ББК 32.973
К 60
УДК 681.3.06

Коляда М.Г.

К 60 Окно в удивительный мир информатики. — Д.: Сталкер, 1997. — 448 с.

ISBN 966-7104-40-0

Книга, которую вы держите в руках, может быть использована как оригинальное учебное пособие в помощь тем, кто решил войти в мир информатики и вычислительной техники.

Она предназначена тем читателям, которые собираются стать не только пользователями вычислительной техники, но и хотят использовать компьютер для решения своих задач. Они найдут в ней необходимые сведения о том, как работает компьютер, как составляются программы на языке Бейсик.

Отличительной особенностью книги является наличие в ней не только базового материала по информатике, но и удивительных фактов о неисчерпаемых возможностях и истории создания ЭВМ. Читатель "окунется" в мир логики и ознакомится с путеводителем в океане программного обеспечения.

Рецензенты: доктор физ.-мат.наук, проф. *Г.А. Атанов*, канд. техн. наук *Е.В. Воеводенко*

Оформление обложки *В.И. Гук*

ББК 32.973

ISBN 966-7104-40-0

Эта книга – первый шаг в мир компьютеров и программирования.

Автор стремился сделать ее доступной для всех тех, кто хотел бы научиться пользоваться компьютером. Она, несомненно, представляет интерес и для тех, кто хочет писать свои собственные программы для ЭВМ на языке Бейсик.

Книга охватывает широкий круг вопросов – от истории создания вычислительной техники до основ программирования и элементов математической логики. Она также дает представление о том, как устроена ЭВМ, как работают основные ее элементы, как ориентироваться в безбрежном море программного обеспечения. Но все же основное внимание уделяется алгоритмизации. Вы шаг за шагом будете постигать язык программирования Бейсик. А когда наступит понимание написанных на этом языке программ, вы сможете приспособить или отладить программы, написанные в книге, а отсюда – лишь один шаг до написания своих собственных программ. Но знание языка программирования – это не "конечная остановка". Самое главное – умение решать поставленные жизнью задачи, разлагать, абстрагировать их в виде, удобном для понимания компьютером. Поэтому вам необходимо постичь азы логики, научиться логически мыслить, а это как раз и есть ключ ко многим тайнам.

Но перед вами не развлекательная книга, которую можно читать через строчку. Несмотря на то, что автор стремился как можно более доступно и понятно изложить материал, все же нужно предупредить читателя, что мир вычислительной техники не только увлекателен и многообразен, но и сложен. Современные ЭВМ и их программное обеспечение по праву относят к самым трудоемким созданиям человеческого разума.

Идеальным будет положение тех читателей, у которых есть возможность поработать с ЭВМ. Лучший способ научиться

всему сказанному в книге – самостоятельно выполнить рассматриваемые операции и действия.

Работая с пособием, учитесь выделять в тексте главную мысль и излагать ее своими словами. Вопросы к параграфам помогут вам понять, хорошо ли усвоен новый материал. Если вы не запомнили основные понятия данного параграфа, просмотрите текст еще раз.

Все то, что вам необходимо твердо запомнить, напечатано в таких рамках.

Книга содержит много разнообразных задач – простых и посложнее. Они помещены в разделе "Тренировка". Точно так же, как спортсмены тренируют свои мышцы, старайтесь тренировать свои навыки. Все задания для тренировки непосредственно связаны с текстом параграфа, поэтому, если вы не можете их выполнить, не огорчайтесь – снова проработайте параграф. И вновь приступайте к решению. Для контроля вы можете сверить свое решение с решением в конце книги.

После каждой главы приведена сводка основных содержащихся в ней фактов, понятий, идей. Это поможет вам при повторении изученного материала.

В нашей книге, как и в любом учебнике, вам встретятся новые термины. Определения, свойства и правила, т.е. базовые понятия, выделены жирным шрифтом. Разумеется, их желательно знать на память, но не просто зазубривать, а понимать их смысл и уметь применять на практике. Если вы хотите быстро найти, что означает то или иное слово, воспользуйтесь предметным указателем. Он расположен в конце книги.

Книга содержит большое количество занимательных и познавательных фактов из мира информатики и вычислительной техники. Это материалы рубрик "Что может компьютер", "Известно ли вам, что... ", "Лицо прошлого с силуэтами будущего", биографии выдающихся личностей из этой области. В общем, надеемся, что читать эту книгу вам будет не только полезно, но и интересно.

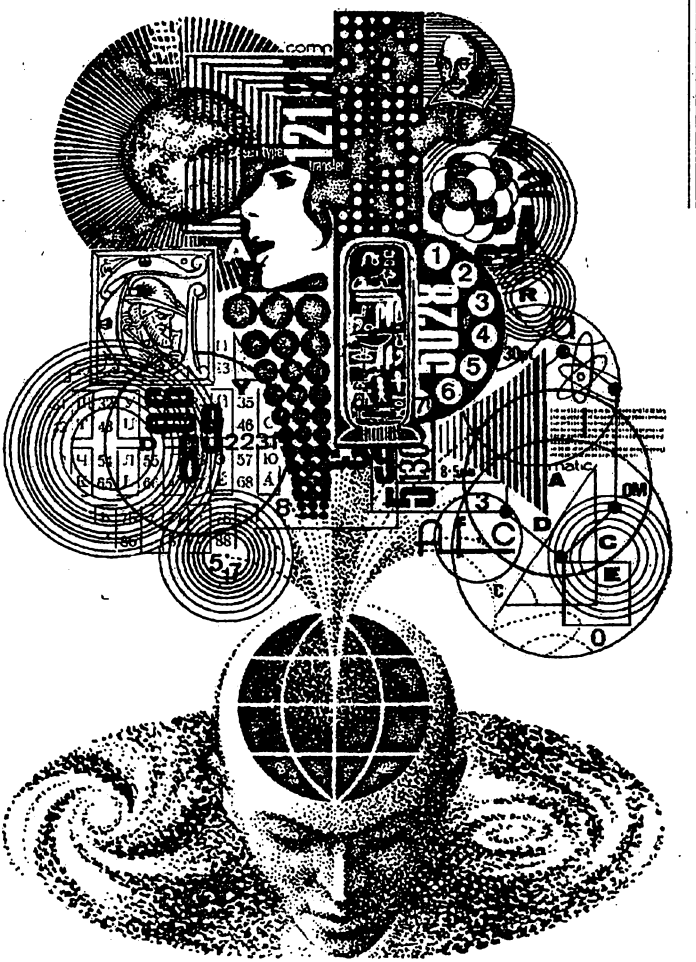
Свои замечания и пожелания автору шлите на адрес редакции с пометкой "Окно в мир увлекательной информатики".

Желаем вам приятной работы и творческих успехов.

Ноябрь 1996 г. ***Михаил Коляда***

Глава 1

ОБЩЕЕ ЗНАКОМСТВО С ЭВМ



"Кибернетика подобна тому, что в средние века считали "черной магией", — она может дать все, что попросите, но она не может подсказать, что нужно просить".

Норберт Винер

§ 1. Что такое информатика?

В конце XVI века Джиан Доменико Кампанелла написал книгу "Город Солнца". У каждого из четырех единодушно избранных народом главных "правителей города Солнца" имелась библиотека всего лишь из одной книги под названием "Мудрость", где удивительно сжато и доступно изложены все науки.

Как просто: открой "Книгу мудрости" и на любой вопрос найдешь ответ. Автор утопического романа понимал нереальность подобных идей. Но он глубоко верил в силу человеческого разума, способного в будущем открыть лучшие пути приобщения людей к знаниям, как ни велик бы был их запас.

