600.

Рассмотрим пример программы для вычисления значения функции по следующему правилу:

$$y = \begin{cases} \sqrt{x}, & \text{если } x \geq 2; \\ x, & \text{если } x < 2. \end{cases}$$

```
10 INPUT X
20 IF X>=2 THEN Y=SQR(X) ELSE
Y=ABS(X)
30 PRINT"Y="Y
40 END
```

Можно было эту программу записать и так:

```
10 INPUT X
20 IF X>=2 THEN 50
30 Y=ABS(X)
40 GOTO 60
50 Y=SQR(X)
60 END
```

## ? Вопросы

1. По какой команде можно заставить ЭВМ, не проверяя никаких условий, изменить порядок выполнения действий?

2. Как в языке Бейсик оформляется ветвление:

- а) в неполной форме;
- б) в полной форме?

## Задания для самостоятельного выполнения

1. Переведите на Бейсик алгоритмы для ВЫЧИСЛИТЕЛЯ, составленные вами при решении задач из главы 4.

2. Найдите и исправьте ошибки в следующей программе нахождения квадратного корня из числа:

```
10 INPUT X
20 IF X>=0 THEN 30
30 PRINT SQR(X)
40 GOTO 60
50 PRINT "Корень вычислить невозможно"
60 END
```

3. Злоумышленник стер команду 100 в следующей программе нахождения корней квадратного уравнения:

```
10 INPUT A,B,C
20 IF A<> 0 THEN 50
30 PRINT "Это уравнение не квадратное"
40 STOP
50 D=B^2-4*A*C
60 IF D<0 THEN 110
70 X1=(-B+SQR(D))/(2*A)
80 X2=(-B-SQR(D))/(2*A)
90 PRINT "Корни уравнения "X1,X2
100 ...
110 PRINT "Уравнение корней не имеет"
120 END
```

### § 31. ЦИКЛЫ В ЯЗЫКЕ БЕЙСИК

Цикл "Пока" мы записывали в следующем виде:  
Пока  $Q$ , повторять:

$P$

Конец цикла.

Здесь  $Q$  — условие,  $P$  — последовательность действий, которую надо выполнять, пока истинно условие  $Q$ .

В языке Бейсик цикл "пока" в зависимости от реализации на ЭВМ оформляется одним из двух способов:

10 WHILE <условие>	10 IF NOT <условие> THEN 40
20 P	20 P
30 WEND	30 GOTO 10
40 ...	40 ...

В первом варианте слово "пока" заменяется на WHILE, слово "повторять" опускается, слова "конец цикла" заменяются на WEND. Нетрудно проверить, что, выполняя последовательность команд из второго варианта, ЭВМ будет повторять действия из последовательности  $P$  до тех пор, пока условие  $Q$  истинно.

Посмотрите на следующем примере, как выглядит такой перевод:

Алгоритм

Запросить  $a, b$ .

Присвоить  $n$  значение 0.

Пока  $a < b$ , повторять:

Присвоить  $a$  значение  $10a$ .

Присвоить  $n$  значение  $n+1$ .

Сообщить "Значения  $a, n$ ".

Сообщить  $a, n$ .

Конец цикла.

Программа на Бейсике

10 INPUT A,B

20 N=0

30 IF A>=B THEN 90

40 A=10\*A

50 N=N+1

60 PRINT "Значения A,N"

70 PRINT A,N

80 GOTO 30

90 END

Подумайте, для решения какой задачи предназначен этот алгоритм.

Теперь о цикле "Для каждого". Вот его запись:

**Для каждого  $I$  от  $L$  до  $R$ :**

**$P$**

**Конец цикла по  $I$**

(Напомним:  $I$  — счетчик цикла,  $L$  — начальное значение счетчика,  $R$  — конечное значение счетчика,  $P$  — тело цикла.)

В языке Бейсик цикл "Для каждого" оформляется так:

**10 FOR I=L TO R**

**20 P**

**30 NEXT I**

(т.е. слова "Для каждого" заменяются на FOR, "от" — на знак равенства, "до" — на TO, "конец цикла по" на NEXT).

Составим программу, выполняя которую ЭВМ напечатает кубы целых чисел от 10 до 100.

**10 FOR X=10 TO 100**

**20 PRINT X^3**

**30 NEXT X**

**40 END**

## ? Вопросы

1. Как оформляется на Бейсике цикл "пока"?
2. Как оформляется на Бейсике цикл "Для каждого"?
3. Какую роль играет команда NEXT?



## Задания для самостоятельного выполнения

1. Переведите на язык Бейсик алгоритмы для ВЫЧИСЛИТЕЛЯ, составленные вами при решении задач из главы 5. (У к а з а н и е: в языке Бейсик действию ВЫЧИСЛИТЕЛЯ "Присвоить Z значение RND(A)" соответствует команда  $Z=2*A*RND(1)-A$ .)

2. Найдите и исправьте ошибки в следующих программах:

а) Программа определения первого отрицательного члена последовательности  $a_n = \sin(n/100)$  ( $n$  — натуральное число):

**10 IF SIN(N/100)<0 THEN 40**

**20 N=N+1**

**30 GOTO 20**

**40 PRINT "ЗНАЧЕНИЯ N, SIN(N/10)" N, SIN(N/100)**

б) Программы определения суммы первых десяти положительных членов последовательности  $a_n = \cos(n)$ , где  $n$  —

натуральное число. В этих программах  $K$  обозначает количество найденных положительных членов последовательности, а  $S$  — сумму этих членов.

10 K = 0	10 K = 0
20 N = 1	20 N = 1
30 WHILE K <= 10	30 IF K > 10 THEN 70
40 IF COS(N) > 0 THEN	40 IF COS(N) > 0 THEN
K = K + 1: S = S + COS(N)	K = K + 1: S = S + COS(N)
50 N = N + 1	50 N = N + 1
60 WEND	60 GOTO 30
70 PRINT "ЗНАЧЕНИЕ	70 PRINT "ЗНАЧЕНИЕ
СУММЫ" S	СУММЫ" S

в) Программа нахождения значений  $\ln(n^2 - n)$  для целых  $n$  от  $-5$  до  $5$ :

```

10 FOR N=-5 TO 5: IF N^2-N<=0 THEN 40
20 PRINT "ПРИ N="N" ЗНАЧЕНИЕ ФУНКЦИИ НЕ
   ОПРЕДЕЛЕНО"
30 ELSE PRINT "ПРИ N="N" ЗНАЧЕНИЕ ФУНКЦИИ
   РАВНО "LOG(N^2-N)
40 NEXT I

```

## § 32. ПОДПРОГРАММЫ В ЯЗЫКЕ БЕЙСИК

Давайте теперь поговорим об оформлении и вызове подпрограмм.

Как вы помните, вспомогательные алгоритмы начинались с заголовка. Вот пример заголовка:

**Алгоритм "Название".**

**Аргументы:**  $x, y$ .

**Результаты:**  $c, d$ .

В Бейсике заголовки и комментарии писать необязательно. Однако если вы хотите облегчить себе и другим понимание алгоритма (некоторые программисты преследуют, по-видимому, противоположные цели), то пишите комментарии, используя слово REM (REMARK — пояснение). Например, приведенный только что заголовок переписывается так:

**1000 REM "Название". Аргументы  $x, y$ ; результаты  $c, d$ .**

В конце подпрограммы надо обязательно ставить команду RETURN (возвратиться).

Нумеруя строки программы, надо располагать подпрограммы после текста основной программы (вам уже знакомо это правило).

Наша подпрограмма "Название" располагается, начиная

со строки 1000. Вспомните: чтобы исполнитель выполнил этот алгоритм при определенных значениях  $x$  и  $y$ , скажем, при  $x=a$ ,  $y=b$ , использовалась команда вызова:

**Выполнить алгоритм "Название" при  $x=a$ ,  $y=b$ .**

На языке Бейсик этой команде соответствует последовательность из нескольких команд: сначала нужно присвоить переменным  $x$  и  $y$  значения  $a$  и  $b$ , а затем записать команду `GOSUB 1000` (выполнить подпрограмму, начинающуюся со строки 1000).

Проиллюстрируем сказанное на примере задачи на нахождение максимума из трех чисел. В § 14 мы составили алгоритм решения этой задачи с использованием вспомогательного алгоритма "Поиск максимума из двух чисел":

**Алгоритм "Поиск максимума из двух чисел".**

**Аргументы:**  $x, y$ .

**Результат:**  $m$ .

**Если  $x > y$ , то:**

**Присвоить  $m$  значение  $x$ .**

**Иначе:**

**Присвоить  $m$  значение  $y$ .**

**Конец ветвления.**

Основной алгоритм мы записывали так:

**Запросить  $a, b, c$ .**

**Выполнить алгоритм "Поиск максимума из двух чисел" при  $x=a$ ,  $y=b$ .**

**Выполнить алгоритм "Поиск максимума из двух чисел" при  $x=m$ ,  $y=c$ .**

**Сообщить  $m$ .**

Приведем перевод этого алгоритма на Бейсик.

10 INPUT A,B,C

20 X=A:Y=B

30 GOSUB 80

40 X=M:Y=C

50 GOSUB 80

60 PRINT M

70 STOP

80 REM Поиск максимума из двух чисел. Аргументы  $x, y$ ; результат  $m$

90 IF X>Y THEN M=X ELSE M=Y

100 RETURN

## ? Вопросы

1. Как оформляются подпрограммы в языке Бейсик?
2. Как оформляется вызов подпрограмм в языке Бейсик?



## Задания для самостоятельного выполнения

1. Переведите на язык Бейсик алгоритмы для ВЫЧИСЛИТЕЛЯ, составленные вами при решении задач из главы 6.

2. Найдите и исправьте ошибки в программе вычисления максимума из значений двух выражений  $\sin(x^x) + \cos(x^x)$  и  $\sin x + \cos x$ :

```
10 REM Поиск максимума из двух чисел.
15 REM Аргументы x, y; результат m
20 IF X>Y THEN M=X ELSE M=Y
30 RETURN
40 INPUT X
50 GOSUB 10:X=SIN(X^X)+COS(X^X):
  Y=SIN(X)+COS(X)
60 PRINT "МАКСИМАЛЬНОЕ ЗНАЧЕНИЕ "M
```

3. Для упорядочивания трех чисел  $a, b, c$  по возрастанию была написана программа. Злоумышленник стер в ней строку под номером 20. Восстановите стертую строку.

```
10 INPUT A,B,C
20 .....
30 GOSUB 100
40 IF C<=L THEN PRINT C,L,R:STOP
50 IF C>=R THEN PRINT L,R,C ELSE PRINT L,C,R
60 END
100 REM ПОДПРОГРАММА УПОРЯДОЧЕНИЯ ДВУХ ЧИСЕЛ.
105 REM АРГУМЕНТЫ X,Y. РЕЗУЛЬТАТЫ L,R (L<R).
110 IF X<Y THEN L=X:R=Y ELSE L=Y:R=X
120 RETURN
```

4. Напишите программу вычисления по длинам двух сторон треугольника и углу между ними длины третьей стороны и двух других углов.

## § 33. ОРГАНИЗАЦИЯ ДАННЫХ В ЯЗЫКЕ БЕЙСИК

В программах на языке Бейсик можно пользоваться таблицами, прямоугольными и линейными. Прежде чем использовать в программе таблицу, надо дать машине указание, чтобы она заранее отвела в памяти место для нее. Такое указание дается командой DIM (сокращение английского слова DIMENSION — размер). В этой команде указывается имя таблицы, а также — в скобках — число ее строк и столбцов, например:

**10 DIM A(5,6) или 23 DIM B(23)**

В первом из этих примеров определяется прямоугольная таблица А из 5 строк, по 6 чисел в каждой; во втором примере определяется линейная таблица из 23 элементов. Одной командой DIM можно описывать несколько таблиц:

```
10 DIM A(100),B(10,30),C1(25)
```

Действие "Запросить таблицу" переводится в случае линейной таблицы, содержащей N элементов, следующим образом:

```
10 FOR I=1 TO N
20 INPUT A(I)
30 NEXT I
```

Выполняя этот фрагмент, ЭВМ последовательно запросит N чисел, помещая их на соответствующие места в таблице. Для прямоугольной таблицы B, состоящей из N строк и K столбцов, перевод таков:

```
10 FOR I=1 TO N
20 FOR J=1 TO K
30 INPUT B(I,J)
40 NEXT J
50 NEXT I
```

Здесь команды 10 и 50 последовательно перебирают номера строк таблицы. Команды 20, 30, 40 предназначены для ввода элементов очередной строки таблицы.

Действие "Сообщить таблицу" переводится на язык Бейсик аналогично, только вместо слова INPUT надо писать слово PRINT.

Запишем с комментариями программу вычисления суммы всех элементов линейной таблицы, состоящей из 20 чисел:

```
10 REM СУММИРОВАНИЕ ЭЛЕМЕНТОВ ТАБЛИЦЫ
   ИЗ 20 ЧИСЕЛ
20 DIM A(20)
30 REM ВВОДИМ ЭЛЕМЕНТЫ ТАБЛИЦЫ А
40 FOR I=1 TO 20
50 INPUT A(I)
60 NEXT I
70 REM БУКВОЙ S БУДЕТ ОБОЗНАЧЕНА СУММА
80 S=0
90 REM ВЫЧИСЛЯЕМ СУММУ, ДОБАВЛЯЯ К S ПО
   ОЧЕРЕДИ ЭЛЕМЕНТЫ ТАБЛИЦЫ
100 FOR I=1 TO 20
110 S=S+A(I)
120 NEXT I
130 REM ПЕЧАТАЕМ РЕЗУЛЬТАТ
140 PRINT"S="S
```

## ? Вопросы

1. Для чего нужна команда DIM?
2. Как переводятся на язык Бейсик команды "Запросить таблицу" и "Сообщить таблицу"?

## Задания для самостоятельного выполнения

1. Переведите на язык Бейсик алгоритмы для ВЫЧИСЛИТЕЛЯ, составленные вами при решении задач из главы 7.

2. Найдите и исправьте ошибки в программах:

а) Программа вычисления произведения элементов линейной таблицы:

```
10 REM Ввод таблицы в ЭВМ
20 FOR I=1 TO 100
30 INPUT A(I)
40 NEXT I
50 FOR I=1 TO 100
60 S=S*A(I)
70 NEXT I
80 PRINT S
```

б) Программа нахождения сумм элементов в каждой строке прямоугольной таблицы:

```
10 DIM A(20,20)
20 FOR I=1 TO 20
30 PRINT "НАХОЖУ СУММУ ЭЛЕМЕНТОВ В "I"
  СТРОКЕ ТАБЛИЦЫ"
40 S=0
50 FOR J=1 TO 20
60 S=S+A(J,I)
70 NEXT J
80 PRINT "НАШЛА! СУММА РАВНА "S". ПЕРЕХОЖУ
  К СЛЕДУЮЩЕЙ СТРОКЕ"
90 NEXT I
```



## КОНСПЕКТ ГЛАВЫ 8

Для того чтобы ЭВМ "поняла" алгоритм решения задачи и выполнила его, нужно записать этот алгоритм на специальном языке. Такие языки называют **языками программирования** или **алгоритмическими языками**. Они являются одним из основных средств общения человека и компьютера. В настоящее время существуют сотни различных языков программирования. Одни из наиболее распространенных — язык программирования Бейсик.

Любой алгоритм состоит из допустимых действий, организованных с помощью алгоритмических конструкций — ветвлений, циклов, вспомогательных алгоритмов. Поэтому

для того чтобы изучить язык программирования, надо узнать, как в нем называются те или иные допустимые действия и как оформляются алгоритмические конструкции.

Вот основные правила записи программ на Бейсике: в начале каждой строки программы ставится номер; при записи программы можно несколько команд записывать в одной строке, разделяя их двоеточием.

Действию "Присвоить X значение F" соответствует команда

**LET X=F**

В большинстве "диалектов" Бейсика слово LET можно опустить.

Переменную в Бейсике можно обозначать не только буквой, но и буквой с цифрой или двумя буквами. Список основных функций и их обозначений в Бейсике см. в § 20.