В надеждах своих Доменико Кампанелла не ошибся. Фантазия писателя приобретает зримые черты. Человек создал "Книгу мудрости" – **электронные вычислительные машины (ЭВМ).**

Аббревиатура ЭВМ прочно вошла в нашу речь, хотя этот термин неточно передает ее сущность. Способность производить вычисление – далеко не основное предназначение современных ЭВМ. С их помощью решаются самые разнообразные задачи обработки информации. Вычислительный характер носят лишь внутренние физические процессы информационно-логических преобразований в ЭВМ. Это отражено и в понятии "компьютер" (англ. *computer* – считаю, вычисляю). Популярность этого термина обусловлена его удобством для образования новых понятий: компьютеризация, компьютерная грамотность и др.

В современном понимании **компьютер** – это информационная машина, универсальный электронный инструмент для разнообразной обработки информации – обработки данных, редактирования текстов, обеспечения диалогового режима общения с человеком и др.

Стремительное развитие и широкое распространение вычислительной техники послужили предпосылками к появлению нового раздела науки, названного информатикой. Слово это появилось в начале 60-х годов во французском языке для обозначения автоматизированной обработки информации в обществе.

Информатика (от фр. *information* – информация и *automatique* – автоматика) – отрасль науки, изучающая структуру и общие свойства научной информации, а также вопросы, связанные с ее сбором, хранением, поиском, переработкой, преобразованием, распространением и использованием в различных сферах человеческой деятельности.

Однако с середины 70-х годов термин "информатика" стал использоваться как синоним английского словосочетания computer science (наука о вычислениях) для обозначения научной дисциплины, связанной с обработкой с помощью ЭВМ информации любой природы.

Однако сегодня информатика не только научная и учебная дисциплина. Можно с полным основанием утверждать, что она превратилась в динамично развивающуюся отрасль народного хозяйства.

§2. Общая схема устройства ЭВМ

Электронные вычислительные машины (компьютеры) – это универсальные электронные устройства обработки и накопления информации.

Какой бы из современных компьютеров мы ни рассматривали, все они, за небольшим исключением, имеют общую принципиальную схему, или, как говорят, архитектуру (рис. 1).

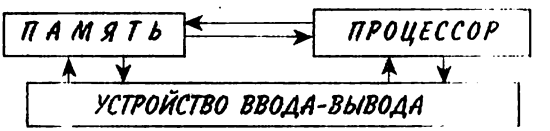


Рис. 1

Эта схема была предложена в 1946 году одним из гениальных создателей первых ЭВМ америка́нцем Джоном Фон Нейманом.

Процессор – это своего рода "мозг" ЭВМ. Он руководит совместной работой всех устройств. Процессор прежде всего осуществляет всевозможные операции над числами – операции сложения, вычитания, умножения и деления над целыми и вещественными числами. По этой причине ЭВМ и называют вычислительными машинами. Кроме того, процессор может выполнять различные преобразования символов и пересылать их по линиям связи с одних устройств ввода-вывода на другие.

В **памяти** компьютера хранятся программы и обрабатываемая информация для текущего и будущего использования. Память хранит информацию, передает ее на обработку процессору и принимает от него полученную в результате обработки новую информацию. Вся память ЭВМ по особенностям организации и использования подразделяется на **внутреннюю** и **внешнюю**.

Основную часть внутренней памяти образуют **оперативное запоминающее устройство (ОЗУ)** и **постоянное запоминающее устройство (ПЗУ)**.

ОЗУ служит для хранения информации во время ее непосредственной обработки. Эта память необходима

Джон Фон Нейман (1903-1957)

Американский ученый, инициатор строительства современных вычислительных машин. Родился в Будапеште, работал в Германии, а затем по приглашению Принстонского университета выехал в Соединенные Штаты Америки, где и остался навсегда.

В период нарастания угрозы мировой войны он начал работу по применению математики и физики в разработке военной техники. Эта проблема потребовала огромной вычислительной работы, что и стало основной причиной интереса Неймана к вычислительным машинам.

Он модернизировал и усовершенствовал первую в мире ЭВМ, а затем построил новую машину "Джаниак". Разработки проекта машины были основаны на некоторых факторах работы человеческого мозга. Нейман специально изучил неврологию и психиатрию и пришел к убеждению, что электронные машины могут строиться, как упрощенные модели человеческого мозга.

процессору для решения той или иной задачи в данный момент времени. В ней хранится текст обрабатываемой им инструкции (программы). Сюда помещаются полученные в процессе вычислений промежуточные и окончательные результаты. После выключения ЭВМ информация в ОЗУ пропадает.

Основное достоинство ПЗУ состоит в том, что содержащаяся в нем информация не уничтожается при выключении питания компьютера. Однако это качество влечет за собой и большой недостаток: ее нельзя изменять. Информация заносится в ПЗУ только один раз – при его изготовлении. Процессор лишь читает ее с высокой скоростью, однако изменять и дополнять не может. В связи с этим ПЗУ сравнивают со складом, с которого можно только получать что-либо. Обычно в ПЗУ заносят инструкции по запуску компьютера.

Внешняя память используется в компьютерах для длительного хранения больших объемов информации (программы и данные к ним), которые невозможно обеспечить с помощью устройств внутренней памяти. Устройство внешней памяти (магнитные ленты и диски) можно по практической ценности сравнить с используемыми человеком бумажными носителями информации: книгами, справочниками, конспектами и др.

Устройство ввода и вывода обеспечивает ввод информации в память ЭВМ и выдачу ее наружу, т.е. обмен информацией с внешним миром.

Различают **внешнюю архитектуру ЭВМ** – это то, что видят люди, которые используют машины для своих це-

Известно ли вам, что...

емкость человеческой памяти колеблется в широких пределах. Средняя величина, которой часто пользуются, лежит в пределах 10^{12}–10^{15} бит. Много это или мало? Например, информация, содержащаяся во всей Большой советской энциклопедии (БСЭ), равна примерно 4×10^8 бит, что составляет ничтожную часть этой величины. Во всем книжном фонде Российской государственной библиотеки в Москве информации 10^{13} бит, то есть она могла бы уместиться в памяти одного (просто одного!) человека.

лей, и **внутреннюю архитектуру ЭВМ** – это то, из чего состоит машина и на чем основаны накопление, обработка и передача информации внутри машины.

§ 3. Архитектура ЭВМ

Внешняя архитектура

Сейчас наиболее широкое распространение получают персональные ЭВМ (сокращенно ПЭВМ), предназначенные для индивидуальной работы.

Персональные ЭВМ – это вычислительные машины, которые могут быть установлены на любом рабочем месте (например, на письменном столе).

В состав профессиональных персональных ЭВМ входят (рис. 2):

1. Устройство отображения информации.
2. Клавиатура с системным блоком.
3. Устройство записи информации.
4. Устройство печати.
5. "Мышь".

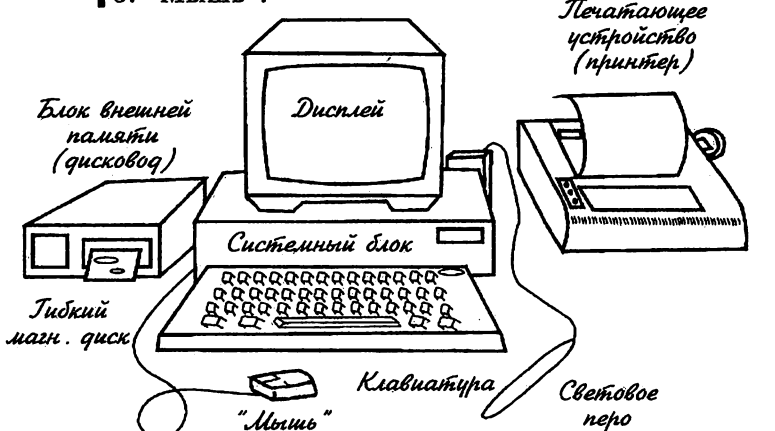


Рис. 2

В качестве устройства отображения информации в персональных ЭВМ применяют специальные **дисплеи** (от англ. *display* – показывать, воспроизводить), или их называют

мониторы (телевизоры). В этих устройствах информация выводится на экран электронно-лучевой трубки в виде двухцветных или многоцветных изображений: текстов, чертежей, схем, рисунков и т.п.