Действию "Запросить A, B" на языке Бейсик соответствует команда.

**INPUT A,B**

Если же ввести в ЭВМ строку

**INPUT "Введите три числа": A,B,C**

то на экране сначала появится надпись ВВЕДИТЕ ТРИ ЧИСЛА?, после чего ЭВМ будет ждать ввода трех чисел.

Действию "Сообщить" на Бейсике соответствует команда

**PRINT ...**

Вместо многоточия после слова PRINT может стоять текст, заключенный в кавычки, а также переменные или выражения. Например, по команде

**PRINT "Корни уравнения" X1,X2**

сначала на экране дисплея появится сообщение КОРНИ УРАВНЕНИЯ, а затем — значения переменных X1 и X2.

Команды STOP и END — для остановки выполнения программы.

По команде

**GOTO...**

где вместо многоточия ставится номер строки программы, ЭВМ перейдет к выполнению указанной строки.

Ветвление в неполной форме записывается на языке Бейсик так (здесь Q — условие, а номера команд взяты произвольно):

**10 IF Q THEN ВЕТЬ ВЕТВЛЕНИЯ**

или так:

**120 IF NOT(Q) THEN GOTO 150**

**121 ветвь ветвления**

**150...**

Ветвление в полной форме записывается так:

IF Q THEN первая ветвь ELSE вторая ветвь

или так:

```
530 GOTO 700
531 первая ветвь
600
601 вторая ветвь
700...
```

В языке Бейсик цикл "Пока" в зависимости от реализации на ЭВМ оформляется одним из двух способов:

```
100 WHILE <условие> 100 IF NOT<условие> THEN 210
110 тело цикла      110 тело цикла
200 WEND             200 GOTO 10
210...              210...
```

Цикл "Для каждого" "со счетчиком" 1 оформляется так:

```
100 FOR I=L TO R
110 тело цикла
120 NEXT I,
```

где L — нижняя, а R — верхняя граница изменения I.

При оформлении подпрограмм в Бейсике заголовки и комментарии записываются в начале подпрограммы в строке комментариев, начинающейся со слова REM. Например:

```
1000 REM "НАЗВАНИЕ". АРГУМЕНТЫ X,Y;
      РЕЗУЛЬТАТЫ C, D
```

В конце подпрограммы надо ставить команду RETURN. Нумеруя строки программы, надо располагать подпрограммы после текста основной программы.

Вызов подпрограммы осуществляется так: сначала аргументам подпрограммы присваиваются необходимые значения, а затем записывается команда GOSUB..., где вместо многоточия указывается номер первой строки подпрограммы.

В программах на языке Бейсик можно пользоваться прямоугольными и линейными таблицами. По команде

**DIM ...**

ЭВМ отведет в своей памяти место для таблицы. Вместо многоточия надо указать имя таблицы, а также — в скобках — число ее строк и столбцов. Например,

```
10 DIM A(5,6) или 23 DIM B(23)
```

Одной командой DIM можно описывать несколько таблиц

```
10 DIM A(100), B(10,30), C1(25)
```

## Глава 9

# СИМВОЛЬНЫЕ ПЕРЕМЕННЫЕ

---

Много лет ученым не удавалось расшифровать язык древнего народа майя (Южная Америка). Лишь недавно их усилия увенчались успехом. Результаты расшифровки позволили создать программу, по которой ЭВМ перевела дошедшие до нас письмены этой загадочной цивилизации. Машина помогла расшифровать и другую письменность — киданьскую (Центральная Азия). Она быстро подсчитала, какие символы и в каких сочетаниях встречаются в словах чаще, а какие — реже. В результате в каждом слове киданьского языка удалось выделить корень, суффиксы и окончание. После этого не составило труда понять киданьский язык (он оказался похожим на монгольский).

С помощью ЭВМ можно решать весьма разнообразные задачи обработки текстов: от составления платежных ведомостей до автоматической верстки газет.

Для того чтобы ЭВМ могла обрабатывать тексты, она должна уметь оперировать не только с числами, но и со словами. О том, как это происходит, и рассказывается в данной главе.

### § 34. СЛОВА И ДЕЙСТВИЯ С НИМИ

Давным-давно, в первом классе, вы составляли слова, старательно выводя букву за буквой. Конечно, если просто записывать буквы алфавита одну за другой, может получиться и нечто бессмысленное. Однако для действий, о которых пойдет речь, неважно, имеют слова смысл или нет. Эти действия можно применять даже к словам, записанным произвольными символами, например нотами или "пляшущими человечками". Да и ЭВМ, конечно, недоступен смысл "человеческих" слов. Поэтому словом в информатике принято считать *любую последовательность символов некоторого алфавита*. **Алфавитом** может служить *любое множество символов*.

Если, скажем, алфавит состоит из цифр 0,1,2,3,4,5,6,7,8,9, то натуральные числа — слова в этом алфавите. Если же добавить в этот алфавит запятую, то появится

*Слово — не воробей,  
а последовательность  
символов алфавита*

возможность записывать и неотрицательные десятичные дроби. Алфавит азбуки Морзе состоит из трех символов: точки, тире и пробела. Обычно, правда, упоминают только два символа —

точку и тире. А ведь без пробела нельзя было бы в передаваемом сообщении отделить одну букву от другой.

Пробел вообще замечательный символ! Никак не обозначенный, он тем не менее присутствует почти в каждой фразе. Посмотрите, сколько пробелов на этой странице! А поля? Они сплошь состоят из пробелов. Без этого скромного труженика полей текст слипсябыниегобылобытрудночитатьипонимать. Если в русский алфавит добавить пробел и все знаки препинания, то любая фраза станет словом (по нашему определению).

Как видите, пробел ничем не хуже других символов. Поэтому можно рассматривать слово, состоящее, скажем, из одного пробела. Впрочем, это еще не самое удивительное слово. Часто бывает необходимым слово, в котором вообще нет символов (вы убедитесь в этом, изучив § 36). Такое слово называется **пустым**.

Итак, повторим, слово (в информатике) — это произвольная последовательность символов (в частности последовательность, не содержащая ни одного символа). *Число символов в слове назовем его длиной* (длина пустого слова равна 0).

Теперь, когда вы освоились с понятием слова, расскажем, какие действия можно совершать со словами.

Вспомните, как на уроках русского языка вы анализировали слова, разбирая их по составу, — выделяли корни и приставки, суффиксы и окончания. Как видите, одна из основных операций, необходимых для анализа слов, — **выделение в слове его части**. Часть слова, конечно, тоже является словом. Можно по-разному указывать, какая часть слова выделяется. Мы договоримся задавать номер первого символа выделяемой части и количество символов в ней. Приведем примеры: слово РОК — часть слова КРОКОДИЛ, начинающаяся со второй буквы и имеющая длину 3. А слова РОД и ДОК не являются его частями. Конечно, операция выделения части слова не всегда выполнима (например, в слове СТОЛ нельзя выделить часть слова, имеющую длину 5).

Слова можно не только разбирать на части, но и собирать из других слов, как поезд из вагонов. Для этого служит операция **соединения слов**. Определение соединения слов очень простое: *соединить два слова — это значит к первому слову справа приписать второе*. Соединение слов

будем обозначать знаком + (как сложение чисел). Например:

КОМ+ПОТ=КОМПОТ,  
БОР+ОДА=БОРОДА,  
ГРАД+УС=ГРАДУС,  
BON+JOUR=BONJOUR,  
BUTTER+BROT=BUTTERBROT  
BUTTER+FLY=BUTTERFLY  
FOOT+BALL=FOOTBALL

(придумайте еще несколько похожих примеров).

Сравним свойства соединения слов и сложения чисел. Сразу ясно, что, в отличие от сложения чисел, для соединения слов переместительный закон неверен — результат, как правило, зависит от порядка, в котором соединяются слова. Например:

ПОТ+КОМ≠КОМ+ПОТ.

В то же время сочетательный закон верен:

(ПАР+О)+ХОД=ПАР+(О+ХОД)=ПАР+О+ХОД=ПАРОХОД.

Вообще, соединяя несколько слов, скобки можно не писать.

А что получится, если некоторое слово соединить с пустым словом? Каждому ясно: от соединения с пустым словом ничего не меняется. Среди слов пустое слово играет ту же роль, что число 0 среди чисел.

*Пустое слово нулю  
подобно: прибавляй,  
не прибавляй,  
а результат  
все тот же*

Используя выделение части слова и соединение слов, можно решать самые разнообразные задачи. В этих задачах слова выступают как данные (точно так же, как числа в вычислительных задачах). Как вы знаете, данные хороши тогда, когда они хорошо организованы, упорядочены. Люди давно придумали замечательный способ упорядочения слов — **алфавитный**. Именно так упорядочены слова в словарях. Из двух слов договоримся считать большим то, которое в словаре стоит дальше:

СЛОН > МОСЬКА,  
ЖИРАФ > ЖИР.

Но как быть, если словаря под рукой нет или сравниваемые слова еще не попали в словарь? Например, какое из слов больше: РОКЕР или БРЕЙ-КЕР? Ясно, что РОКЕР больше. Ведь буква Р в алфавите стоит дальше, чем буква Б. А что, если первые буквы слов совпали (например, РОК и РЫК)? Тогда надо смотреть на вторые буквы. А что, если и вторые совпали?..Ну тогда... Закон-

*— Что же больше,  
"СЛОН" или "КИТ"?  
— Загляните  
в алфавит!*

чите формулировку правила сравнения слов самостоятельно.

Итак, сравнивать слова можно и без словаря. Достаточно знать порядок букв (символов) в соответствующем алфавите.

## ? Вопросы

1. Что такое слово?
  2. Что такое алфавит?
  3. Какое слово называется пустым?
  4. Что такое длина слова? Какова длина пустого слова?
  5. В чем состоит операция выделения части слова?
- Всегда ли эта операция выполнима?
6. Что такое соединение двух слов?
  7. Выполняется ли для соединения слов:
    - а) переместительный закон;
    - б) сочетательный закон?
  8. Какое слово при соединении слов играет роль нуля?
  9. Как сравнить два слова между собой?

## III Задания для самостоятельного выполнения

1. а) Придумайте пример двух слов, результат соединения которых не зависит от их порядка.

б)\* Определите, какими должны быть два слова, для которых результат соединения не зависит от их порядка.

2. Даны слова: ИНФОРМАТИКА, ЭЛЕКТРИФИКАЦИЯ. Какие получатся слова, если к данным словам применить операцию выделения части, имеющей длину 5, начиная с третьей буквы?

3. Какой длины должно быть слово, чтобы в нем можно было выделить часть, имеющую длину  $n$ , начиная с  $k$ -го символа?

4. Злоумышленник стер по одному "слагаемому" в каждом из следующих равенств:

- а) ...+КОЗА=СТРЕКОЗА;
- б)  $2+3+\dots=235$ ;
- в) НЕ+...+ХОЧУ = НЕ ХОЧУ;
- г)  $\%N_2 + \dots + () = \%N_2 \langle \rangle ()$ .

Восстановите пропавшие "слагаемые".

5. С помощью операций выделения части и соединения из слова ЖЕЛЕЗНОДОРОЖНИК можно сделать слово ДОЗОР. Для этого нужно соединить следующие части слова ЖЕЛЕЗНОДОРОЖНИК: часть длины 2, начинающуюся с 8-й буквы, часть длины 1, начинающуюся с 5-й буквы, и часть длины 2, начинающуюся с 9-й буквы. С помощью

тех же операций составьте из слова ЖЕЛЕЗНОДОРОЖНИК слова ДОНОР, ЖЕЗЛ, КОЛЕНО, КРОКОДИЛ.

6. Дано слово длины 5. Используя действия выделения части и соединения, составьте слово, записанное теми же символами, но в обратном порядке. Например, из слова 12345 должно получиться слово 54321.

7. Алфавит племени Мумбо-Юмбо состоит из трех букв: Ъ, Ь, Ы, расположенных в указанном порядке. Упорядочите следующие слова по возрастанию (в скобках указан перевод этих слов):

ЬЫ (вкусно),  
ЪЪЫЫЫЬЬЬ (мясо),  
ЪЪЪЪ (вождь),  
ЫЫЫЫЫ (табу),  
ЫЬЬЬЬ (информатика).

8. а) Найдите слово русского языка, которое больше, чем слово ПАР, и меньше, чем слово ПАРУС.

б) Два слова называются соседними, если не существует слов, больших одного из них и меньших другого. Приведите пример двух соседних слов русского языка.

в) Докажите, что два различных слова одной длины не могут быть соседними.

9\*. Если натуральные числа записывать обычным образом в алфавите, состоящем из цифр, то сумма чисел не совпадает с соединением соответствующих слов (приведите пример). Какой алфавит надо выбрать и как надо записывать числа, чтобы сумма чисел была равна соединению соответствующих слов?

## § 35. СИМВОЛЬНЫЕ ПЕРЕМЕННЫЕ И ОПЕРАЦИИ НАД НИМИ В ЯЗЫКЕ БЕЙСИК

В этом параграфе мы расскажем о тех возможностях языка Бейсик, которые позволяют использовать ЭВМ для работы со словами. Начнем с алфавита, из символов которого составляются слова. Этот алфавит изображен на клавиатуре ЭВМ. Для каждой ЭВМ он свой, но в нем всегда есть русские и латинские буквы, цифры, знаки препинания, пробел, а также символы операций (+, -, \*, /, ^).

Для работы со словами часто необходимо знать их длину. Для обозначения длины в языке Бейсик используется слово LEN (length — длина). Например, число

LEN("СВЕТИТЬ — И НИКАКИХ ГВОЗДЕЙ!")

равно 28 (проверьте!). Обратите внимание: слово, длина которого вычисляется, заключено в кавычки. Кавычки, естественно, не учитываются при вычислении длины. Для

чего же тогда они нужны? Судите сами: должна же ЭВМ отличать, скажем, слово 123 от числа 123, букву I от переменной I, слово A(2,3) от элемента A(2,3) таблицы A. Обнаружив кавычки, ЭВМ сразу же понимает, что она имеет дело со словом. А что произойдет, если между кавычками не записать ни одного символа, т. е. чему равно LEN("")? Как вы помните, слово без символов называется пустым, а его длина равна нулю. Значит, LEN("")=0.

Каждому ясно, что в вычислительных алгоритмах трудно обойтись без переменных, принимающих различные числовые значения. Мы советовали вам (в лабораторной работе 6) представлять переменную как ящик, в котором можно хранить число. Когда в этот ящик кладут новое число, старое бесследно исчезает. На каждом ящике — табличка, на которой написано имя переменной. Имена переменных можно использовать в записи различных алгебраических выражений. При вычислении значений этих выражений имена заменяются числами, взятыми из соответствующих ящиков.

Для работы со словами, конечно, тоже нужны переменные. Их называют **символьными переменными**. Это такие же ящики, только хранятся в них не числа, а слова — значения символьных переменных. В языке Бейсик символьную переменную можно обозначить, как и числовую, латинской буквой или буквой и цифрой. Только чтобы не путать имена числовых и символьных переменных, в конце имени символьной переменной ставится символ \$. Вот примеры имен символьных переменных:

A\$, B\$, Z1\$.

Изменить значение символьной переменной проще всего, присвоив ей новое значение. Например, чтобы переменная A\$ получила значение ЗНАНИЕ — СИЛА, надо применить команду присваивания

**A\$ = "ЗНАНИЕ — СИЛА".**

Обратите внимание, что в команде присваивания слово заключается в кавычки.

С помощью команды присваивания можно также сделать копию значения символьной переменной, присвоив его другой символьной переменной:

**C\$=A\$.**

После выполнения этой команды одно и то же слово будет "храниться" в двух символьных переменных — A\$ и C\$.

Конечно, возможности команды присваивания гораздо шире: с ее помощью можно не только менять значение переменной, но и производить действия со словами. Вы, разумеется, догадываетесь, о каких действиях идет речь: о соединении слов и выделении части слова.