Клавиатура предназначена для передачи информации от человека к ЭВМ и похожа на клавиатуру обычной пишущей машинки (клавиши латинского и национального (обычно русского) алфавита, а также цифры, специальные знаки и другие функциональные клавиши).

В **системный блок** входят программируемое устройство управления (процессор) и внутренняя память (ОЗУ и ПЗУ).

В качестве **устройства записи информации** используют внешние магнитные носители: магнитофоны, специальные накопители, которые называют дисководами (НГМД – накопители на гибких магнитных дисках и НЖМД – накопители на жестких магнитных дисках, их по-другому называют "винчестерами"), проигрыватели лазерных компакт-дисков – CDROM (*compact disk only memory* – компакт-диск только для чтения – см. § 53).

Устройство печати, или **принтер** (от англ. *print* – печать): его назначение полностью указано в его названии – быстро и точно распечатать информацию на бумаге. К устройствам печати относятся и разного рода **плоттеры, графопостроители** и т.п. Эта техника объединяется под названием **устройства получения твердых копий**.

"Мышь" – это устройство для передачи информации от человека к ЭВМ. "Мышь" представляет собой небольшую коробочку с шариком на донышке, которую человек может перемещать по плоскости стола и которая при этом перемещении посылает сигналы в ЭВМ. С помощью этого устройства можно, например, вводить в ЭВМ картинки. Для этого достаточно положить картинку на стол и обвести ее "мышкой".

Кроме того, ЭВМ может содержать устройства ввода и вывода для получения информации от разного рода датчиков и подачи сигналов управления различным манипуляторам, роботам и другим **исполнителям**.

Например, ввод графической информации может проводиться с экрана дисплея с помощью **светового пера**, подсоединенного к ЭВМ. Внутри пера находится фотодиод, который может реагировать на пучок электронов, испус-

каемых электронно-лучевой трубкой. Сигнал о фиксации пучка поступает в блок управления световым пером, оттуда в компьютер, который передает в блок управления дисплеем сигнал о выводе светящейся точки в том месте, где находится световое перо.

Внутренняя архитектура ЭВМ

Связь и обмен информацией между компонентами ЭВМ осуществляется с помощью магистрали.

Магистраль – это общая линия проводов, к которой параллельно подсоединяют все компоненты ЭВМ. Посылая по магистрали электрические сигналы, любая компонента ЭВМ может передавать информацию другим компонентам (или другой машине по каналу).

Под компонентами мы понимаем различные устройства (их еще называют модулями) (рис. 3).

Подсоединяя к магистрали разные наборы модулей, можно получать различные ЭВМ. Такой магистрально-модульный принцип построения ЭВМ получил сейчас широкое распространение, так как обладает важными достоинствами:

1. Процессор управляет вашими устройствами с помощью тех же команд, которыми он работает с памятью (т.е. не нужны специальные команды).

2. Можно подключать к магистрали новые внешние устройства. При этом не требуется никаких изменений в уже существующих устройствах, процессоре, памяти.

3. Из готовых модулей можно легко составлять ЭВМ разной мощности и назначения. Состав ЭВМ можно легко изменять в процессе эксплуатации.

Рис. 3

Проверьте свои знания ?

1. Какие элементы входят в общую схему ЭВМ? Кем она была предложена?
2. Каково назначение процессора?
3. Какие виды памяти вам известны?
4. Каково назначение устройства ввода-вывода?
5. Что подразумевают под внешней архитектурой ЭВМ?
6. Что такое магистраль?
7. В чем суть магистрально-модульного принципа ЭВМ?

§ 4. Первый раз в дисплейном классе. Правила работы

Вы пришли в дисплейный класс. Прежде всего нужно ознакомиться с правилами техники безопасности при работе с ЭВМ:

1. Будьте внимательны, дисциплинированы, осторожны. Точно выполняйте указания преподавателя.
2. Не держите на рабочем месте предметы, не требующиеся при выполнении задания.
3. Перед выполнением работы внимательно изучите ход ее выполнения.
4. Не включайте ЭВМ без разрешения преподавателя.
5. Включение ЭВМ производите последовательно.
6. Не прикасайтесь к экрану дисплея, не трогайте провода.
7. По окончании работы последовательно отключите ЭВМ.
8. Во время работы ЭВМ не перемещайтесь по классу, избегайте резких движений.

Основной особенностью компьютера как партнера является жесткая ограниченность его языка и форм ведения диалога с пользователем (т.е. с вами). Все процессы обработки информации и ведения диалога четко определены и запрограммированы заранее. Это накладывает ограничения на язык общения с компьютером, делает недопустимыми отклонения, а тем более ошибки при вводе информации.

Прежде чем приступить к решению задач на компьютере, пользователю необходимо приобрести хотя бы элементарные навыки общения с ним, научиться правильно оперировать с клавиатурой, усвоить символику входного языка компьютера, особенности ввода и вывода информации.

Следует помнить, что устойчивые навыки общения формируются не сразу. Они складываются в процессе многократного выполнения практических действий, решения разнообразных задач. Однако сократить период вхождения в устойчивую работу с компьютером, избежать множества повторений типовых ошибок поможет внимательное изучение материалов, представленных в данной главе.

Работа с компьютером требует от пользователя **собранности, точности, постоянного внимания и самодисциплины.** Во многих случаях приходится часами отлаживать программу из-за допущенных в ней погрешностей. Специалисты правильно говорят, что при работе с компьютером мелочей нет, здесь и малые и большие ошибки одинаково значимы. Поэтому для начинающих утверждение "лучше медленно, но правильно" должно стать золотым правилом. Скорость и уверенность в работе придут по мере накопления опыта.

Использование персональных ЭВМ состоит в основном в работе на клавиатуре и чтении информации либо восприятии рисунков на экране дисплея.

Клавиатура компьютера преобразует любой вводимый символ в комбинацию электрических сигналов, в машинный код, поступающий в память машины. Ошибочно введенный символ можно оперативно удалить с экрана нажатием специальной клавиши, и это не повлияет на ход работы. Ошибочный ввод какой-либо команды может привести к необратимым последствиям.

Клавиатура каждого компьютера обычно имеет свои конструктивные особенности, форму и обозначения клавиш, различия в их расположении. Однако общей чертой множества вариантов клавиатуры является наличие в них одинаковых по назначению функциональных зон (полей). Количество таких зон может быть различным, но чаще всего их бывает четыре.

Клавиши, относящиеся к одной зоне, располагаются вместе и имеют одинаковый цвет. Центральное положение занимает самая многочисленная группа светлых кла-

виш стандартной клавиатуры пишущей машинки (**алфавитно-цифровое поле**). По обе стороны от нее вертикально размещаются стандартные **управляющие клавиши** более темного цвета; многие из них также имеются в пишущих машинках. В крайней правой группе скомпонованы светлые клавиши, которые служат другим целям: во-первых, они могут выполнять роль **цифровой клавиатуры** калькулятора и, во-вторых, с их помощью можно управлять положением курсора на экране (иногда зона управления курсором выделена в отдельном поле). Крайнюю левую группу (иногда она располагается крайним горизонтальным рядом) образуют клавиши, называемые **функциональными** или **программируемыми**; их назначение может меняться программным способом.

Важной особенностью алфавитно-цифровой зоны является то, что ее клавиши имеют двойные обозначения и несут в процессе работы двойную нагрузку. Каждая клавиша служит для ввода двух символов (знаков, букв), что обусловлено алфавитом входного языка компьютера, состоящего из русских (или других национальных) и латинских букв и множества специальных знаков.

Для обеспечения однозначности ввода того или иного символа используются регистровые клавиши "рус/лат" ("shift") – русский-латинский алфавит, "вр/нр" – верхний-нижний регистр, клавиша фиксации режимов "фикс" ("Alt") и клавиша забоя "зб" ("delete", "rubout", "erase"). Очень важной является клавиша, имеющая вид изогнутой стрелки, ее называют "вк" – возврат каретки (или клавиша ввода – "Enter").

Что может компьютер

Библиотека в кармане

Любители кроссвордов используют самые разные словари: толковые, специализированные по областям наук, частотные, орфографические, словари иностранных слов, энциклопедические, атласы. Все эти справочники – солидные, с широким корешком, иногда в нескольких томах, с трудом умещающиеся на книжной полке. Потому пользоваться ими во время прогулки по парку, на солнечном пляже или в городском транспорте не всегда удобно.

При вводе любого символа следует прежде всего обратить внимание на место обозначения на клавише и в соответствии с этим предварительно нажать нужную регистровую клавишу. Только после этого можно вводить символ нажатием клавиши с его обозначением. Нарушение указанного порядка ввода – причина наиболее частых ошибок, допускаемых начинающими.

Ряд символов русского алфавита (А, О, М, Т и другие) совпадают по своему начертанию с латинскими. В то же время каждый из этих символов по-разному воспринимается машиной на русском и латинском языках.

§ 5. Азы общения с компьютером

Работа с ЭВМ всегда начинается с ее включения: включения накопителя, монитора, клавиатуры. О включении ЭВМ можно судить по горящим лампочкам – светодиодам или по заставке и сообщениям, появляющимся на ее экране. Существует общее правило включения и выключения питания ЭВМ:

"Включайте внешние узлы ЭВМ в любом порядке, но делайте это до того, как включите системный блок".

Относительно выключения вычислительной системы справедливо обратное правило:

"Сначала выключите системный блок, а затем выключайте внешние узлы в любом порядке".

Приобретя карманный "процессор слов", разработанный одной из американских фирм, страстные кроссвордисты не только смогут всегда иметь при себе емкий словарь, вмещающий тысячи слов, но и получат в свое распоряжение хорошего помощника, который в считанные секунды найдет нужное слово по нескольким произвольным буквам, запомнит новые слова, внесет изменения в свой архив. В специализированном компьютере размером всего десять на восемь и толщиной два сантиметра спрятано 4,5 мегабайта памяти для хранения слов и два специализированных микропроцессора для поиска и обработки текстовой информации.

Теперь разработчикам "процессора слов" стоит, наверное, подумать над созданием карманного "генератора кроссвордов" для обеспечения своего детища работой.

Это правило нацелено на сохранение самых важных (и дорогостоящих) внутренних элементов компьютера – микросхем – от воздействия экстратоков, возникающих именно в момент включения-выключения электропитания.

Через несколько секунд после включения машины и дисплея ЭВМ начинает выводить данные на экран. Первое, что появляется на экране, мерцающая подчеркивающая черточка. Это – **курсор**. Он указывает, в каком месте экрана появится очередной символ. Курсор может принимать и иную форму, в том числе форму прямоугольника, квадрата, стрелки, руки и т.д.

Запуск нужных программ производится методом поиска, а затем загрузки в память ЭВМ с помощью определенных программ. Предварительно эти программы должны быть записаны на кассете магнитной ленты или дискете. Для загрузки программы эти носители информации должны быть поставлены на соответствующее устройство – на магнитофон или дисковод на ЭВМ преподавателя или учащегося.

Работа пользователей на персональных ЭВМ обычно проходит в форме диалога. Человек набирает команды, нажимает на определенные клавиши, вводит слова, фразы и т.п., а ЭВМ в ответ на вводимые команды выводит сообщения, рисунки и выполняет заданные команды.

Мы не будем приводить конкретных примеров диалога человека с ЭВМ, эти задания на уроке вам предложит преподаватель. Скажем только, что при вводе вами неправильных команд ЭВМ выводит на экран определенные сообщения.

Важным способом организации диалога и ознакомления с возможностями программ являются различного рода "меню".

МЕНЮ – это перечень команд, режимов и других возможностей, предоставляемых ЭВМ при использовании данной программы.

Возможен случай, когда сам компьютер обучает пользователя работе с собой. При затруднениях пользователю достаточно нажать клавишу со словом "Помощь" ("Help"), и компьютер на экране даст справку-подсказку и на при-

мерах покажет, как можно или следует поступить в сложившейся ситуации.

Примером такой программы может быть программа "клавиатурный тренажер".

Приведем еще несколько рекомендаций, которые тоже можно отнести к "золотому фонду" правил начинающего пользователя (рис. 4):

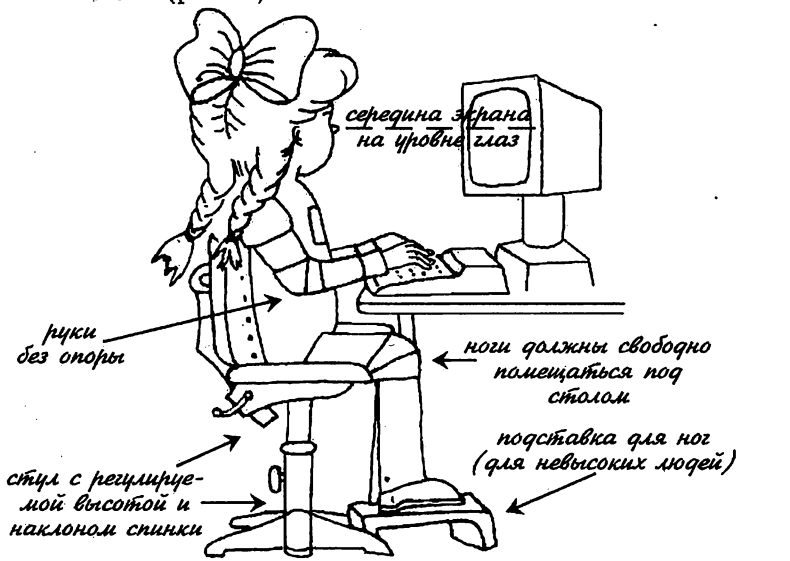


Рис. 4

- перед компьютером нужно сидеть свободно без напряжения. Это достигается с помощью специального стула с регулируемой высотой и наклоном спинки;
- в случае, если такого нет в наличии, постарайтесь сами занять такую позицию, чтобы она удовлетворяла вышеуказанному пункту;
- ноги должны свободно помещаться под столом;
- для невысоких людей необходимо под ноги подложить подставку;
- середина экрана должна находиться на уровне глаз, перпендикулярно линии взора, на расстоянии 55-65 см от глаз;

- руки лежат свободно, без опоры;
- клавиатура располагается на одном уровне с локтями;
- нажимать на клавиши необходимо "мягко", без особых ударов;
- время касания пальцами клавиш должно быть коротким, с таким расчетом, чтобы на экране от нажатия появлялась одна буква, а не целый ряд букв.