Соединение слов в языке Бейсик обозначается знаком + (предвидя это, мы начали использовать + для соединения слов уже в предыдущем параграфе). Например, по команде

**D\$="СТЕРЕО"+"ПРИЕМНИК"**

значением переменной D\$ станет слово СТЕРЕОПРИЕМНИК. Часть слова, начинающаяся с А-го символа и имеющая длину b, обозначается в языке Бейсик так:

**MID\$(...,a,b).**

Здесь вместо многоточия записывается исходное слово (разумеется, в кавычках). Например, по команде

**ES = MID\$("ГИППОПОТАМ",4,3)**

переменной ES будет присвоено значение ПОП.

Комбинируя операции + и MID\$, можно проделывать со словами очень сложные преобразования. Так, получение слова ДОЗОР из слова ЖЕЛЕЗНОДОРОЖНИК (см. задачу 5 к § 34) можно записать следующим образом:

**PS = MID\$("ЖЕЛЕЗНОДОРОЖНИК",8,2)+  
MID\$("ЖЕЛЕЗНОДОРОЖНИК",5,1)+  
MID\$("ЖЕЛЕЗНОДОРОЖНИК",9,2)**

Вы прекрасно знаете, что числовые операции используются для работы не только с конкретными числами, но и с переменными, из которых конструируют алгебраические выражения. Операции MID\$ и + также можно использовать для "сборки" слов из значений символьных переменных. Для этого в операциях указывают имена переменных, значения которых используются при "сборке". Например, выполнение предыдущей команды присваивания можно заменить последовательным выполнением двух команд:

**QS = "ЖЕЛЕЗНОДОРОЖНИК"  
PS = MID\$(QS,8,2)+MID\$(QS,5,1)+MID\$(QS,9,2)**

Как видите, запись стала более элегантной.

Итак, вы узнали, как ЭВМ перерабатывает символьные переменные. Для общения с ЭВМ этого явно недостаточно. Надо еще знать, как ЭВМ запрашивает исходные данные и сообщает результаты своей работы. Для этого предназначены знакомые вам команды INPUT и PRINT.

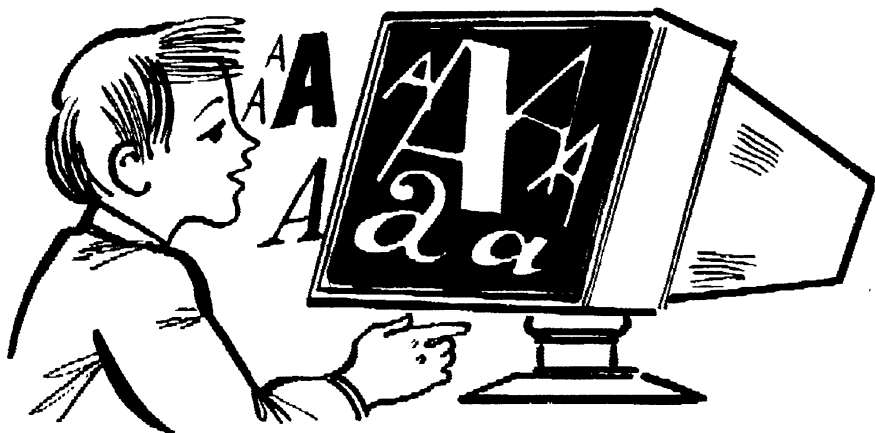
По команде

**INPUT A\$,B\$,C\$**

ЭВМ запросит три слова и обозначит их соответственно A\$,B\$ и C\$. Вводя слова в ЭВМ, надо заключать их в кавычки и разделять запятыми. При этом ЭВМ ведет себя так же, как и при вводе чисел: печатает знак вопроса и ждет до тех пор, пока в нее не будут введены три слова.

По команде

**PRINT A\$**



ЭВМ напечатает на экране значение переменной  $A\$$ . Кроме того, как вы помните, с помощью команды `PRINT` можно напечатать слово (сообщение), заключив его в кавычки.

Теперь, когда вы знаете основные команды обработки слов, можно приступить к решению задач. Есть два типа задач, наиболее часто возникающих при работе с текстами:

а) подсчитать, сколько раз данное слово встречается в тексте;

б) заменить в тексте одно слово другим.

Такие задачи возникают, скажем, когда нужно зашифровать или расшифровать сообщение (этим мы с вами займемся в следующем параграфе).

Разберем по одной задаче каждого типа.

**Задача 1.** Определить, сколько раз в данном слове встречается буква  $A$ .

Обозначим исходное слово через  $W\$$ , а результат (количество букв  $A$  в  $W\$$ ) через  $S$ . Как найти  $S$  — понять несложно.

Сначала ЭВМ должна запросить слово  $W\$$  и присвоить  $S$  начальное значение 0. Затем, просматривая слово  $W\$$  слева направо, нужно добавлять к  $S$  единицу всякий раз, когда встретится буква  $A$ . По окончании просмотра ЭВМ должна сообщить найденное значение  $S$ . Вот блок-схема алгоритма решения этой задачи (рис. 54).

Как видите, нам потребовались циклы и развилки. В них проверяются некие условия. В качестве условий в языке Бейсик допускаются соотношения не только между числами, но и между словами. А в числовых выражениях можно использовать функцию `LEN`. Приведем программу решения нашей задачи:

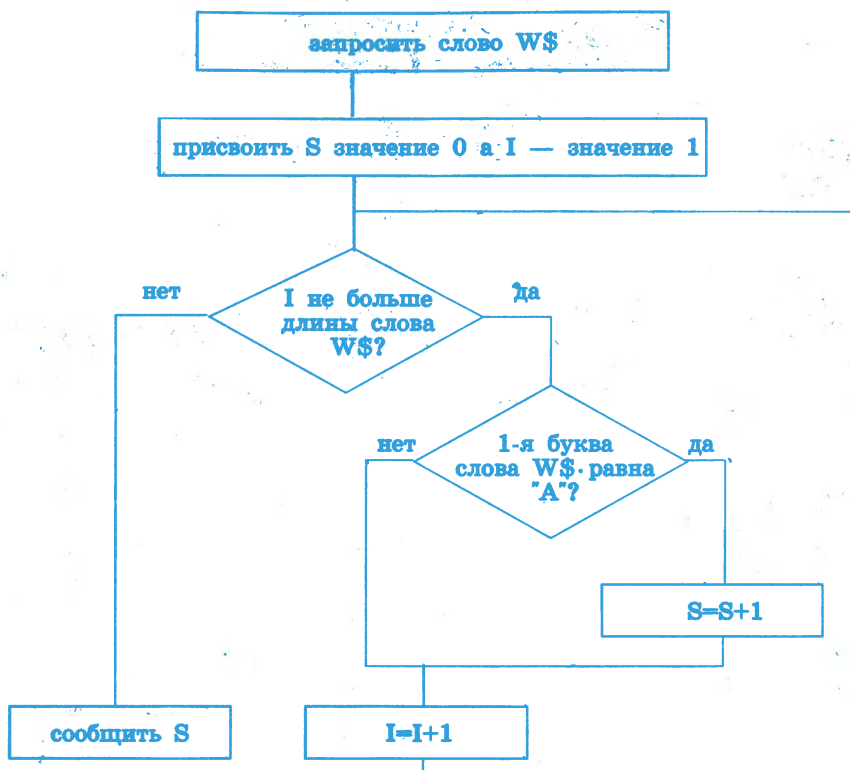


Рис. 54

```

10 INPUT W$
20 S=0
30 FOR I=1 TO LEN(W$)
40 IF MID$(W$,I,1)="A" THEN S=S+1
50 NEXT I
60 PRINT "В СЛОВЕ " W$
70 PRINT S " БУКВ А"

```

**Задача 2.** В данном слове заменить всюду букву А на букву Я

Хотя задача 2 на первый взгляд отличается от задачи 1, алгоритмы их решения очень похожи.

Такая ситуация не редкость. Информатика, как и любая наука, позволяет обнаруживать общее в решении, казалось бы, совершенно различных задач. Вы неоднократно встречались с этим в предыдущих главах (сравните, к примеру, алгоритмы нахождения самого влажного месяца и определения оптимального плана для участка завода в §25 и 26).



Как и в предыдущей задаче, обозначим исходное слово через  $W\$$ . ЭВМ, конечно, сначала должна запросить его. Затем, просматривая  $W\$$  слева направо, она должна каждый раз, встретив букву А, заменить ее буквой Я.

Единственная трудность — с помощью операций + и MID\$ осуществить замену в слове  $W\$$  одной буквы на другую. Пусть буква А в слове  $W\$$  стоит на  $i$ -м месте и ее надо заменить на букву Я. Слово  $W\$$  разобьем на три куса: первый кусок — часть слова  $W\$$  от начала до буквы А (его длина  $i-1$ ), второй кусок — сама буква А, третий кусок — все остальное (его длина равна  $LEN(W\$)-i$ ). Замена коснется только второго куска, а первый и третий куски останутся нетронутыми. Поэтому результат такой замены можно записать так:

$$MID\$(W\$,1,i-1)+\text{Я}+MID\$(W\$,i+1,LEN(W\$)-i).$$

Приведем программу решения задачи 2:

```

10 INPUT W$
20 FOR I=1 TO LEN(W$)
30 IF MID$(W$,I,1)="A" THEN
   W$=MID$(W$,1,I-1)+"Я"+MID$(W$,I+1,LEN(W$)-I)
40 NEXT I
50 PRINT "ВОТ НОВОЕ СЛОВО: " W$

```

### ? Вопросы

1. Как обозначается в языке Бейсик:
  - а) длина слова;
  - б) символьная переменная;
  - в) часть слова;
  - г) соединение слов?
2. Что такое символьная переменная?

3. Какие преобразования слов позволяет осуществлять команда присваивания?

4. С помощью каких команд языка Бейсик ЭВМ запрашивает и сообщает значения символьных переменных?

5. Проверку каких соотношений между словами позволяет осуществлять язык Бейсик?



### Задания для самостоятельного выполнения

1. По какой команде языка Бейсик к значению переменной A\$ будет дописано справа значение переменной B\$? А слева?

2. Чему будут равны переменные A\$ и B\$ после выполнения следующих программ?

а) 10 A\$="РАДИО"

20 B\$="ВЕЩАНИЕ"

30 B\$=A\$+B\$

40 A\$=MID\$(B\$,3,3)+MID\$(A\$,3,1)

50 B\$=MID\$(B\$,1,1)+MID\$(B\$,5,1)+MID\$(B\$,8,2)

60 STOP

б) 10 A\$="КРИМИНАЛИСТИКА"

20 IF MID\$(A\$,3,2)>MID\$(A\$,5,2) THEN 60

30 B\$=MID\$(A\$,8,6)

40 A\$=MID\$(A\$,1,3)+MID\$(A\$,12,4)

50 GOTO 80

60 A\$=MID\$(A\$,4,2)+MID\$(A\$,10,5)

70 B\$="Г"+MID\$(A\$,2,1)+MID\$(A\$,1,1)+"Н"+  
MID\$(A\$,7,1)+MID\$(A\$,3,5)

80 STOP

3. Составьте программу для определения:

а) содержится ли в данном слове данная буква;

б) имеется ли в данном слове часть, равная заданному слову;

в) сколько раз встречается в данном слове часть, равная заданному слову.

4. Составьте программу для замены в данном слове всюду:

а) одной буквы на другую букву (ЭВМ должна запросить обе буквы);

б) части ДОМ на часть РОД;

в) части ДЕТЕКТИВ на часть РОМАН. Например, текст  
ДЕТЕКТИВ БЕЗ ПОГОНИ — ЭТО НЕ ДЕТЕКТИВ

должен стать таким:

РОМАН БЕЗ ПОГОНИ — ЭТО НЕ РОМАН

А из текста

НЕ МОГУ ЗАСНУТЬ БЕЗ ДЕТЕКТИВА

должен получиться текст

## НЕ МОГУ ЗАСНУТЬ БЕЗ РОМАНА

г)\* части РОМАН на часть ДЕТЕКТИВ;

д)\* части, равной одному слову, на часть, равную другому слову (ЭВМ должна запросить оба слова).

Почему в задачах 3,г и 3,д нельзя воспользоваться циклом "для каждого" от 1 до LEN(A\$)?

5. Составьте программу, вычеркивающую все пробелы из данного слова.

6\*. Даны два слова. Составьте программу, позволяющую определить, можно ли из букв, входящих в первое слово, составить второе слово. Буквы можно переставлять, но нельзя использовать букву большее число раз, чем она встречается в первом слове. Например, из слова ИНФОРМАТИКА можно составить слово РАКИТА и нельзя составить слово МОТОР.

7. В задаче 7 к предыдущему параграфу вы познакомились с алфавитом племени Мумбо-Юмбо.

а) Составьте программу, которая позволит определить, какое из двух слов мумбо-юмбовского языка больше.

б) Составьте программу упорядочения трех слов мумбо-юмбовского языка. (У к а з а н и е: воспользуйтесь программой, составленной при решении задачи 7,а, как подпрограммой.)

8. Составьте программу для определения стоимости телеграммы по ее тексту (напомним, что стоимость одного слова в телеграмме 5 р. и, кроме того, за каждую телеграмму взимается комиссионный сбор — 10 р).

9. Составьте программу для определения:

а) является данная буква гласной, согласной или одной из букв й,ъ,ь;

б) обозначает ли данная согласная буква звонкий звук или глухой. (У к а з а н и е: используйте символьные переменные A\$="АЯОЕЭИЫУЮ", Z\$="БВГДЖЗЛМНР", G\$="КПСТФХЦЧЩ", W\$="ЙЪЬ".)

10. Составьте программу для проверки правописания приставок *из-* и *ис-* в словах. (У к а з а н и е: используйте решение предыдущей задачи.)

11. а) Выполняя лабораторную работу к § 4, вы с помощью текстового редактора преобразовывали телеграфный текст в обычный, заменяя сокращения "тчк", "зпт", "впр" и "вск" соответствующими знаками препинания. Пора эту работу полностью поручить ЭВМ! Составьте соответствующую компьютерную модель (ориентированную на использование языка Бейсик) и программу.

б) Составьте компьютерную модель и программу преобразования обычного текста в телеграфный.

12\*. Составьте компьютерную модель и программу решения следующей задачи:

**Задача.** Школьник и злоумышленник написали сочинения на одну и ту же тему. Определить, списывал ли злоумышленник у школьника. (Указание: используйте методы, описанные в § 28.)

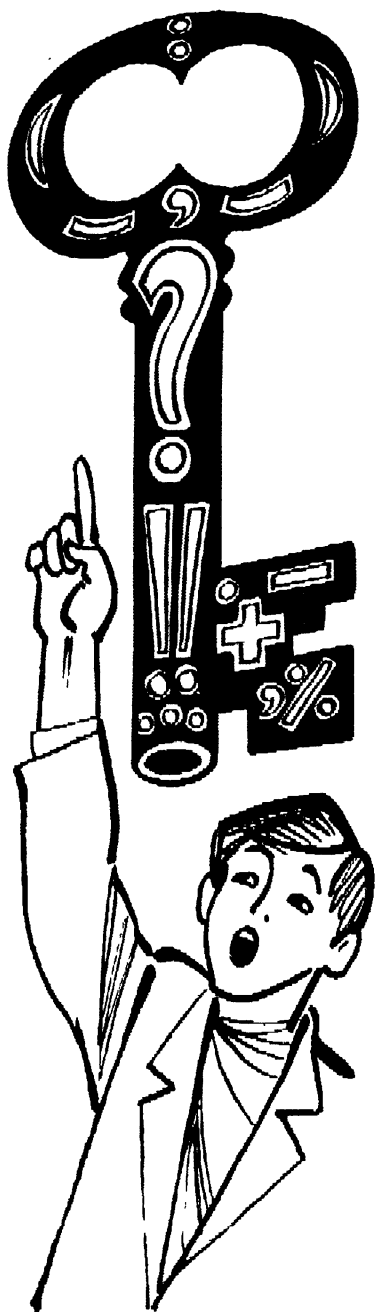
### § 36. ИСПОЛЬЗОВАНИЕ ЭВМ ДЛЯ ШИФРОВКИ И ДЕШИФРОВКИ СООБЩЕНИЙ

...умею ли я находить с помощью электронно-вычислительной машины значения функций, а также объемы многогранников?!

Фраза, которую вы только что прочитали, замечательна. Возможно, ей не хватает поэтичности и глубины содержания. Зато в ней встречаются все буквы русского алфавита и, кроме того, запятая, точка, тире, пробел, восклицательный и вопросительный знаки. Именно такие фразы используются для шифровки и дешифровки сообщений. Профессионалы называют эти фразы ключевыми.

Все делается просто. Некто, скажем, Юстас, пишет секретный текст, скажем, Алексу. После этого шифровальщик заменит каждый символ текста на номер этого символа в ключевой фразе. Получившийся набор чисел передается шифровальщику Алекса, который с помощью той же ключевой фразы дешифрует сообщение.

Как видите, работа шифровальщиков — чисто автоматическая. Значит, ее можно поручить



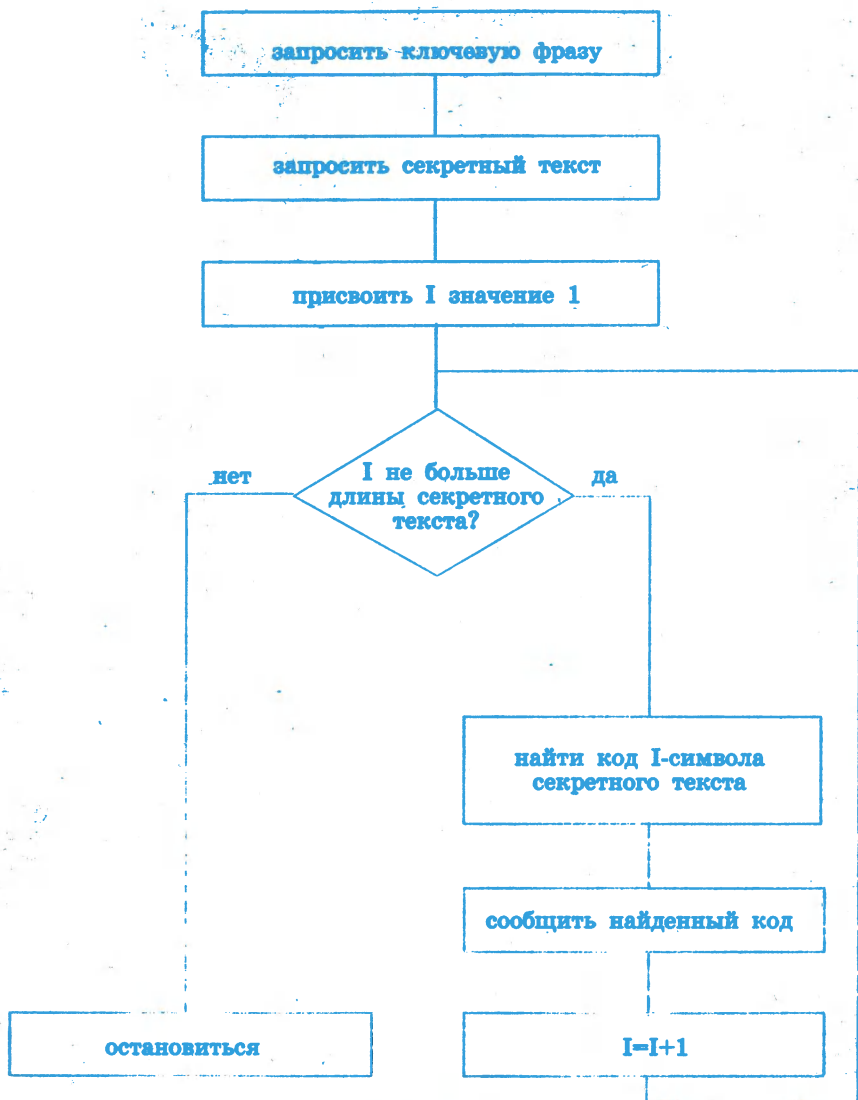


Рис. 55

ЭВМ. К тому же, как известно, профессионалы стремятся уменьшить количество людей, знающих ключевую фразу. Итак,

**Задача.** Составить программы шифровки и дешифровки текстов с помощью ключевой фразы.

Сначала составим программу шифровки текстов. Машина должна запросить ключевую фразу и секретный

текст. Затем, просматривая секретный текст, ЭВМ должна сообщать код каждой буквы (т. е. номер этой буквы в ключевой фразе). На рисунке 55 показана соответствующая блок-схема.

В детализации нуждается лишь блок поиска кода  $i$ -го символа. Будем предполагать, что у нас уже есть подпрограмма "КОДИРОВАНИЕ БУКВЫ", реализующая этот блок и начинающаяся со строки 500. Аргументами ее служат символ P\$, код которого требуется найти, и ключевая фраза A\$. Результат — код C символа P\$.

Запишем основную программу:

```
5 PRINT "ПРОГРАММА ШИФРОВКИ ТЕКСТОВ"  
10 PRINT "ВВЕДИТЕ КЛЮЧЕВУЮ ФРАЗУ"  
20 INPUT A$  
30 PRINT "ВВЕДИТЕ СЕКРЕТНЫЙ ТЕКСТ"  
40 INPUT B$  
50 FOR I=1 TO LEN(B$)  
60 P$=MID$(B$,I,1)  
70 GOSUB 500  
80 PRINT C  
90 NEXT I  
100 PRINT "ШИФРОВКА ЗАКОНЧЕНА"  
110 STOP
```

Составим теперь подпрограмму "КОДИРОВАНИЕ БУКВЫ", предназначенную для определения номера позиции, которую занимает символ P\$. Вдумчивый ученик должен спросить: а как быть, если символ встречается в ключевой фразе несколько раз? Например, в нашей ключевой фразе буква "а" стоит на 15, 60, 68, 83, 86 и 105-й позициях. Получается, что один символ может быть закодирован несколькими числами. Договоримся для простоты в качестве кода символа брать наименьшее из этих чисел (так, буква "а" имеет код 15).

ЭВМ в роли шифровальщика должна просматривать ключевую фразу слева направо, разыскивая символ — значение переменной P\$. Найдя его, она должна присвоить C номер соответствующей позиции и закончить просмотр.

При поиске кода символа естественно воспользоваться циклом "Для каждого":

```
500 REM ПОДПРОГРАММА КОДИРОВАНИЯ БУКВЫ.  
    АРГУМЕНТЫ: СИМВОЛ P$, КЛЮЧЕВАЯ ФРАЗА A$;  
    РЕЗУЛЬТАТ: C (НОМЕР СИМВОЛА P$ В СЛОВЕ A$).  
510 FOR J=1 TO LEN(A$)  
520 IF MID$(A$,J,1)<>P$ THEN 540  
530 C=J  
540 NEXT J  
550 RETURN
```

А теперь проверьте, какой код получит буква "а" после работы этой подпрограммы... Увы, получилось 105, а вовсе не 15. В чем же дело? Дело в том, что ЭВМ, вопреки нашему соглашению, продолжает просматривать ключевую фразу и после того, как нашла букву "а". Поэтому переменной С присваивается номер не первой, а последней позиции буквы "а". Чтобы этого не происходило, надо прекращать выполнение цикла, как только нужный символ будет найден. Можно поступить так: сразу после первого присваивания переменной С значения J (строка 530) сделать значение счетчика цикла равным верхней границе цикла, т. е. числу LEN(A\$). Исправим нашу подпрограмму:

```
500 REM ПОДПРОГРАММА КОДИРОВАНИЯ БУКВЫ.  
    АРГУМЕНТЫ: СИМВОЛ P$, КЛЮЧЕВАЯ ФРАЗА A$;  
    РЕЗУЛЬТАТ: С (НОМЕР СИМВОЛА P$ В СЛОВЕ A$)  
510 FOR J=1 TO LEN(A$)  
520 IF MID$(A$,J,1)<>P$ THEN 540  
530 C=J  
535 J=LEN(A$)  
540 NEXT J  
550 RETURN
```

Займемся теперь дешифровкой. ЭВМ опять должна запросить ключевую фразу. Затем, запрашивая последовательно числа, кодирующие символы зашифрованного сообщения, она составит из соответствующих символов исходный текст. Например, расшифруем сообщение

. 61 10 75 38 17 44 36 15

и запишем его в символьную переменную B\$. Сначала переменной B\$ присвоим пустое слово. Затем берем 61-й символ ключевой фразы (это буква "ш") и добавляем его к B\$. После этого допишем к B\$ последовательно 10, 75, 38, 17, 44, 36 и 15-й символы ключевой фразы. Получим слово "шифровка".

Кажется, все ясно, и можно приступить к составлению программы. Но это только кажется. Ведь в разобранный пример мы видели шифрованное сообщение целиком и легко могли определить, когда кончать расшифровку. Трудность состоит в том, что ЭВМ будет воспринимать шифровку число за числом. Она не в состоянии определить, ждать ей от нас еще коды символов или сообщать расшифрованную фразу. Значит, надо договориться о том, как машина будет обнаруживать окончание шифрованного сообщения. Закончив шифровку, введем число 0. Договоримся, что ЭВМ будет воспринимать его как сигнал об окончании сообщения.

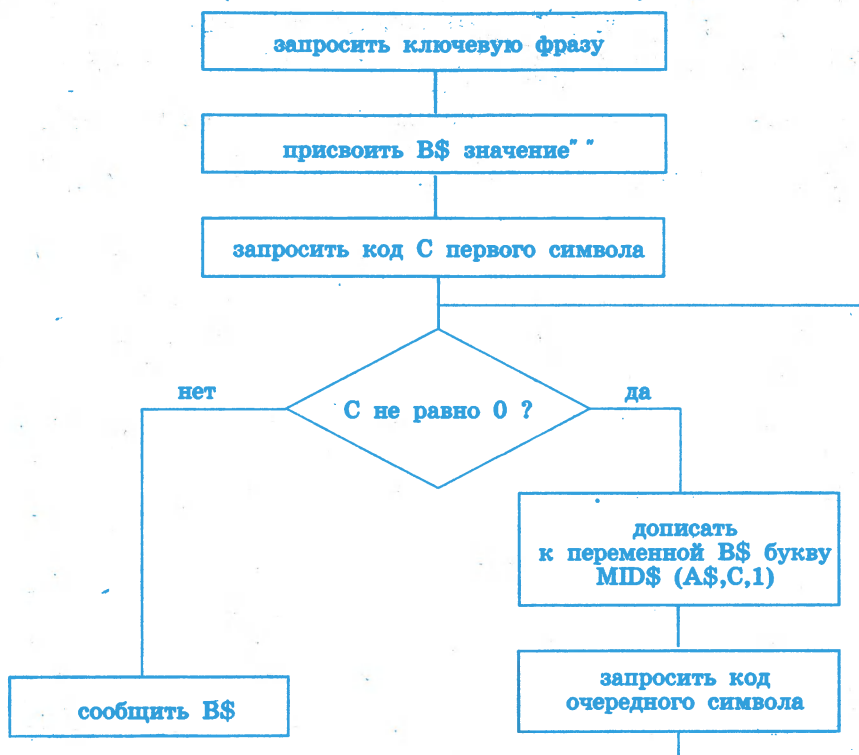


Рис. 56

Как и раньше, будем считать, что ключевая фраза "хранится" в переменной А\$, а дешифрованное сообщение "накапливается" в переменной В\$ (вспомните, что переменные — это своего рода ящики для хранения данных).

Итак, ЭВМ должна сначала запросить ключевую фразу. Затем она должна запрашивать по порядку числа шифровки, каждый раз проверяя, не поступило ли число 0. Если запрошенное число окажется не равным нулю, ЭВМ должна найти соответствующий символ и приписать его к символьной переменной В\$. Получив число 0, ЭВМ должна сообщить окончательное значение В\$, которое и будет результатом дешифровки.

Блок-схема дешифровки изображена на рисунке 56. Разберитесь в ней и напишите самостоятельно программу.



### Задания для самостоятельного выполнения

1.а) Придумайте свою ключевую фразу.

б) Зашифруйте с помощью своей ключевой фразы следующее сообщение:

юстас — алексу. дорогой алекс! я нахожусь под колпаком.

искренне ваш юстас.

2. Один из методов шифровки сообщений состоит в том, что после каждого символа вставляется некоторая буква (каждый раз, вообще говоря, разная). Например, из фразы

я изучаю информатику  
может получиться фраза

яэ ииызхукччаяую риннпфыовртмвактхидкфуф.

а) Расшифруйте зашифрованную таким способом фразу:  
як аллюрболлюр ухтэйвоир омлеттсакля!!

б) Составьте программу шифровки таким способом произвольного текста, используя для вставок одну и ту же букву.

в)\* Составьте программу шифровки тем же способом, выбирая для каждой вставки букву случайным образом.

г) Составьте программу дешифровки сообщений.

3. Иногда хочется что-то сказать, но так, чтобы никто не понял. Лучше всего в таких случаях пользоваться специальным шифром: произносить слова задом наперед. Например, слово "пылесос" произносится так: "соселып", а фраза "Я ничего не смыслю в кулинарии" произносится так: "Я огечин ен юлсымс в ииранилук". Составьте программы шифровки таким методом:

а) слов русского языка;

б)\* предложений русского языка.

Убедитесь в том, что эти же программы годятся и для дешифровки.

4. Очень легко переводить с русского языка на тарабарский: надо каждую глухую согласную заменять соответствующей звонкой согласной и наоборот. Например, фраза

Сегодня — школьник, завтра — академик  
на тарабарском звучит так:

Зекотня — жгольниг, сафдра — агатемиг.

Составьте программу перевода с русского языка на тарабарский.

У к а з а н и е: воспользуйтесь решением задачи 7 к предыдущему параграфу.

5\*. Предлагается следующий метод шифровки сообщений на русском языке (метод сдвига). Выбирается целое число  $k$  от 1 до 32. Буквы русского алфавита записываются вдоль окружности по часовой стрелке (так, что буква

"а" соседствует с буквами "б" и "я"). Затем в сообщении каждая буква заменяется на букву, отстоящую от нее на  $k$  букв по часовой стрелке. Знаки препинания и пробел не меняются. Например, если  $k=3$ , то "а" заменится на "г", "ц" — на "ь", "ю" — на "б", а фраза "Учение — свет, неученье — тьма" превратится в "Цьирэи — феих, рицьирэи — хэпг". Составить программу шифровки и дешифровки методом сдвига.

**У к а з а н и е:** воспользуйтесь тем, что при сдвиге буква с номером  $n$  заменяется на букву с номером  $m$ , вычисляемым по формулам:

$$\begin{aligned} n+k, & \text{ если } n+k \leq 32; \\ m = & \\ n+k-32, & \text{ если } n+k > 32. \end{aligned}$$

6. В памяти ЭВМ "Ямаха" каждый символ кодируется восемью нулями и единицами. Приведем коды некоторых символов:

А — 11100001,  
Б — 11100010,  
Ц — 11100011,  
Д — 11100100,  
Е — 11100101.

Составьте программу кодирования слов, составленных из перечисленных символов.



## Лабораторная работа 19

### ЗАДАЧИ ШИФРОВКИ И ДЕШИФРОВКИ

Вы знаете, что на уроках нельзя посылать друг другу записки. А иногда очень нужно что-то кому-то сообщить, причем по возможности скрытно. Сегодня вам будет предоставлена такая возможность. Решая задачу 1, каждый из вас придумал свою ключевую фразу. Посоветуйтесь между собой и выберите из этих фраз три самые удачные. Затем каждый должен написать по три записки на любую тему, зашифровав их с помощью выбранных ключевых фраз. Листочки с зашифрованными сообщениями передайте учителю. Перемешав их, он выдаст каждому по три шифровки (проследите, чтобы вам не досталась ваша собственная записка). Теперь вы должны расшифровать полученные сообщения. Это не просто, тем более что вы не знаете, с помощью какой ключевой фразы зашифрована каждая из записок. Разумеется, для шифровки и дешифровки надо применять соответствующие программы. Действуйте!



**Слово** (в информатике) — это произвольная последовательность символов некоторого **алфавита**. Алфавитом может служить любое множество символов. Одним из равноправных символов алфавита может быть пробел. Последовательность, не содержащая ни одного символа, называется **пустым словом**. Например, натуральные числа 0,1,2,3,4,5,6,7,8,9 — слова в алфавите.