Самая распространенная "болезнь" начинающих пользователей – это использование для печати на клавиатуре одного-двух пальцев. Мы постараемся предостеречь вас от этой болезни.

На большинстве клавиатур пишущих машин и компьютеров клавиши расположены, как показано на рис. 5. Часто используемые буквы находятся в центре клавиатурного поля, а редко используемые – на периферии.

Чтобы понять, какой палец на какую клавишу нужно ставить, достаточно посмотреть на рис. 5 и на рис. 6. Например, центральный ряд клавиш нужно нажимать средними пальцами.

Клавиатура разделена на две половины, чтобы показать, какие клавиши нажимать правой рукой, а какие левой.

Если пальцы установлены на клавиши среднего ряда клавиатуры, то говорят, что пальцы находятся в основной позиции.

Начиная печатать, поставьте пальцы в основную позицию и возвращайтесь в нее после нажатия любой другой клавиши.

Клавиши регистров нажимайте мизинцем руки, противоположной той, которой вы нажимаете основную клавишу.

Пробел нажимается одним из больших пальцев. С самого начала **старайтесь печатать вслепую**, т.е. не глядя на клавиши. В этом случае нужно смотреть только на экран или бумагу, контролируя правильность печати. Клавищи, которые вы нажимаете реже (например, с цифрами и знаками препинания), расположены слишком далеко от основных клавиш, которые можно нажимать, не глядя на клавиатуру, поэтому при выборе удаленных клавиш можно смотреть, где они находятся. Обычно цифра нажимается пальцем, который находится под ней в основной

позиции. Мизинцем пользуются для набора ближайшего к нему знака препинания. Для более удаленных клавиш пользуйтесь наиболее удобным для вас пальцем.

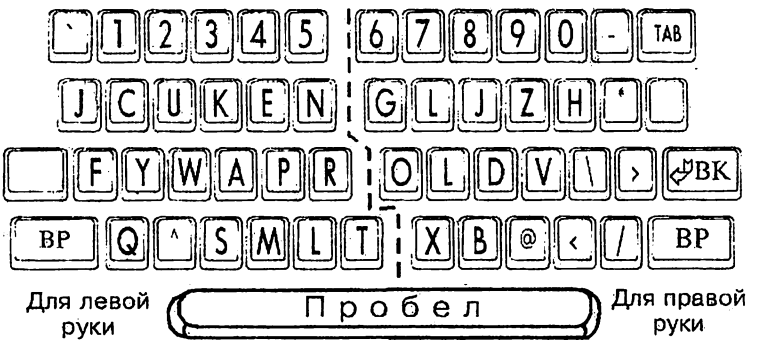


Рис. 5

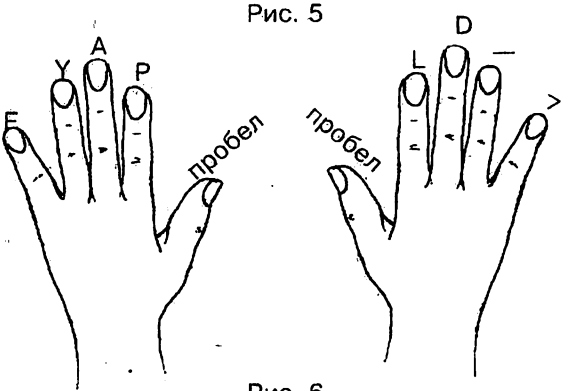


Рис. 6

Когда вы хорошо освоите этот метод, то сможете перепечатывать материал очень быстро, т.е. со скоростью чтения материала.

Запомните: лучше вначале правильно научиться ставить пальцы, чем потом переучиваться! Переучиваться всегда труднее, чем учиться!

Чтобы научиться печатать вслепую, нужно много тренироваться. Самое главное – это привыкнуть ставить определенные пальцы на определенные клавиши.

На уроке вам будут предложены рациональные примеры и упражнения для усвоения печати вслепую. Желаем успеха!

Проверьте свои знания

1. Назовите правила техники безопасности при работе с ЭВМ.
2. Назовите известные вам функциональные зоны клавиатуры компьютера.
3. Каковы общие правила включения и выключения ЭВМ?
4. Что такое курсор? Зачем он нужен?
5. Что такое меню?
6. Назовите основные правила начинающего пользователя ЭВМ.
7. Что значит "печатать вслепую"?

Тренировка

I. Вы впервые печатаете на компьютере. Напечатайте свою фамилию. Для этого смотрите на клавиатуру и нажимайте клавиши с нужными буквами.

II. Перепечатайте это слово, но уже не глядя на клавиатуру. Делайте это до тех пор, пока ваша попытка не увенчается успехом.

III. Теперь, также не глядя на клавиатуру, попытайтесь напечатать свою фамилию и имя.

Не забывайте возвращать пальцы в основную позицию, нажав другую клавишу. Пробел вводите большим пальцем.

IV. Попытайтесь напечатать фразу: "Раз-два-три-четыре-пять!".

Вначале не обращайте внимание на скорость печатания: она увеличится по мере осваивания вами клавиатуры. Сосредоточьте усилия на правильной постановке пальцев.

V. Продолжите печатать стихотворение З.Александровой:

"...Будем пальчики считать,
Крепкие, дружные,
Все такие нужные."

Перед тем как нажать клавишу со знаком препинания, бросьте беглый взгляд на клавиатуру.

VI. Не глядя на клавиатуру, попытайтесь напечатать все буквы русского алфавита от "А" до "Я". Сначала напечатайте все строчные буквы, разделяя их пробелом, затем – все прописные буквы.

VII. Попробуйте напечатать ответы, не глядя на клавиши:

1. Имя?
2. Адрес?
3. Дата рождения?
4. Рост?
5. Вес?
6. Цвет волос?
7. Цвет глаз?
8. Любимая еда?
9. Любимый напиток?
10. Любимая телевизионная программа?
11. Любимое занятие?
12. Мечта?

Когда вы научитесь печатать эти ответы вслепую, попробуйте освоить более трудные упражнения:

VIII. Представьте себе, что основные клавиши, как магниты, притягивают ваши пальцы в основную позицию.

Сначала отпечатайте шесть рифмующихся слов:

FOX	BOX	SOKS	LOCKS	CLOCKS	HNOCKS
ЛИСА	КОРОБКА	НОСКИ	ЗАМКИ	ЧАСЫ	ПРИДИРКИ

IX. Чтобы вспомнить, где находится нужная клавиша и каким пальцем ее нужно нажать, смотрите на схему клавиатуры (рис. 5), а не на саму клавиатуру.

Теперь, не глядя на клавиатуру, попытайтесь напечатать все буквы латинского алфавита от "A" до "Z" (сначала строчные, затем прописные).

X. Напечатайте предложения, состоящие из всех букв алфавита:

"The quick brown fox jumps dver tne lazy dog."

"Six squirrels were keenly jumping as the badger from the nearly zoo came into view."

XI. Попробуйте заполнить анкету, используя латинский алфавит:

1. Имена всех своих друзей.
2. Все, что вы ели вчера.
3. Дни недели.
4. Месяца года.
5. Ваша любимая поп-группа.
6. Все цвета радуги.
7. Мебель в комнате, в которой находитесь.
8. Все места, где вы были на каникулах.
9. Все, что надето на вас сейчас.