**Длиной слова** называется число символов в нем (длина пустого слова равна 0). Над словами можно производить следующие действия: выделение в слове его части и соединение.

Часть слова тоже является словом. Для выделения части слова нужно задать номер первого ее символа и количество символов в ней. Исходное слово при этом не меняется. Например, слово КР — часть слова БАНКРОТ, начинающаяся с четвертого символа и содержащая два символа.

**Соединить два слова** — это значит к первому слову справа приписать второе. Соединение слов обозначают знаком + (как сложение чисел). Например, КОМ+ПОТ=КОМПОТ, FOOT+BALL=FOOTBALL.

Для соединения слов переместительный закон не верен, а сочетательный — верен. От соединения с пустым словом ничего не меняется: среди слов пустое слово играет ту же роль, что число 0 среди чисел.

Слова можно сравнивать между собой. Для сравнения слов достаточно знать порядок символов в алфавите. Из двух слов больше то, которое в словаре должно стоять дальше. Например, СЛОН > КИТ, если буквы русского языка считать упорядоченными естественным образом.

В языке Бейсик для обозначения длины слова используется слово LEN, для обозначения соединения слов — знак +. Часть слова, начинающаяся с N-го символа и содержащая M символов, обозначается так:

MID\$(..., N,M).

Здесь вместо многоточия записывается исходное слово, заключенное в кавычки. Слово надо заключать в кавычки и тогда, когда оно участвует в других действиях со словами. Например: LEN("СОЛОВЕЙ-РАЗБОЙНИК"), "ХЛЕБ" + "ВОДА". Обнаружив кавычки, ЭВМ понимает, что она имеет дело со словом. Пустое слово обозначается так: "".

Для работы со словами используются переменные, значения которых — слова. Эти переменные называют **символьными**. Для того чтобы отличать символьные пере-

менные от числовых, в конце имени символьной переменной ставится символ \$. Например, A\$, B5\$.

Действия MID\$ и + можно использовать для "сборки" слов из значений символьных переменных. Для этого в операциях указывают имена переменных, значения которых используются при "сборке".

Значения символьных переменных можно менять с помощью команды присваивания, подобной команде присваивания для числовых переменных. Слева от знака равенства записывается имя символьной переменной, а справа — символьное выражение, составленное из имен символьных переменных с помощью операций MID\$ и +. Например:

$P\$ = \text{MID}\$(Q\$,8,2) + \text{MID}\$(Q\$,5,1) + \text{MID}\$(Q\$,9,2)$

В Бейсике значения символьных переменных можно запрашивать и сообщать при помощи команд INPUT и PRINT. Например, по команде

**INPUT A\$,B\$,C\$**

ЭВМ запросит три слова и обозначит их соответственно A\$,B\$ и C\$, а по команде

**PRINT A\$**

ЭВМ напечатает на экране значение переменной A\$.

Использование символьных переменных позволяет автоматизировать процессы обработки текстов. Например, можно определить, сколько раз в данном тексте встречается данное слово, заменить в тексте одно слово на другое слово, закодировать и декодировать текст. Очень часто используется кодирование слов последовательностями чисел с помощью так называемой ключевой фразы. Ключевой называется фраза, содержащая все символы некоторого алфавита. При кодировании каждый символ сообщения заменяется номером этого символа в ключевой фразе.

## Глава 10

# КОМПЬЮТЕРЫ В ОКЕАНЕ ИНФОРМАЦИИ

---

В предыдущих главах вы познакомились с тем, как применять ЭВМ для решения различных задач. Теперь мы расскажем о том, как устроены компьютеры, каковы перспективы их развития и использования.

Мы уже говорили, что ЭВМ — основной инструмент в технологии сбора, хранения и переработки информации. Вам известно и то, что без информации жизнь человечества так же невозможна, как без крови жизнь человека. А роль "кровеносной системы" в жизни общества играют каналы связи. Для каждого канала связи существенной характеристикой является его пропускная способность, т. е. *максимальное количество информации, которое может быть передано по этому каналу в единицу времени*. Имея дело с любым каналом связи, нельзя не учитывать его пропускную способность. Вы, конечно, знаете это по себе: ведь нервная система человека — тоже канал связи, и у нее есть своя пропускная способность. Именно поэтому, если несколько человек начинают говорить одновременно, их зачастую невозможно понять. Физиологи и психологи научились определять количество информации, которое человек может воспринимать с помощью органов чувств, удерживать в памяти и подвергать обработке. Эти данные позволяют, например, научно обоснованно организовать обучение, верно дозировать порции учебной информации.

Особое значение приобретает количественная оценка объемов обрабатываемой информации в связи с применением ЭВМ. Ведь компьютер также пронизан каналами связи и от их пропускной способности зависит эффективность его работы.

### § 37. ИНФОРМАЦИЯ И ИЗМЕРЕНИЕ ЕЕ КОЛИЧЕСТВА

Читая газеты и журналы, слушая радио, смотря телепередачи, каждый из вас (вольно или невольно) впитывает информацию. В интуитивном, житейском смысле под информацией понимают сведения, знания, события... И чем интереснее сообщаемые сведения, тем больше информации

(с житейской точки зрения) в них содержится. Не случайно наиболее содержательные телепередачи носят название "информационные" — сравните информационную программу "Новости" и "Утреннюю звезду", информационно-развлекательную программу "До и после полуночи" и "Спокойной ночи, малыши".

Можно ли, опираясь лишь на такое понимание информации, точно оценить ее количество? Конечно, нет! Ведь у каждого человека свои интересы. Одному интересна "Литературная газета", а другому "Известия". Каждый будет утверждать, что его любимая газета содержит больше информации.

Поэтому, говоря о восприятии информации человеком, нужно помнить, что количество информации существенно зависит от уровня ее понимания и интереса к ней. При этом понимание полученной нами информации сильно зависит от наших знаний. Скажем, телеграмма, полученная вами от вашей любимой тети, может содержать единственное слово "еду" без уточнения, кто, куда и с какой целью едет. Человек, не располагающий всеми перечисленными сведениями, вряд ли правильно поймет это сообщение. Итак, с житейской точки зрения количество информации мало связано с тем текстом (последовательностью букв), который эту информацию содержит, мало зависит от его длины.

Однако развитие разнообразных средств автоматической передачи, хранения и переработки информации (особенно при помощи ЭВМ) потребовало ввести количественную меру информации, не зависящую от субъективного человеческого восприятия. Чтобы понять, как вводится такая мера, давайте немного пофантазируем.

Представьте себе, что где-то по случаю вы приобрели небольшой радиотелескоп. И теперь, направив его в нужную точку звездного неба, вы с нетерпением ждете сигналов от братьев по разуму из какой-нибудь ближайшей галактики. Ваши ожидания не напрасны: ведь хорошо известно, что всякая высокая развитая цивилизация посылает в космос радиосигналы, стремясь установить контакт с мыслящими существами других миров. Вряд ли вы успели к началу передачи и вряд ли дождетесь ее конца. В вашем распоряжении будет только небольшая последовательность сигналов. Но и в ней содержится некая информация. Однако сообщение от братьев по разуму — это не телеграмма от любимой тети. Ведь вам, скорее всего, не известна никакая предварительная информация о внеземной цивилизации. Поэтому на вопрос о том, какую информацию содержит полученная вами последовательность сигналов, наиболее полным ответом будет предъявление самой

последовательности сигналов. Итак, в данном случае информацией выступает сама последовательность сигналов; каждый новый сигнал увеличивает количество этой информации.

Точно так же, когда речь идет об автоматической передаче информации, ее хранении и переработке, **информация** — это произвольная последовательность символов, т. е. любое слово (в терминологии главы 9); каждый новый символ увеличивает количество информации.

Как же измерить количество информации, содержащейся в данной последовательности символов? Да так же,

*Дорожите словом —  
носителем  
информации!*

как мы измеряем длину или массу чего-нибудь: сравнить с соответствующим эталоном. Надо только выбрать эталон. Например, в мультфильме "38 попугаев" эталоном длины служит

длина шага попугая. Однако попугай — малоудобный материал для эталона длины. Как вы, наверно, знаете, долгое время эталоном длины служил платиново-иридиевый стержень длиной 1 м. А сейчас эталон — длина волны излучения газа криптон-86.

Какое же слово взять в качестве эталона количества информации? Прежде чем выбирать эталонное слово, надо выбрать алфавит — материал, из которого будет сделан эталон. Обычно алфавит берут двухсимвольным. Например, он может состоять из цифр 0 и 1. *Эталонным* считается слово, состоящее из одного символа такого алфавита. *Количество информации, содержащейся в этом слове, принимают за единицу, называемую битом* (от binary digit — двоичный разряд). Имея эталон количества информации, надо теперь научиться сравнивать любое слово с эталоном.

*Длина слова —  
мера информации*

Проще всего сравнивать с эталоном те слова, которые записаны в том же двухсимвольном алфавите. **Количество информации**, заключенной в таком слове, по-

лагают равным его длине, ведь, как уже говорилось, каждый новый символ добавляет информацию. Чтобы измерить количество информации в произвольном слове, его кодируют в двухсимвольном алфавите, а затем находят длину получившегося слова. Разумеется, количество информации при таком определении зависит от способа кодирования. Если способ кодирования зафиксирован (а это обычно так, когда канал связи автоматический), то количество информации в сообщении определяется однозначно.

В современных ЭВМ каждый вводимый в машину символ (буква, цифра, знак операции и т. д.) кодируется сло-

вом длины 8 в двухсимвольном алфавите, например, буква "b" кодируется словом 01100010. Для удобства введены более крупные, чем бит, единицы количества информации. *Восемь бит информации называют байтом.* Байт — единица количества информации в Международной системе СИ. Значит, вводя в ЭВМ символ, вы передаете машине 1 байт информации. Количество информации в 1024 байта называется килобайтом и обозначается буквой К. Один мегабайт — это 1024 К.

Выбор двухсимвольного алфавита для кодирования информации в ЭВМ объясняется тем, что электронные элементы, из которых строились и строятся ЭВМ, могут находиться только в двух хорошо различимых устойчивых рабочих состояниях. По существу, эти элементы представляют собой обычные выключатели. Как известно, выключатель может быть либо включен, либо выключен. Третьего не дано. Отличие электронного выключателя от выключателя настольной лампы состоит в том, что в электронном выключателе нет механических движущихся частей и переключается он не рукой, а электрическим сигналом от другого выключателя. Время переключения поэтому оказывается очень малым, порядка  $10^{-9}$  с. Одно из состояний выключателя обозначают цифрой 1, другое — цифрой 0.

## ? Вопросы

1. Что такое информация?
2. Что служит эталоном количества информации?
3. Как измеряется количество информации?
4. Каковы единицы измерения количества информации?
5. Сколько бит в одном байте? Сколько байт в одном килобайте? Сколько килобайт в одном мегабайте?
6. Что называется пропускной способностью канала связи?



## Задания для самостоятельного выполнения

1. Сколько бит в одном мегабайте?
2. Сколько информации содержится на этой странице?
3. Укажите способ измерения количества информации в метрах; в килограммах.
4. Сколько символов можно закодировать словами длины 8 в алфавите, состоящем из 0 и 1?
5. Юстасу необходимо передать открытым текстом следующее сообщение: Дорогой Алекс! От всей души поздравляю Вас с Новым годом. Желаю счастья, здоровья и успехов в работе. Ваш Юстас.

Пеленгатор определяет место передачи, если она длится не менее 3 мин. С какой скоростью (в битах в секунду) должен передавать Юстас радиogramму?

а) Какую пропускную способность должен иметь канал связи, чтобы обеспечить бесперебойную передачу ?

в)\* Составьте математическую модель, алгоритм и программу решения следующей задачи:

7. Какое количество информации содержится в следующей картинке, напечатанной компьютером?

8. Представьте себе, что из Туманности Андромеды (одной из ближайших к нам галактик) пришло сообщение в виде последовательности радиосигналов двух видов (один из них мы обозначили нулем, а другой — единицей):

а) Сколько бит информации содержит это сообщение?

234

предварительной информации о цивилизации, попытайтесь расшифровать сообщение. (У к а з а н и е: разложите количество бит информации на простые множители.)

## § 38. КОДИРОВАНИЕ ЧИСЛОВОЙ ИНФОРМАЦИИ

В предыдущем параграфе мы рассказали о том, что вся информация в ЭВМ кодируется в двухсимвольном алфавите. Но закодировать информацию можно по-разному (о некоторых способах кодирования рассказывалось в главе 9). Одну и ту же информацию можно закодировать последовательностями нулей и единиц разной длины. Какой же из многих способов кодирования выбрать при хранении и переработке информации в ЭВМ? Конечно же более экономный, т. е. такой, при котором закодированная информация занимает меньшее место в памяти ЭВМ. Ведь память ЭВМ не безгранична.

Поэтому, когда создавались первые компьютеры, предназначенные главным образом для работы с числовой информацией, перед их создателями сразу встал вопрос о наиболее экономном кодировании чисел. К счастью, создателям ЭВМ не пришлось придумывать такой способ самим. Он был известен задолго до появления ЭВМ. Это — **двоичная система счисления**.

Еще в пятом классе вы узнали, что привычная нам система счисления — позиционная и десятичная. Для записи любых чисел в ней используется десять всем хорошо известных цифр (0,1,2,3,4,5,6,7,8,9), а значение каждой цифры (ее "вес") определяется той позицией, которую цифра занимает в записи числа. Так, запись 23 означает, что это число можно составить из 3 единиц и 2 десятков. Если мы поменяем позиции цифр, то получим совсем другое число — 32. Это число содержит 3 десятка и 2 единицы. "Вес" двойки уменьшился в десять раз, а "вес" тройки в десять раз возрос. Цифры десятичной записи числа — это просто коэффициенты его представления в виде суммы степеней числа 10 основания системы счисления:

$$25076 = 2 \cdot 10^4 + 5 \cdot 10^3 + 0 \cdot 10^2 + 7 \cdot 10^1 + 6 \cdot 10^0.$$

(Запишите сами аналогичное представление для числа 341,89.)

На самом деле числа можно записывать как сумму степеней не только числа 10, но и любого другого натурального числа, большего 1. Например, в Древнем Вавилоне использовалась система счисления с основанием 60. Делением часа на 60 минут, а минуты на 60 секунд мы обязаны именно этой системе счисления.

Тот факт, что основанием используемой нами системы счисления является число 10, объясняется, вероятно, тем, что природа наделила нас десятью пальцами на руках и десятью пальцами на ногах. Парнокопытные, наверное, выбрали бы двоичную систему счисления, т.е. систему счисления с основанием 2.

Как же получить запись числа в этой системе счисления? Вы уже догадались: нужно представить число как сумму степеней двойки и выписать коэффициенты такого представления. При записи числа в различных системах счисления договоримся указывать основание используемой системы, например, справа внизу после самого числа:

$$27_{10} = 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 11011_2$$

$$9,25_{10} = 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 + 0 \cdot 2^{-1} + 1 \cdot 2^{-2} = 1001,01_2$$

$$111010,1_2 = 1 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^1 + 1 \cdot 2^{-1} = 58,5_{10}$$

(Приведите сами несколько примеров перехода от десятичной записи числа к двоичной и наоборот.)

Для представления информации вне ЭВМ применять двоичную систему с ее громоздкой записью неудобно. Здесь часто используются восьмеричная и шестнадцатеричная системы. В восьмеричной системе счисления числа записываются с помощью восьми цифр: 0,1,2,3,4,5,6,7. Сама восьмерка (как и основание любой системы счисления) записывается двумя цифрами: 10 (почему?). Для записи чисел в шестнадцатеричной системе необходимо располагать уже шестнадцатью различными символами, используемыми как цифры. В качестве первых десяти шестнадцатеричных цифр используются те же, что и в десятичной системе. Для обозначения остальных шести цифр (в десятичной системе они соответствуют числам 10,11,12,13,14,15) используют буквы латинского алфавита — А,В,С,Д,Е,Ғ.

Особая привязанность программистов к восьмеричной и шестнадцатеричной системам объясняется тем, что переход к записи числа из этих систем от его двоичной записи прост. При переводе в восьмеричную систему двоичное число из трех или менее цифр записывается одной цифрой:

Восьмеричная запись	Двоичное представление
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Пусть теперь нам нужно представить двоичное число, например,  $1010110101111_2$  в восьмеричной форме. Для этого двоичную запись числа разбивают на группы по три цифры справа налево, начиная с младшего разряда:

1 010 110 101 111.

Затем каждую тройку цифр заменяют соответствующей цифрой восьмеричной системы счисления: 12657.

Еще несколько примеров:

$$100011_2 = 43_8$$

$$10001010_2 = 212_8$$

Обратный переход — от восьмеричной системы к двоичной — осуществляется заменой каждой восьмеричной цифры соответствующей тройкой двоичных цифр. Переход от двоичной системы к шестнадцатеричной и обратно осуществляется аналогично. Только шестнадцатеричная цифра заменяется не на три, а на четыре двоичные цифры. Например:

$$B5_{16} = 1011\ 0101_2$$

$$1001\ 1010\ 1111_2 = 9AF_{16}$$

## ? Вопросы

1. Что такое система счисления?
2. Как переходить от записи числа в восьмеричной системе счисления к записи в двоичной и обратно?
3. Сколько цифр используется:
  - а) в двоичной системе;
  - б) в шестнадцатеричной системе;
  - в) в стащестидесятипятичной системе?

## Задания для самостоятельного выполнения

1. а) Переведите в десятичную систему двоичные числа  $1000011110101$ ,  $1010010111011$ ,  $1110011001100110101$ .  
б) Переведите в двоичную систему десятичные числа 123, 456, 1024, 4095.  
в) Какое максимальное число можно записать в двоичной системе счисления двенадцатью цифрами?
2. а) Переведите в восьмеричную систему двоичные числа  $11101110101001101001100110101$ ,  $110110110000110001011010101$ ,  $111011101010011$ .  
б) Переведите в двоичную систему восьмеричные числа 54321, 545253, 777, 1010001.
3. а) Переведите в шестнадцатеричную систему двоичные числа  $1110111010100110100110011$ ,  $110110110110001011010101$ ,  $111011100001010011$ .  
б) Переведите в двоичную систему шестнадцатеричные числа 1AB, A1B, E2E4, E7E5.

## § 39. ПРИНЦИПЫ ФОН НЕЙМАНА

Вы уже хорошо усвоили, что вся информация, вводимая в ЭВМ и возникающая в ходе ее работы, хранится в памяти машины. Как устроена память? Ее можно представлять себе как длинную страницу, состоящую из отдельных строк. Каждая такая строка называется **ячейкой памяти** и, в свою очередь, разделяется на **разряды**. Содержимым любого разряда может быть либо 0, либо 1. Значит, в любую ячейку памяти записывается некоторый набор нулей и единиц — **машинное слово**. Все ячейки памяти занумерованы. Номер ячейки называют ее **адресом**.

Наличие у каждой ячейки адреса позволяет обращаться к любой ячейке, чтобы записать в нее новую информацию или извлечь ту информацию, которая в ней хранится. Нужно помнить, что при считывании хранящегося в ячейке машинного слова содержимое ячейки не изменяется. А при записи в ячейку информации прежнее содержимое ячейки исчезает (стирается).

Вы, наверно, вспомнили: похожим образом выполняется присваивание переменной нового значения. Сходство здесь не только внешнее. На каждую переменную в памяти ЭВМ отводится определенное количество ячеек; присваивание переменной нового значения — это запись в эти ячейки новой информации (соответствующего набора нулей и единиц).

Все ЭВМ работают в принципе одинаково. Для наглядности мы будем рассматривать совсем маленькую микро-микро-микро...ЭВМ, которую назовем КРОХА. В памяти этой машины всего восемь ячеек, в каждой из которых по двенадцать разрядов. Когда бы вы ни заглянули в память ЭВМ, в ее ячейках хранятся наборы нулей и единиц. На рисунке 57 — память машины КРОХА в один из моментов времени. Слева от ячеек записаны их адреса (в двоичной системе счисления). Поскольку ячеек мало, то для записи адреса достаточно трех двоичных разрядов (рис. 57).

000	0	0	0	1	0	0	0	0	0	1	1	0
001	0	0	1	1	1	0	1	1	1	0	1	1
010	1	1	1	0	0	0	0	0	0	0	0	0
011	0	0	0	0	0	0	0	0	0	0	0	0
100	0	1	1	1	0	0	1	1	0	0	0	0
101	0	0	0	0	0	1	0	0	0	1	1	1
110	0	0	1	0	1	0	1	0	1	0	1	0
111	0	1	1	0	1	1	1	0	1	1	0	1

Рис. 57

Каждому видно, что числа, которые можно записывать в ячейки памяти ЭВМ КРОХА, имеют не более 12 двоичных разрядов, т. е. не превосходят  $2^{12} - 1 = 4095$  (см. задачу 1, в к §38).

Создатели любой ЭВМ наделяют ее умением выполнять ряд элементарных операций (команд) — сложений, умножений и т. д. Но арифмометры или, скажем, арифметические калькуляторы тоже способны выполнять эти элементарные операции. Чем же тогда ЭВМ отличается от арифмометра? Вы, конечно, сразу ответите на этот вопрос. Главное отличие в том, что ЭВМ можно заставить выполнить без участия человека не только одну команду, но и длинную последовательность команд (программу). В этом и состоит один из основных принципов работы ЭВМ — **принцип программного управления**.

Каждая команда кодируется некоторой последовательностью из 12 нулей и единиц и помещается, как и число, в одной ячейке оперативной памяти. Команда состоит из двух частей: кодовой (3 разряда) и адресной (9 разрядов). Кодовая часть команды указывает, какое действие должно быть выполнено, а адресная определяет расположение в памяти компьютера исходных данных и результата. Общий вид команды машины КРОХА таков:

$$K A_1 A_2 A_3,$$

где  $K$  — код действия, а  $A_1$ ,  $A_2$ ,  $A_3$  — адреса ячеек памяти (на каждый адрес отводится по три разряда). Для выполнения команд служит специальное **арифметикологическое устройство**. Оно состоит из двух особых ячеек — **счетчика команд** и **регистра команд**, а также **сумматора**. При выполнении КРОХОЙ программы в счетчик команд последовательно заносятся номера ячеек, где содержатся исполняемые команды, сами команды помещаются в регистр команд, а в сумматоре происходят арифметические действия. Сумматор также имеет свою ячейку для промежуточных результатов вычислений. Отметим, что команды первых ЭВМ имели похожий вид и выполнялись аналогично (в современных ЭВМ команда может занимать несколько ячеек памяти).

Вот некоторые из команд ЭВМ КРОХА:

000  $A_1 A_2 A_3$  — переписать содержимое ячейки с адресом  $A_1$  в ячейку с адресом  $A_3$  (на выполнение этой команды содержимое ячейки  $A_2$  не влияет);

001  $A_1 A_2 A_3$  — сложить числа из ячеек, имеющих адреса  $A_1$  и  $A_2$ , а результат записать в ячейку с адресом  $A_3$ ;

011  $A_1 A_2 A_3$  — найти модуль разности чисел из ячеек с адресами  $A_1$  и  $A_2$ , результат записать в ячейку с адресом  $A_3$ ;

101  $A_1 A_2 A_3$  — перемножить числа из ячеек с адресами  $A_1$  и  $A_2$ , а результат записать в ячейку с адресом  $A_3$ ;

111  $A_1 A_2 A_3$  — остановиться и напечатать содержимое ячеек с адресами  $A_1$ ,  $A_2$  и  $A_3$ .

Для примера разберем, как выполняется команда

101 111 011 110.

По этой команде КРОХА извлечет из ячейки памяти с адресом 111 содержащееся там число и перепишет ее в ячейку сумматора. Затем КРОХА умножит содержимое ячейки сумматора на число, хранящееся в ячейке с адресом 011, а результат запишет в ячейку сумматора. После этого содержимое ячейки сумматора будет переписано в ячейку с номером 110.

Конечно, при выполнении этой (или какой-нибудь другой) команды может получиться число, большее чем 4095 (наибольшее число, вмещающееся в ячейку машины КРОХА). В этом случае КРОХА отказывается продолжать работу и выдает сообщение об ошибке.

Как видите, *одна и та же последовательность нулей и единиц, хранимая в ячейке памяти, может рассматриваться ЭВМ и как число, и как команда*. Это еще один важный принцип работы ЭВМ — **принцип хранимой программы**. На принципах двоичного кодирования, программного управления и хранимой программы основывается работа всех ЭВМ — от самых первых до современных. Эти принципы выдвинул американский математик Дж. фон Нейман в 40-х годах нашего века.

Как же КРОХА выполняет записанную в ее памяти программу? Свою работу она начинает с выполнения команды, записанной в ячейке с адресом 000. Выполнив ее, она переходит к следующей команде и т. д. до тех пор, пока не встретит команду "Остановиться" или такую команду, которую она не сможет выполнить.

Если запустить ЭВМ КРОХА, память которой заполнена так, как показано на рисунке 59, то выполняться команды, записанные в ячейках с адресами 000, 001 и 010. Разберитесь самостоятельно, как будет выполняться эта программа. Что будет записано в памяти ЭВМ после выполнения этой программы?

Язык команд КРОХИ сильно отличается, скажем, от Бейсика. Сразу видно: команды Бейсика универсальнее и записываются на естественном (английском) языке. Но главное в том, что программирование на Бейсике практически освобождает от забот по распределению памяти ЭВМ: машина сама определяет, где в ее памяти будет располагаться программа, а где — данные. Программирование в машинных командах — совсем другое дело. Прежде чем

приступать к написанию программы, программист должен спланировать распределение памяти. Не забывайте этого, когда будете решать задачи к данному параграфу.

Возникает естественный вопрос: как ЭВМ понимает программы, написанные на языке Бейсик, если, как вы теперь знаете, она умеет выполнять лишь самые простые команды? Дело в том, что при выполнении программы, записанной на языке программирования, в памяти ЭВМ все время (незримо для нас) присутствует программа-переводчик. Эта программа синхронно переводит каждую команду языка Бейсик на язык машинных команд. Программа-переводчик очень сложна: в ней десятки тысяч машинных команд. Но польза этой программы намного превосходит затраты на ее создание. Представьте себе, насколько больше сил и времени требовалось бы потратить на написание в машинных командах тех программ, которые вы писали на языке Бейсик!

Программы-переводчики (их называют трансляторами) вместе с другими полезными программами, написанными на "машинном языке", входят в так называемое **программное обеспечение** всех современных компьютеров. Как правило, оно включает базы данных, электронные таблицы, редакторы текстов, другие пакеты прикладных программ. Наличие хорошего программного обеспечения позволяет успешно использовать ЭВМ в своей повседневной работе различным специалистам (в том числе и школьникам), не являющимся программистами-профессионалами.

## Вопросы

1. Что такое ячейка памяти?
2. Что такое адрес ячейки памяти?
3. Что происходит с содержимым ячейки памяти при записи и считывании информации?
4. В чем заключается принцип программного управления работой ЭВМ?
5. В чем состоит принцип хранимой программы?
6. Для чего предназначены счетчик команд и регистр команд?
7. Для чего предназначен сумматор?
8. В чем главное отличие программирования в машинных командах от программирования на алгоритмическом языке (например, Бейсике)?
9. Зачем нужно и из чего состоит программное обеспечение ЭВМ?

## Задания для самостоятельного выполнения

1. Опишите, как выполняет ЭВМ КРОХА команду  
011 101 010 101.

2. Память ЭВМ КРОХА заполнена следующим образом:

а) 000	0	0	0	1	0	0	0	0	0	1	1	0
001	0	0	1	1	1	0	1	1	1	0	1	1
010	1	1	1	0	0	0	0	0	0	0	0	0
011	1	1	1	1	0	0	1	1	0	0	0	0
100	1	1	1	1	0	0	1	1	0	0	0	0
101	0	0	0	0	0	1	0	0	0	1	1	1
110	0	0	1	0	1	0	1	0	1	0	1	0
111	0	1	1	0	1	1	1	0	1	1	0	1

б) 000	0	0	0	1	0	0	0	0	0	1	1	0
001	0	0	1	1	1	0	1	1	1	0	1	1
010	0	1	1	1	0	0	1	1	1	1	1	1
011	1	1	1	0	0	0	0	0	0	0	0	0
100	0	0	1	1	1	0	1	1	0	0	0	0
101	0	0	0	1	0	1	0	0	0	1	1	1
110	0	0	1	0	1	0	1	0	1	0	1	0
111	0	1	1	0	1	1	1	0	1	1	0	1

Как будет выполняться программа?

3. Составить для ЭВМ КРОХА программу определения:

- а)° периметра прямоугольника (по длинам сторон);
- б) объема прямоугольного параллелепипеда (по длинам ребер);

в)\* половины полной поверхности прямоугольного параллелепипеда (по длинам ребер);

г) пути, пройденного телом при равноускоренном движении (исходные данные — начальная скорость, время движения и половина ускорения).

4°. Наделим машину КРОХА еще одной командой:

110  $A_1 A_2 A_3$  — если содержимое ячейки  $A_1$  больше содержимого ячейки  $A_2$ , то перейти к выполнению команды, записанной в ячейку с адресом  $A_3$ , иначе перейти к выполнению следующей команды.

Напишите для ЭВМ КРОХА программу нахождения наибольшего из двух чисел, записанных в ячейках ее памяти.



## Лабораторная работа 20

### ПРОГРАММИРОВАНИЕ ДЛЯ УЧЕБНОЙ ЭВМ КРОХА

Картинка, появившаяся на экране вашего компьютера, изображает "внутренний мир" ЭВМ КРОХА (рис.58). Тут и память, и экран, и арифметико-логическое устройство.

Кроме того, на экране находятся и некоторые поясняющие надписи. Одни из них объясняют значения функциональных клавиш, другие указывают режим работы ЭВМ (редактирование памяти, "пошаговое" или "непрерывное" исполнение программы).

Вы уже составили несколько программ для КРОХИ (см. задачи 3 и 4 к § 39). Давайте заставим ее выполнить программу вычисления периметра прямоугольника по его сторонам (задача 3,а). Введите программу в память ЭВМ, проставляя нули и единицы в разряды ячеек. А теперь заставьте КРОХУ найти периметр прямоугольника со сторонами 2 и 3. Числа 2 и 3, переведенные в двоичную систему счисления, вы должны записать в ячейки, которые отведены вами для данных. Обратите внимание, что при заполнении ячеек памяти их содержимое в двоичной записи дублируется в окне "Регистр команд", а рядом (справа) приводится их десятичное значение. Если нажать на клавишу <F1>, то КРОХА немедленно выполнит вашу программу и на "Экране" появится результат (если вы, конечно, предусмотрели его выдачу в команде остановки).

Правда, вряд ли вы успели рассмотреть подробно, как КРОХА выполняет программу. Чтобы КРОХА "не спешила" при выполнении программы, воспользуйтесь клавишей <F3>. При каждом нажатии на клавишу <F3> КРОХА выполняет один "такт" работы. Посмотрите, как изменяется сумматор, счетчик команд и регистр команд. Следите за комментариями!

Подсчитайте периметры прямоугольников с другими длинами сторон, а затем введите в память КРОХИ программу решения задачи 4 и запустите ее при различных исходных данных.