XII. Ниже приводится отрывок из рассказа, автором которого является французская вычислительная машина "Ка-

леоппа". Попытайтесь переписать его вслепую. Расставьте знаки препинания:

"Мой горизонт состоит лишь из красной портьеры откуда с перерывом исходит удушливая жара. Едва можно различить мистический силуэт женщины гордой и ужасной эта знатная дама должно быть одна из времен года. Кажется она прощается. Я больше ничего не вижу и продвигаюсь к занавесу который мои руки судорожно раздвигают. Вот по ту сторону странный трагический пейзаж: циветта скребет землю птицы летают с обеих сторон садятся на ветви деревьев наполовину иссохших. А тут и черепаха застывшая неподвижно почувствовала мое присутствие. Но почему она покрыта инеем. Мальчик подбегает его пухленькие руки его серьезное и смуглое лицо придают ему вид молодого героя."

ПОДВЕДЕМ ИТОГИ

☝ **Информатика** в школе – это учебный предмет, направленный на вооружение учащихся знаниями и навыками использования современной вычислительной техники.

☝ **Компьютер** – это универсальное электронное устройство, необходимое для поиска, сбора, хранения, преобразования и использования информации в самых различных сферах человеческой деятельности.

☝ Принципиальная схема ЭВМ была предложена Джоном Фон Нейманом, которая включает в себя три основных элемента: **процессор, память, устройство ввода-вывода.**

☝ В понятие **архитектуры** ЭВМ входят как представляемые машиной возможности реализации алгоритмов, так и средства, позволяющие задавать нужное использование этих возможностей.

☝ Различают **внешнюю архитектуру** ЭВМ – это то, что видят люди, которые используют машину для своих целей, и **внутреннюю архитектуру** ЭВМ – это то, из чего состоит машина и на чем основано накопление, обработка и передача информации внутри машины.

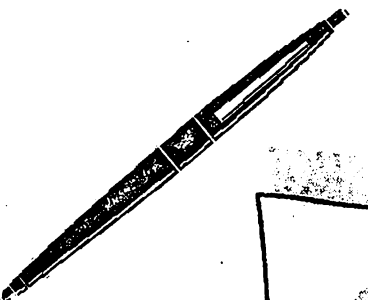
☝ Работа с компьютером требует от человека **точности, внимания** и **самодисциплины**. В общении с ЭВМ мелочей и погрешностей не должно быть: малые и большие ошибки одинаково значимы.

Глава II

ПРЕДСТАВЛЕНИЕ И ИЗМЕРЕНИЕ ИНФОРМАЦИИ В ЭВМ

"Кибернетика, которая сравнивает очень длинные цепи упорядоченных процессов, выбирая оптимальный результат, всегда будет иметь своей главной целью преобразование человеческого труда".

Б.Г. Кузнецов



§ 6. Что такое информация?

Проснувшись утром, мы сразу попадаем в мир информации: смотрим – получаем информацию; слышим – получаем информацию; берем предметы, едим, нюхаем – тоже получаем информацию. Информацию нам несут окружающие нас предметы и приборы: книги, журналы, газеты, телевизор, радио и т.п.

Общение людей друг с другом дома и в школе, на работе и на улице – это передача информации: сведений и суждений, данных и сообщений (рис. 7).

С самых древних времен человек окружен информацией. Потребность выразить и передать информацию привела к появлению речи, письменности, изобразительного, музыкального искусства, вызвала к жизни книгопечатание, почтовую связь, телеграф, телефон, радио и телевидение. Почему же об информации заговорили так настойчиво только теперь, после появления ЭВМ?

Дело в том, что ЭВМ дала человеку совершенно новые возможности для поиска, накопления, передачи и обработки этой информации.

В "докомпьютерное" время в термин "информация" вкладывали отражения предметного реального мира.

Нельзя сказать, что с появлением ЭВМ это положение перестало выполняться. Просто оно сузилось до определенных рамок. Теперь "информация есть все сведения, являющиеся объектом хранения, передачи и преобразования" или, по-другому, "способность воспринимать что-либо, сохранять в памяти, передавать это в сигналы, направляющие деятельность в соответствующую сторону".

Ученые не боятся признаться, что они "полностью игнорировали человеческую оценку информации".

Канал связи в ЭВМ бездушен. Бездушен не только потому, что это "мертвое тело", "неодушевленная система", потому, что ему безразлично, что он передает: радость или печаль, сообщение о рождении или смерти. Передающей системе важно одно – передать нужное количество информации.

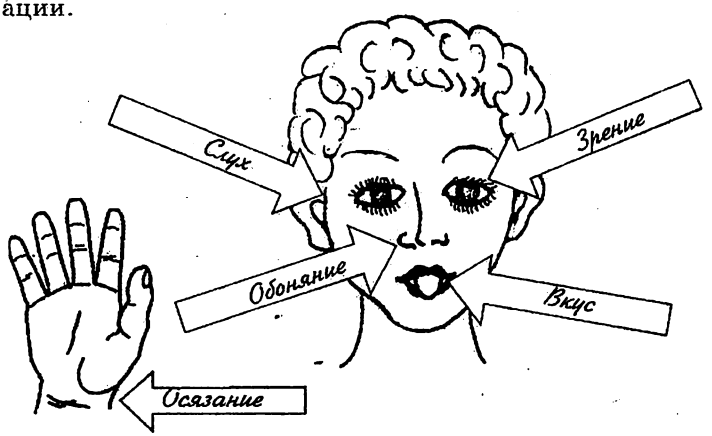


Рис.7

§ 7. Количество информации

Оценка количества информации основывается на законах теории вероятностей. Это и понятно. Сообщение имеет ценность, несет информацию только тогда, когда мы узнаем из него об исходе события, имеющего случайный характер, когда оно в какой-то мере неожиданно, ведь сообщение об уже известном никакой для нас информации не несет.

Если вам, допустим, кто-то позвонит по телефону и скажет: "Солнце всходит утром", – то такое сообщение вас удивит лишь своей нелепостью, а не новостью, которую оно содержит.

Иное дело, например, результат финала в шахматном турнире на первенство мира. Кто выигрывает: Карпов или Каспаров? Или партия закончится вничью? Исход здесь имеет случайный характер.

Чем больше интересующее нас событие имеет случайных исходов, тем ценнее будет сообщение о его результате, тем больше информации оно несет.

Сообщение о событии, у которого только два одинаково возможных исхода, содержит одну единицу информации, называемую **битом** (binary bigit – двоичный разряд).

Выбор такой единицы количества информации не случаен. Он связан с наиболее распространенным способом ее кодирования при передаче и обработке на ЭВМ.

§ 8. Единицы количества информации. Объем памяти

Для измерения длины, массы, температуры, времени, напряжения и т.д. придуманы приборы и введены единицы измерения соответственно: метр, килограмм, градус, секунда, вольт (в международной системе единиц СИ).

А как узнать количество информации, обрабатываемой ЭВМ, в каких единицах эту информацию измерять?

В настоящее время в качестве измерителя количества информации приняты следующие единицы: бит, байт (в системе СИ).

Биты и байты используются также для измерения "емкости" (объема) памяти.

Скорость передачи (быстродействие) измеряется количеством передаваемых бит в секунду (например, 1000 бит/с).

БИТ – наименьшая (элементарная) единица количества информации, соответствующая одному разряду двоичного кода (о кодах см. ниже). Один бит – количество информации, содержащейся в сообщении с двумя возможными равновероятными исходами типа "да" – "нет", что в двоичном коде равнозначно 1 – 0.

БАЙТ – основная единица количества информации в компьютерной технике, соответствующая восьми разрядам двоичного кода: 1 байт = 8 бит.

За 1 байт принимается количество информации в сообщении об одном из 256 возможных равновероятных собы-

тий. То есть одному байту равна информация об одном из возможных 256 символов, которые могут быть использованы в ЭВМ для ввода, хранения и переработки информации (почему 256 – читайте в § 10). Байт записывается в памяти машины, читается и обрабатывается обычно как единое целое, т.е. имеет свойства логической и числовой неделимости. По своей практической значимости байт и бит могут быть сравнены с рублем и копейкой в нашей денежной системе.