ПАМЯТЬ										ЭКРАН					
Адрес	Код операц.			A1			A2			A3			Двоичн.	Десят.	
000	0	0	0	0	0	0	0	0	0	0	0	0			
001	0	0	0	0	0	0	0	0	0	0	0	0			
010	0	0	0	0	0	0	0	0	0	0	0	0			
011	0	0	0	0	0	0	0	0	0	0	0	0	РЕДАКТОР		
100	0	0	0	0	0	0	0	0	0	0	0	0	F1 — Запуск программы		
101	0	0	0	0	0	0	0	0	0	0	0	0	F2 — Останов. программы		
110	0	0	0	0	0	0	0	0	0	0	0	0	F3 — Исполнение по шагам		
111	0	0	0	0	0	0	0	0	0	0	0	0	F4 — Подсказка		
Счетчик команд : 000															
Регистр команд : 000 000 000 000										0 — десятичное значение					
Регистр сумматора : 000000000000										0 — десятичное значение					

Рис. 58

## § 40. ПОКОЛЕНИЯ ЭВМ. МИКРОПРОЦЕССОРНЫЕ СИСТЕМЫ

Вам хорошо знаком внешний вид и основные устройства того персонального компьютера, с которым вы работали. В состав любого персонального компьютера входит клавиатура, с помощью которой работающий с ЭВМ человек вводит всевозможную информацию. Непременным атрибутом микроЭВМ является и дисплей, на экран которого отображается вводимая в ЭВМ информация и получаемые результаты. Основа вычислительной машины — процессор. В нем расположены **арифметико-логическое устройство, устройство управления**. Арифметико-логическое устройство осуществляет непосредственную обработку данных: сложение двух чисел, умножение одного числа на другое, перенос информации из одного места памяти в другое и т. д. Устройство управления координирует взаимодействие различных частей ЭВМ. В запоминающем устройстве (памяти ЭВМ) в закодированном виде хранится информация: и та, которая вводится в машину (программа и данные), и та, которая возникает в ходе ее работы. Многие персональные компьютеры располагают дополнительным запоминающим устройством (его называют внешним), использующим в качестве носителя информации магнитный диск (дискету).

За годы существования электронных вычислительных машин (а первая ЭВМ появилась в 1945 г.) их характеристики сильно изменились. Если первые машины совершали в секунду лишь несколько сотен операций и могли "помнить" чуть больше тысячи чисел, то для современных компьютеров доступно быстроедействие в десятки и сотни миллионов операций в секунду, а их память вмещает объемы информации, исчисляемые десятками мегабайт. Такой прогресс вычислительной техники был обеспечен изменением физических элементов, из которых строились ЭВМ. В первых машинах основным элементом была электронная лампа. В пятидесятых годах стали появляться ЭВМ на полупроводниках. Это было время вычислительных машин первого и второго поколений. Технологической основой появления вычислительных машин третьего поколения стало создание **интегральных микросхем**. Такое название получили электронные схемы, размещаемые на кремниевых пластинках малого размера. Микросхема позволяет разместить в том объеме, который раньше занимала электронная лампа, сотни тысяч и даже миллионы полупроводниковых элементов. Эти элементы действуют значительно надежнее и быстрее, чем элементы машин предыдущих поколений. Такие микросхемы стали называть **большими и сверхбольшими интегральными схемами** (БИС и СБИС). Название, конечно, связано не с размерами схем (наоборот,

сверхбольшие схемы становятся все меньше), а с количеством содержащихся в них элементов. БИС и СБИС позволили создать микропроцессор — основу современного персонального компьютера.

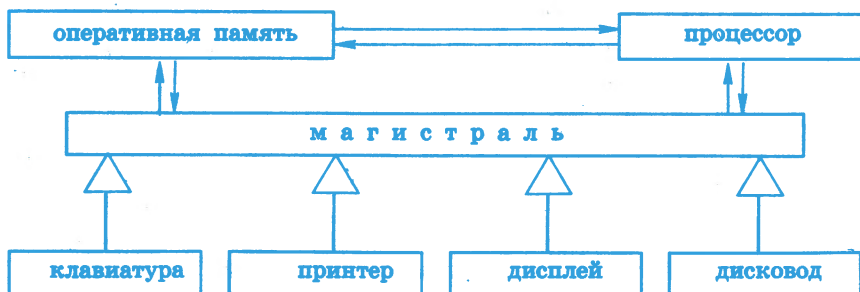
В предыдущем параграфе мы рассказали о том, как процессор и память взаимодействуют между собой во время выполнения программы. Но процессор, кроме того, организует работу остальных устройств ЭВМ — клавиатуры, дисплея, дисковод и т. д. Эти устройства осуществляют связь ЭВМ с "внешним миром" и потому называются внешними.

Процессор, выполняя определенную программу, координирует работу внешних устройств, посылая им и принимая от них информацию. Информация при этом передается в виде электрических импульсов двух видов. Импульс высокого напряжения обычно обозначают нулем, а импульс низкого напряжения — единицей. Тем самым информация в ЭВМ кодируется, как вы уже знаете, двумя символами.

Процессор связан с внешними устройствами через **магистраль**. По сути это телефонный кабель, пучок проводов. К магистрали параллельно подсоединены все внешние устройства, как телефоны к телефонному кабелю; "обращение" процессора к внешнему устройству похоже на вызов абонента по телефону. Все устройства пронумерованы. Когда нужно обратиться к внешнему устройству, в магистраль посылается его номер. Как и телефон, устройство может быть свободно или занято. Приняв сигнал "свободно", процессор посылает "абоненту" необходимую информацию.

Каждое внешнее устройство снабжено специальным "приемником" сигналов — его называют **контроллером**. Контроллер играет роль телефонного аппарата. Он принимает сигнал от процессора и дешифрует его. Например, контроллер экрана, приняв сигнал 01000001 (код буквы "А"), преобразует его в указание электронно-лучевой трубке изобразить на экране букву "А".

Вот общая схема микроЭВМ (треугольниками изображены контроллеры):



Отметим принципиально важное обстоятельство. Процессор лишь подает команду. Ему "безразлично", как она будет выполняться, — за это отвечает контроллер соответствующего внешнего устройства. Поэтому можно заменять одни внешние устройства ЭВМ другими, нужны лишь соответствующие контроллеры.

Присоединяя процессоры к различным внешним устройствам — станкам или роботам, можно сделать управление этими устройствами автоматическим. Успехи микроэлектроники позволили создать процессоры, полностью размещаемые на одной монокристаллической пластинке кремния или германия площадью менее  $0,1 \text{ см}^2$ , — **микропроцессоры**. Они дешевы, надежны. Технические системы, управляемые микропроцессорами, называют **микропроцессорными системами**. Создание этих систем — одно из основных направлений использования ЭВМ в современном мире.

## ? Вопросы

1. Что такое внешнее устройство ЭВМ?
2. Для чего служат магистраль и контроллеры?
3. Что такое микропроцессорная система?

## § 41. ГДЕ ПРИМЕНЯЮТСЯ ЭВМ

Возможность встраивать ЭВМ в уже существующие и вновь создаваемые машины и системы машин приведет в перспективе к тому, что человек будет освобожден от работ, связанных с тяжелыми, вредными или опасными условиями, с монотонными, однообразными, нетворческими действиями. Быть может, самые распространенные микропроцессорные системы — станки с программным управлением. В качестве внешних устройств выступают двигатели, перемещающие инструменты (резцы, сверла и т. д.) и обрабатываемые детали. Каждый двигатель имеет свой набор допустимых (элементарных) действий. Например, таких: "опустить резец", "переместить резец вправо на  $1 \text{ см}$ " и т. п. Для изготовления любой детали составляется алгоритм, состоящий из этих элементарных действий. Затем этот алгоритм переводится на язык команд процессора (они похожи на команды ЭВМ КРОХА). Процессор, выполняя получившуюся таким образом программу, дает внешним устройствам указания на выполнение действий, необходимых для изготовления детали. Например, хорошо знакомый вам ЧЕРТЕЖНИК можно рассматривать как станок с программным управлением. К сожалению, станки с программным управлением плохо адаптируются к обстановке: даже если деталь сломается, станок продолжит ее обработку.

Более сложными микропроцессорными системами являются промышленные роботы. Они снабжаются простейшими "органами чувств" — датчиками, позволяющими своевременно реагировать на изменения ситуации. Применение роботов позволяет полностью автоматизировать работу цехов и целых заводов. Однако всегда будут области деятельности, где ЭВМ не сможет полностью заменить человека. Это те области, где требуется неформальный, творческий подход к делу. Тем не менее ЭВМ может облегчить и творческий труд. Для этого создаются так называемые **автоматизированные рабочие места (АРМ)**. АРМ — это комплекс высокопроизводительных устройств, присоединенных к ЭВМ, заменяющий привычные малопроизводительные орудия труда (справочники, пишущие машинки, телефоны и т. д.). Разумеется, АРМ композитора не подходит директору завода: состав АРМ определяется его назначением.

Например, АРМ конструктора состоит из дисплея, на котором можно изображать чертежи, графопостроителя для выдачи чертежей на бумагу, принтера и других внешних устройств. АРМ, конечно, не может выполнять свои функции без соответствующего программного обеспечения. Для конструкторов это различные **системы автоматизированного проектирования (САПР)**. Такая система подскажет конструктору, какие материалы он может использовать, поможет оформить документацию, произведет расчет элементов конструкции, создаст чертеж по эскизу.

Программное обеспечение АРМ директора предприятия должно содержать **автоматизированную систему управления (АСУ)**. АСУ быстро выдаст на экран дисплея или на бумагу оперативную сводку о положении дел на предприятии (наличие ресурсов, ход выполнения заказа, сведения о работниках предприятия и т. д.), поможет в выборе смежников, а также экономической стратегии и тактики. Создаются АСУ, предназначенные для обеспечения оптимального взаимодействия уже не отдельных станков или автоматических линий, а цехов, производственных объединений в масштабах целой отрасли. Они возьмут на себя сбор, обработку, хранение и представление информации, необходимой для обоснованного принятия решений в хозяйственной, научной и других сферах деятельности.

Расширение области применения ЭВМ происходит не только за счет увеличения количества устройств, к которым подсоединяется ЭВМ, но и за счет роста "интеллектуальных способностей" компьютеров. Так информационно-поисковые системы и базы данных перерастают в **базы знаний**. В базах знаний будут храниться не только данные, но и правила вывода новых утверждений из уже имеющихся. Такая база способна не только выдавать хранящуюся

юся в ней информацию, но и логически "рассуждать", выводя новые знания. Немного пофантазировав, представьте себе, что у вас есть дискета с базой знаний по школьной геометрии. Скажем, вызывает вас учитель: мол, решите такую-то задачу. Недолго думая, вы вставляете дискету в дисковод школьного компьютера. Несколько нажатий клавиш — и на экране появляется подробное решение. Замечательно! Просто мечта двоечника!

В заключение о применении ЭВМ в быту. "Быт заедает". Эта фраза, наверное, знакома каждому из вас. Быт заедает — и приходится отложить встречу с друзьями, быт заедает — и интересная книга так и будет прочитана, быт заедает — и отодвигается свидание с природой (и если бы только с природой!). Для того чтобы облегчить нелегкий домашний труд, придуманы многочисленные приборы и приспособления — утюги, стиральные машины, пылесосы... Но, увы, они не могут выполнять работу сами по себе, без постоянного контроля и управления со стороны человека. По сути дела, эти приборы заменяют скучную и рутинную физическую работу умственной, но столь же скучной и рутинной работой. Ясно, что эту работу лучше поручить ЭВМ — ведь она и создана, чтобы освободить нас от рутины. Вы, наверное, уже догадались, как это сделать, — бытовая техника в принципе не отличается от промышленной и тоже может выступать в роли внешних устройств для ЭВМ. Вы подходите к двери своей квартиры, звоните. Приятный голос (тембр выбирали вы сами) спрашивает: "Кто там?" Услышав ответ, вам открывают дверь, и тот же приятный голос приветствует вас (с помощью датчика случайных чисел выбирается одно из придуманных вами приветствий). Тут же вам сообщается о всех телефонных звонках, которые были в ваше отсутствие. Из кухни доносится запах обеда, приготовленного по вашей программе и подогретого к вашему приходу. Все в квартире блестит чистотой. Белье выстирано, высушено и выглажено заботливой механической рукой робота-манипулятора...

Конечно, мы не навязываем вам именно такой образ жизни. Многие, например, предпочтут готовить себе обед без участия ЭВМ: человек вкладывает душу в любимое дело, а машина лишь добросовестно исполняет алгоритмы. Но согласитесь, возможность использования ЭВМ делает человека более свободным, предоставляя ему выбор. Точно так же, если, скажем, у человека есть видеоманитофон, он может выбирать — смотреть ему фильм дома или в кинотеатре, а если видеоманитофона нет, то и выбора нет — придется идти в кинотеатр.

Как видите, применения ЭВМ многообразны, а их будущее фантастично.

## ? Вопросы

1. Для чего предназначены станки с числовым программным управлением?
2. Что такое автоматизированное рабочее место?
3. Что такое система автоматизированного проектирования?
4. Что такое автоматизированная система управления?
5. Чем отличается база знаний от базы данных?



## Задания для самостоятельного выполнения

1. Опишите, из чего, на ваш взгляд, должно состоять автоматизированное рабочее место:
  - а) учащегося;
  - б) учителя;
  - в) директора школы;
  - г) министра образования Российской Федерации.
2. Какие бытовые приборы можно, по вашему мнению, оснастить микропроцессорами? Опишите дополнительные возможности, которые при этом появятся.
3. Каковы, по вашему мнению, возможности применения ЭВМ:
  - а) в науке;
  - б) в образовании;
  - в) в сельском хозяйстве;
  - г) в искусстве?
4. Посетив вычислительный центр какого-либо предприятия, опишите, какие задачи решаются в этом центре с помощью ЭВМ.
5. Напишите рассказ (эссе, очерк, фельетон, роман, учебник) о применении ЭВМ в современном мире.



## КОНСПЕКТ ГЛАВЫ 10

Успешная работа представителей многих профессий требует постоянного изучения больших объемов информации. Поэтому информация и средства ее обработки становятся полноправной частью производительных сил общества.

Обычное, "бытовое" определение информации как сведений, интересных кому-либо, а также определение количества информации как "меры интереса" слишком субъективно, чтобы его можно было использовать при работе с автоматическими каналами связи и с ЭВМ. Развитие разнообразных средств автоматической передачи, хранения и переработки информации (особенно при помощи ЭВМ) потребовало ввести количественную меру информации, не зависящую от субъективного человеческого восприятия.

Информация — это произвольная последовательность

символов; каждый новый символ увеличивает количество информации.

За единицу количества информации (1 бит) берут количество информации, содержащейся в слове длины 1 в двухсимвольном алфавите (этот алфавит может состоять, например, из 0 и 1). Количество информации, содержащейся в любом слове, записанном в двухсимвольном алфавите, равно длине этого слова. Чтобы измерить количество информации в произвольном слове, его кодируют в двухсимвольном алфавите (нулями и единицами), затем находят длину получившегося слова.

Более крупные, чем бит, единицы количества информации — байт (8 бит), килобайт (1024 байта), мегабайт (1024 килобайта). Каждый символ, введенный в ЭВМ, несет 1 байт информации.

Общие принципы работы ЭВМ не изменялись с 1945 года. Вот эти принципы:

1) двоичное кодирование информации (любая информация в ЭВМ кодируется двумя символами — двумя состояниями электронных элементов);

2) хранимая программа (программы, по которым работает ЭВМ, хранятся в оперативной памяти, каждая команда кодируется последовательностью нулей и единиц; данные, с которыми оперирует программа, также располагаются в оперативной памяти);

3) программное управление (ЭВМ работает автоматически по введенной в нее программе).

Для двоичного кодирования чисел обычно используется двоичная система счисления.

Основа вычислительной машины — процессор. Одна из постоянных забот процессора — организация работы остальных устройств компьютера: клавиатуры, дисплея, дисковода и так далее. Эти устройства называются внешними.

Каждое внешнее устройство снабжено специальным "приемником" сигналов — его называют контроллером. Контроллер принимает сигнал от процессора и дешифрует его, чтобы соответствующее устройство смогло принять этот сигнал и правильно отреагировать на него. Важно отметить, что процессор лишь подает команду и ему "безразлично", как она будет выполняться, — за это отвечает контроллер соответствующего внешнего устройства. Поэтому можно заменять одни внешние устройства ЭВМ другими, нужны лишь соответствующие контроллеры. Присоединяя процессоры к различным внешним устройствам — станкам или роботам, можно сделать управление этими устройствами автоматическим и освободить человека от многих вредных и тяжелых работ.