Наряду с битами и байтами для измерения количества информации в двоичных сообщениях используются и более крупные единицы:

1 кбит (один килобит) = 2^{10} = 1024 бит (приближенно равно 1 тыс. бит);

1 Мбит (один мегабит) = 2^{20} = 1024x1024 бит = 1048576 бит (приближенно равно 1 млн. бит);

1 Гбит (один гигабит) = $2^{30} \approx 10^9$ бит (≈1 миллиард бит);

1 кбайт (один килобайт) = 2^{10} = 1024 байт (≈1 тыс. байт);

1 Мбайт (один мегабайт) = 2^{20} = 1024x1024 = 1048576 байт (≈1 млн. байт)

1 Гбайт (один гигабайт) = 2^{30} = 1024 Мбайт (≈1 миллиард байт).

К единицам измерения многих физических величин мы привыкли, и нам не нужно пояснять, что такое один метр или одна секунда. Мы хорошо представляем скорость автомобиля, равную 20 м/с (72 км/ч) или площадь в

Клод Шеннон

Американский математик и инженер. Клод Эльвуд Шеннон родился в 1916 г. в городе Гейлорд в США. Разрабатывал теорию релейных и переключательных схем. Своими исследованиями Шеннон заложил основы теории информации и в значительной степени определил развитие общей теории дискретных автоматов, теории вероятностных схем и теории управляющих систем, т.е. почти всего того, что входит в понятие кибернетика.

4 кв.м, а бит, байт, килобайт, мегабайт, гигабайт – много это или мало?

1 байт – это количество информации об одном символе (букве, цифре, знаке). Если учесть, что одна страница нашего учебника, напечатанного через два интервала, содержит чуть меньше 50 строк, в каждой строке – примерно 60 знаков (сюда входят и пробелы между словами), то количество информации (без учета ее смысла) в одной странице можно приближенно принять равным 2,5 кбайт. Если учебник содержит 400 страниц, то его информационный объем будет равен 1 Мбайт. Если ОЗУ компьютера имеет емкость 64 кбайт, то в нем можно разместить информацию в 26 машинописных страниц.

Емкость магнитных лент составляет примерно 300 кбайт. Емкость гибких магнитных дисков – от 500 кбайт до 2 Гбайт. На одной кассете магнитной ленты можно записать около 120 страниц текста, на гибких дисках – от 200 до 800 страниц текста. В частности, для хранения этого учебника требуется один магнитный диск объемом в 1,4 Мбайта (включая служебную информацию, обслуживающую компьютер).

Можно привести и другие сравнения. В одном номере четырехстраничной газеты содержится информации приблизительно в 150 кбайт. Если учесть, что газета выходит пять раз в неделю, а в году 52 недели, то информация годовой подписки такой газеты составит примерно 40 Мбайт.

Учеными подсчитано, что за всю свою жизнь каждый из нас может оси-

Известно ли вам, что...

Многие считают, что глаза читающего плавно скользят по строчкам текста. На самом же деле за час непрерывного чтения в среднем 57 минут глаза читающего находятся в полном покое и только 3 минуты уходят на движение зрачков.

...Способность человека усваивать информацию (точнее, его скорость ввода информации в память) составляет в среднем 25 бит/с, или около одного слова в секунду. Скорость же работы современных ЭВМ перешагнула рубеж сотен миллионов бит в секунду.

лить 6-8 тысяч томов книг. В перерасчете на единицу количества информации, это составит 3-4 гигабайта.

Обычно информацию больших объемов в ЭВМ хранят на накопителях с жесткими магнитными дисками. Такие диски имеют объем памяти до 2 Гбайт. Они позволяют создавать в ЭВМ всевозможные картотеки, архивы документов и базы данных.

Особо большие объемы информации позволяют накапливать оптические диски (видеодиски). Объем памяти на этих дисках составляет от 100 Мбайт до 1 Гбайта. Считывание информации с таких дисков производится маломощным лазерным лучом (поэтому их по-другому называют "лазерные диски"). Объем памяти таких дисков позволяет записывать двухчасовой видеофильм (до 100 тысяч изображений).

Проверьте свои знания ?

1. Что вы понимаете под словом "информация"?
2. В каких единицах измеряют количество информации?
3. Что такое бит и байт? Как они связаны между собой?
4. Приведите объемы памяти известных вам носителей информации.

Тренировка

1. Посчитайте, сколько бит содержат следующие выражения:

а) VELE, VIDE, VICI
(пришел, увидел, победил – Ю.Цезарь);

б) NIL VOLENTI DIFFICILE EST
(ничего нет трудного, если есть желание);

в) NIHIL HUMANI A ME ALIENUM ESSE PUTO
(ничто человеческое мне не чуждо – Теренций).

II. Сколько бит в одном гигабайте?

III. Средняя скорость чтения учащихся 9-11 классов составляет 160 слов в минуту. Подсчитайте, сколько байт информации вы успеете переработать за семь часов непрерывного чтения? Сколько это страниц текста?

IV. Вы вводите в ОЗУ ЭВМ текст, читая его со скоростью 180 слов в секунду. Через сколько времени память ЭВМ в 32 кбайта будет полностью заполнена?

V. Школьная контролирующая программа занимает в ОЗУ компьютера 19 кбайт памяти. Инструкция пользователю в этой программе помещается в одном кадре дисплея размером 24 строки по 80 символов. Какую часть программы эта инструкция занимает?

§ 9. Десятичная система счисления

Изобретение десятичной системы счисления относится к главным достижениям человеческой мысли (наряду с алфавитным письмом). Без нее вряд ли могла существовать, а тем более возникнуть современная техника.

Появилась десятичная система, вероятно, в Индии. Выбор графических изображений для цифр, разумеется, не принципиален. Современные изображения цифр – простая стилизация древних арабских цифр. Марокканский историк Абделькари Боужибар считает, что арабским цифрам в их первоначальном варианте было придано значение в строгом соответствии с числом углов, которые образуют фигуры.

В самом деле, если посмотреть на нижеприведенную схему, это предположение кажется не лишенным глубокого смысла (рис. 8).

Так, единица создает лишь один угол, тройка – три, пятерка – пять и т.п. Нуль не образует никакого угла, поэтому он не имеет никакого содержания.

Рис. 8

Всем нам эта система знакома с первого класса. Мы знаем этот ряд чисел от 0 до 9. Возникновение десятичной системы счисления связывают со счетом на пальцах рук, практически не употребляемым. Выяснилось, что для повседневного счета была бы удобнее двенадцатиричная сис-

тема (в ней хорошо записывается треть и четверть). Были придуманы названия для дополнительных цифр и для круглых чисел (дюжина = 12 шт., грос – 12 дюжин). Но на двенадцатиричную систему люди не перешли, чтобы не переучиваться.

Все же десятичная система кажется нам и самой простой, и самой удобной. Обратимся к принципам ее организации.

Рассмотрим какое-нибудь число, например 2358765. Каждая из цифр в данном числе несет двойную информацию: во-первых, свое собственное значение – 2, 3, 5 и т.д., а во-вторых – место (позицию), которое занимает в записи числа (т.е. разряд). Такие системы счисления называются **позиционными.**

Разобьем наше число на разряды: 5 – в разряде единиц; 6 – в разряде десятков; 7 – в разряде сотен, т.е. наше число может выглядеть и так:

$$2x1000000 + 3x100000 + 5x10000 + 8x1000 + \\ + 7x100 + 6x10 + 5x1.$$

Занумеруем все разряды справа налево, причем привычный нам разряд единиц будем считать нулевым; тогда разряд десятков будет первым, сотен – вторым, тысяч – третьим и т.д. Такая нумерация весьма естественна, поскольку единица – это 10^0, десятки – 10^1; сотни – 10^2; тысячи – 10^3 и т.д., то есть расположение той или иной цифры в записи числа есть не что иное, как прямое указание, какой степенью 10 его можно заменить. А само значение цифры показывает, сколько раз надо взять 10 в заданной степени.