## ПРЕДМЕТНЫЙ УКАЗАТЕЛЬ

- Адрес ячейки памяти 238
- Алгоритм 41,45,54
  - вспомогательный 117,120
  - линейный 70
- Алфавит 209
  - азбуки Морзе 210
- Анализ результатов работы ЭВМ 41
- Аргументы вспомогательного алгоритма 118
  
- База данных 16
  - знаний 248
- Байт 233
- Банк данных 17
- Бит 231
- Блок-схема 73
  - линейного алгоритма 73
  - разветвляющегося алгоритма 73
  - цикл "Для каждого" 162
  - "пока" 98
- Ветвление 71, 75
- Вызов вспомогательного алгоритма 117
- Данные исходные в модели 32
  - — — ВЫЧИСЛИТЕЛЯ 50, 111, 159
  - — — РОБОТА-МАНИПУЛЯТОРА 156
  - — — ЧЕРТЕЖНИКА 63, 78
- Дискета 8
- Дисковод 8
- Дисперсия 184
- Дисплей 8
- Длина слова 210
- Действия исполнителя допустимые 49,73
- Задача артиллерийская 91,93
  - биологическая 132, 137
  - о дорожном происшествии 82, 87
  - коммерческая 30,57
  - криминалистическая 185, 188
  - о двух лодках 123, 130
  - о дешифровке сообщений 221, 227
  - о замене в слове одной буквы другой буквой 217
  - о наилучшем расположении станции 143, 148
  - о нахождении площади фигуры методом Монте-Карло 109, 114
  - о подкове 32
  - о трех туристах 11, 13
  - о шифровке сообщений 221, 227.
  - об определении числа вхождения буквы в слово 216
  - об отводе чертежника от края листа 78,86
  - обработки метеорологических наблюдений 158, 168
  - оптимизации 180
  - планирования 170, 176, 179, 182
  - экологическая 103, 107
- "Замораживание" переменных 120
- Запись арифметических выражений 9
  - элемента таблицы 155
- Запрос для поиска данных в ИПС 17
- Запросов правила записи 17
- Знак возведения в степень 9
  - деления 9
  - умножения 9
- Значение среднее 184
  
- Имитация исполнителя с помощью ЭВМ 50
- Имя вспомогательного алгоритма 117
  - переменной 52, 214
- Информатика 5
- Информация 5, 231, 232
- Информационно-поисковая система 16
  - — "Озера" 17,20
- Исполнитель алгоритмов 37,49
  - ВЫЧИСЛИТЕЛЯ 50
  - ЧЕРТЕЖНИКА 62
  - РОБОТА-МАНИПУЛЯТОРА 156

- Канал связи 230
- Канала связи пропускная способность 230
- Килобайт 233
- Клавиатура ЭВМ 8
- Клавиша "выбор" 10
  - "переход" 14
  - "перевод строки" 9
- Клавиши функциональные 9
- Количество информации 232
- Команда ВЫПОЛНИТЬ ПОДПРОГРАММУ 129
  - ЗАПРОСИТЬ 57
  - машинная 240
  - НАЧАТЬ РАБОТУ 87
  - присваивания 58
  - ПУСК 61
  - СООБЩИТЬ 8
  - СТОП 90
  - ТАБЛИЦА 168
  - ТЕКСТ 61
  - END 195
  - DIM 205
  - GOSUB 203
  - GOTO 196
  - INPUT 194
  - LET 193
  - LIST 61
  - PRINT 195
  - REM 202
  - RETURN 203
  - RUN 61
  - STOP 195
- Команды микроЭВМ КРОХА 239, 240, 242
- Контроллер 245
- Курсор 8
- Магистраль 245
- Мегабайт 233
- Место рабочее автоматизированное 247
- Метод деления пополам 93
  - Монте-Карло 110
  - пошаговой детализации 138
- МикроЭВМ 7
  - КРОХА 238
- Микропроцессор 246
- Микросхема большая 244
  - интегральная 244
  - сверхбольшая 244
- Модель задачи 32
  - — компьютерная 37
  - — математическая 37
  - — вероятностная 114
- Обеспечение ЭВМ программное 241
- Обозначение переменных в Бейсике 194
  - функций в Бейсике 194
- Операция выделения части слова 210
  - соединения слов 210
- Отладка программы 107
- Оформление ветвления в неполной форме 72
  - — в полной форме 72
  - — в языке Бейсик 197
  - вспомогательных алгоритмов 117, 118
  - — — в языке Бейсик 126, 203
  - цикла "Пока" 98
  - — — в языке Бейсик 200
  - — "Для каждого" 160
  - — — в языке Бейсик 201
- Ошибка округления 130
- Пакет прикладных программ 178
  - — — "Оптим" 179, 181
- Память 8
  - внешняя 8
  - оперативная 8
- Переменная 52
  - сигнальная 131
  - символьная 214
- Переменные, используемые в основном и вспомогательном алгоритмах 120
- Подпрограмма 128
- Подпрограммы в языке Бейсик 203
- Порядок слов алфавитный 211
- Построение алгоритма последовательное 138
- Предположения, на которых основана модель 32
- Признак 16
- Признака значение 16

- Принтер 8
- Принцип программного управления 239
  - хранимой программы 240
- Программа 41
  - "Клавиатура" 8
  - прикладная 178
- Процессор 7
- Разряд 238
- Регистр команд 239
  - сумматора 239
- Редактор графический 24
  - — учебный 25
  - текстов 22
  - — учебный 23
- Режим "отладка" 107
- Результаты вспомогательного алгоритма 118
  - в модели задачи 32
- Робот промышленный 247
- Символ пробел 210
- Система автоматизированного проектирования (САПР) 25, 247
  - — управления 247
  - микропроцессорная 246
  - счисления 235
  - — восьмеричная 236
  - — двоичная 235
  - — десятичная 235
  - — позиционная 235
  - — шестнадцатеричная 236
- Слово 209
  - машинное 238
  - пустое 210
- Способ задания последовательности рекуррентный 133
- Станки с программным управлением 246
- Схема решения задач с использованием ЭВМ 42
- Счетчик команд 239
  - цикла 160
- Таблица линейная 155
  - прямоугольная 155
  - электронная 10
  - — учебная 13
- Тело цикла 98
- Транслятор 241
- Указатель "Конец ветвления" 72
  - "Конец подпрограммы" 129
  - "Конец цикла" 98
- Условие, проверяемое исполнителем 73
  - — — ВЫЧИСЛИТЕЛЕМ 77
  - — — РОБОТОМ-МАНИПУЛЯТОРОМ 157
  - — — ЧЕРТЕЖНИКОМ 78
  - NOT ( ) 197
- Устройства ввода-вывода 8
  - ЭВМ внешние 245
- Устройство арифметико-логическое 239
  - управления 244
- Фраза ключевая 221
- Функция извлечения корня SQR(X) 130
  - получения случайного числа RND (A) 111
  - — — — в языке Бейсик 111, 201
  - целевая 180
  - LEN 213
  - MID\$ 215
- Цели исполнителя достижимые 62
- Цикл 98
- Цифра 235
- Число случайное 111
- ЭВМ 6
- Эксперимент вычислительный 62
  - компьютерный 42
- Язык алгоритмический 138
  - программирования 41
  - — Бейсик 192
- Ячейка памяти 238

## ОГЛАВЛЕНИЕ

Предисловие . . . . .	3
Как работать с учебником . . . . .	4
<b>Глава 1. Знакомство с ЭВМ . . . . .</b>	<b>5</b>
§ 1. Что умеет ЭВМ . . . . .	—
Лабораторная работа 1. Первый раз в дисплейном классе . . . . .	7
§ 2. Организация вычислений с помощью ЭВМ . . . . .	10
Лабораторная работа 2. Учебная электронная таблица . . . . .	13
§ 3. Информационно-поисковые системы . . . . .	15
Лабораторная работа 3. Решение задач с использованием учеб-	
ной ИПС . . . . .	20
§ 4. Редактирование текстов с помощью ЭВМ . . . . .	21
Лабораторная работа 4. Учебный редактор текстов . . . . .	23
§ 5. Редактирование изображений с помощью ЭВМ . . . . .	24
Лабораторная работа 5. Учебный графический редактор . . . . .	25
Конспект главы 1 . . . . .	27
<b>Глава 2. Искусство построения моделей . . . . .</b>	<b>30</b>
§ 6. Что такое хорошо и что такое плохо поставленная задача . . . . .	—
§ 7. Модели задач и исполнители . . . . .	35
§ 8. Модель построена, что дальше? . . . . .	41
Конспект главы 2 . . . . .	42
<b>Глава 3. Алгоритм и его свойства . . . . .</b>	<b>44</b>
§ 9. Понятие алгоритма . . . . .	—
§ 10. Исполнители алгоритмов . . . . .	48
Лабораторная работа 6. Коммерческая задача . . . . .	57
§ 11. Достижимые цели . . . . .	62
Конспект главы 3 . . . . .	66
<b>Глава 4. Ветвления в алгоритмах . . . . .</b>	<b>70</b>
§ 12. Ветвления . . . . .	—
§ 13. Ветвления и исполнители алгоритмов . . . . .	77
§ 14. Задача о дорожном происшествии . . . . .	82
Лабораторная работа 7. Решение задач с использованием развет-	
вляющихся алгоритмов . . . . .	86

§ 15. Артиллерийская задача . . . . .	91
Лабораторная работа 8. Метод деления пополам . . . . .	93
Конспект главы 4 . . . . .	94
<b>Глава 5. Циклы в алгоритмах . . . . .</b>	<b>97</b>
§ 16. Циклы . . . . .	—
§ 17. Циклы и исполнители алгоритмов . . . . .	102
Лабораторная работа 9. Решение задач с использованием циклических алгоритмов . . . . .	107
§ 18. Метод Монте-Карло . . . . .	108
Лабораторная работа 10. Решение задач с использованием циклических и разветвляющихся алгоритмов . . . . .	114
Конспект главы 5 . . . . .	115
<b>Глава 6. Вспомогательные алгоритмы . . . . .</b>	<b>116</b>
§ 19. Понятие вспомогательного алгоритма . . . . .	—
§ 20. Вспомогательные алгоритмы и исполнители алгоритмов . . . . .	122
Лабораторная работа 11. Подпрограммы и их использование при решении задач . . . . .	128
§ 21. Задача о производстве вакцины . . . . .	132
Лабораторная работа 12. Решение задач с использованием ветвлений, циклов и вспомогательных алгоритмов . . . . .	136
§ 22. Последовательное построение алгоритмов . . . . .	138
Лабораторная работа 13. Решение задач с помощью пошаговой детализации . . . . .	142
§ 23. Задача о выборе места для строительства железнодорожной станции . . . . .	143
Лабораторная работа 14. Задача о выборе места для строительства железнодорожной станции . . . . .	148
Конспект главы 6 . . . . .	151
<b>Глава 7. Организация данных . . . . .</b>	<b>154</b>
§ 24. Табличный способ организации данных . . . . .	—
§ 25. Таблицы и исполнители . . . . .	156
Лабораторная работа 15. Решение задач с использованием табличной формы организации данных . . . . .	166
§ 26. Задачи планирования . . . . .	169
Лабораторная работа 16. Решение задач планирования . . . . .	176
§ 27. Пакеты прикладных программ . . . . .	178
Лабораторная работа 17. Работа с прикладной программой "Оптимизация" . . . . .	181
§ 28. Немного статистики . . . . .	183
Лабораторная работа 18. Криминалистическая задача . . . . .	188
Конспект главы 7 . . . . .	189
<b>Глава 8. Язык программирования Бейсик . . . . .</b>	<b>192</b>
§ 29. Основные команды языка Бейсик . . . . .	193
§ 30. Ветвления в языке Бейсик . . . . .	196
§ 31. Циклы в языке Бейсик . . . . .	200

§ 32. Подпрограммы в языке Бейсик . . . . .	202
§ 33. Организация данных в языке Бейсик . . . . .	204
Конспект главы 8 . . . . .	206
<b>Глава 9. Символьные переменные . . . . .</b>	<b>209</b>
§ 34. Слова и действия с ними . . . . .	—
§ 35. Символьные переменные и операции над ними в языке Бейсик . . . . .	213
§ 36. Использование ЭВМ для шифровки и дешифровки сообще- ний . . . . .	221
Лабораторная работа 19. Задачи шифровки и дешифровки . . .	227
Конспект главы 9 . . . . .	228
<b>Глава 10. Компьютеры в океане информации . . . . .</b>	<b>230</b>
§ 37. Информация и измерение ее количества . . . . .	—
§ 38. Кодирование числовой информации . . . . .	235
§ 39. Принципы фон Неймана . . . . .	238
Лабораторная работа 20. Программирование для учебной ЭВМ КРОХА . . . . .	242
§ 40. Поколения ЭВМ. Микропроцессорные системы . . . . .	244
§ 41. Где применяются ЭВМ . . . . .	246
Конспект главы 10 . . . . .	249
<b>Предметный указатель . . . . .</b>	<b>251</b>

## **ВНИМАНИЮ ПРЕПОДАВАТЕЛЕЙ ИНФОРМАТИКИ**

**Программное обеспечение для нашего учебника можно получить, обратившись по адресу:**

**620151, г. Екатеринбург, улица К. Либкнехта, 9, Уральский государственный педагогический университет, проблемная лаборатория по применению электронно-вычислительной и микропроцессорной техники в учебном процессе.**

**Телефон 57-83-63.**

*Авторский коллектив*

Составьте задание на поиск

ГЛУБИНА(М)	КОНТИНЕНТ	СТРАНЫ	ВЫСОТА(М)	ПРОИСХОЖДЕНИЕ

Задание на поиск

КОНТИНЕНТ	= АФРИКА	и	ГЛУБИНА(М) > 10
Печать на экран			

Образец задания

Площадь	>	1200 или Язык	= английский
---------	---	---------------	--------------

Назначение клавиш

+ -на символ вправо + -на символ влево + -к строке печати + -к параметрам печати + -в поле справа	BS-удалить предид. символ DEL-удалить текущий символ ESC-выход в меню	ЦИФРЫ F1-печать информации
---	---	-------------------------------

Имена:	A	B	C	D	E	F	G
Формулы:						A#C	A#D
Палатки	7	1	1			7	
Котел	0.35	2			2		
Топор	3	2		2			6
Консер	0.4	15	5	10		2	4
Крупа	0.5	4			4		
Сухари	0.4	3			3		
Сахар	0.5	2			2		
Чай	0.05	6			6		
Сгущен	0.45	5			5		
М							
Е							
Функции:						Sum	Sum
Значения						3	10

F1-Min F2-Max F3-Sum F4-S/n F5-S2/n

Номер	x(1)	x(2)	x(3)	x(4)	x(5)	x(6)	Знак	Скоб. чтение
(1)	1						>	2
(2)		1					>	3
(3)	3		.5	1.5			<	32
(4)	3	6	4	4			<	54
(5)	3	3	5	6			<	36
(6)			1				<	5
(7)				1			<	4
(8)								
(9)								
(10)								
(11)								
(12)								
Г:	5	7	12	18				

Ответ:

81.4	2	3	4.2	8
------	---	---	-----	---

F1 min F2 max F3 Запись F4 Чтение F5 Удалить

Сброс Выполнение

Выполняю команду по адресу 001

сообщения

PC 001 процессор  
cmd [100]+[110]  
acc 0 011 100 110 000

экран  
0  
0  
0

команда	число	адр	команда	A1	A2	A3
[A1]+[A3]	262	000	0 0 0	1 0 0	0 0 0	1 1 0
[A1]+[A3]	955	001	0 0 1	1 1 0	1 1 1	0 1 1
стоп	3584	010	1 1 1	0 0 0	0 0 0	0 0 0
[A1]+[A3]	0	011	0 0 0	0 0 0	0 0 0	0 0 0
[A1]-[A2] +[A3]	1840	100	0 1 1	1 0 0	1 1 0	0 0 0
[A1]+[A3]	71	101	0 0 0	0 0 1	0 0 0	1 1 1
[A1]-[A2] +[A3]	1840	110	0 1 1	1 0 0	1 1 0	0 0 0
[A1]-[A2] +[A3]	1773	111	0 1 1	0 1 1	1 0 1	1 0 1