Таким образом, окончательно наше число запишется в следующем виде:

$$2x10^6 + 3x10^5 + 5x10^4 + 8x10^3 + 7x10^2 + 6x10^1 + 5x10^0.$$

Если число имеет дробную часть, то разложение добавляется суммой оснований 10 с отрицательными степенями. Например:

$$321,409 \quad 3x10^2 + 2x10^1 + 1x10^0 + 4x10^{-1} + 0x10^{-2} + 9x10^{-3}.$$

До сих пор некоторое употребление имеет римская система счисления. Это система **непозиционная**; скорее, ее

следует назвать **аддитивной**, поскольку число образуется при сложении и вычитании значений специальных значков.

I	V	X	L	C	D	M
1	5	10	50	100	500	1000

III	IV	VI	XL	LX	XC	CIX
3	4	6	40	60	90	109

MCMXCVI = 1000 + (1000 − 100) + 90 + 6 = 1996

Но выполнять арифметические действия с этими числами безнадежно. Попытайтесь решить примеры, приведенные ниже, и вы сами убедитесь в этом. Представления о десятичных (или иных позиционных) дробях в Древнем Риме не было.

$$\begin{array}{r} \text{LXVI} \\ +\ \text{LXVI} \\ \hline ? \end{array} \qquad \begin{array}{r} \text{LXVI} \\ \times\ \text{LXVI} \\ \hline ? \end{array} \qquad \begin{array}{r|l} \text{MCMXCIV} & \text{LXVI} \\ & \hline & ? \end{array}$$

В десятичной системе счисления считал знаменитый французский ученый Блез Паскаль, создавший первую вычислительную машину. Свое механическое счетное колесо он сделал десятичным: в нем было десять зубьев. С тех пор в десятичной системе счет можно было осуществить не только вручную с помощью 10 пальцев, но и механически – с помощью 10 зубьев колеса (см. с. 138).

Затем та же десятичная система "перекочевала" в электромеханические счетные машины. В них был применен шаговый искатель с десятью позициями.

И первые ЭВМ пользовались все теми же десятью "пальцами" – десятью триггерами (см. с. 168). На десятичной системе счисления работала, например, американская машина "Эниак". Но для нее требовалось столько дорогого оборудования, что конструкторы стали искать способы сократить число триггеров.

В основу своих поисков инженеры и математики положили двоичную (двухпозиционную) природу элементов, "органов" вычислительной техники.

Проверьте свои знания ?

1. Расскажите об одной из гипотез возникновения арабских цифр.
2. С чем связывают возникновение десятичной системы счисления?
3. В чем сущность позиционных систем счисления?
4. Приведите пример аддитивной системы счисления.
5. Могут ли существовать смешанные позиционные системы?

Тренировка

I. Представьте следующие десятичные числа в виде позиционной записи:
а) 576; б) 842,3; в) 1924,803; г) 10000; д) 0100,00; е) 0,002.

II. Имеются позиционные записи десятичных чисел:
а) $8x10^2 + 5x10^1 + 3x10^0 + 7x10^{-1} + 6x10^{-2}$;
б) $0x10^4 + 1x10^3 + 8x10^2 + 4x10^1 + 0x10^0 + 0x10^{-1} + 9x10^{-2}$;
в) $9x10^5 + 4x10^3 + 3x10^0 + 4x10^{-2} + 4x10^{-3}$.
Чему равны сами числа?

III. Переведите римскую запись в арабскую:
а) LX; б) XL; в) CXI; г) IXC; д) MDCCCXII; е) MCMLXI.

IV. Переведите арабскую запись чисел в римскую:
а) 45; б) 55; в) 900; г) 1500; д) 1554; е) 1917.

§ 10. Двоичная система счисления

В какой бы форме ни представлялась подлежащая обработке информация, она в конечном счете должна быть переведена компьютером на язык, доступный для автоматической обработки. **Язык компьютера** – это язык чисел, причем чисел не обычных (десятичных), а **двоичных**, алфавит которых состоит всего лишь из двух цифр – 0 и 1. Двоичная система наиболее проста и удобна для автоматизации. Наличие в системе всего лишь двух символов упрощает их преобразование в электрические сигналы.

Символы двоичной системы – 0 и 1 – можно передавать и записывать с помощью электрического тока. На-

пример, меняя продолжительность его протекания по цепи: коротко – точка, длиннее – тире, как в азбуке Морзе. Можно менять направление тока: плюс – минус. А можно менять амплитуду: есть сигнал – единица, нет сигнала – ноль. Последний способ потому применяется в вычислительных машинах, что он надежен, а отсутствие или появление сигнала легко различается в устройствах машины.

И новые машины стали "считать" с помощью 0 и 1.

Не думайте, что двоичная система – современница электронных машин. Нет, она намного старше. Двоичным счислением люди интересуются давно. Особенно сильным это увлечение было с конца XVI до XIX века. Знаменитый Г.В.Лейбниц считал двоичную систему простой, удобной и красивой. И даже по его просьбе была выбита медаль в честь этой "диадической" системы (так тогда называлась двоичная система). Как же получить запись числа в двоичной системе счисления? Вы уже догадались: нужно представить число как сумму степени двойки и выписать коэффициенты такого представления. При записи числа в различных системах счисления пользуются указателями оснований используемых систем. Это может быть справа внизу маленькая цифра или в конце буква латинского алфавита D, B, H, O:

D (desimal) – десятичный;

B (binary) – двоичный;

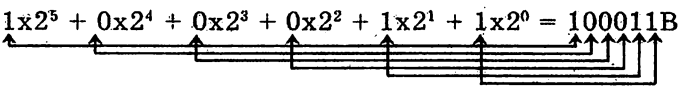
H (hexadcimal) – шестнадцатиричный;

O (octal) – восьмеричный.

Если вам встретится число 35 или 35D, то обе записи обозначают одно и то же: десятичное число – 35. Если же число 100011_2 или 100011B, то оба эти числа обозначают одно и то же двоичное число – 100011. Последнее читается так: "Один, ноль, ноль, ноль, один, один" в двоичной системе счисления.

Например:

$35D = 1x2^5 + 0x2^4 + 0x2^3 + 0x2^2 + 1x2^1 + 1x2^0 = 100011B;$



$58D = 1x2^5 + 1x2^4 + 1x2^3 + 0x2^2 + 1x2^1 + 0x2^0 = 111010B;$

$53{,}375D = 1x2^5 + 1x2^4 + 1x2^2 + 1x2^0 + 1x2^{-2} + 1x2^{-3} =$
$= 110101{,}011B.$

Двоичная система, как и обычная десятичная, по своей структуре относится к позиционным системам счисления. Значение любого числа определяется не только его разрядностью, количеством позиций, но также "весовым" значением и алфавитом системы счисления. Весовое значение позиций зависит от основания системы (т.е. разряда) (табл. 1):

Таблица 1

Весовые значения разрядов и коды чисел											Примеры десятичных чисел
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	
128	64	32	16	8	4	2	1	0,5	0,25	0,125	
				1	0	0	1				7
	1	0	1	1	1	0	1				13
		1	1	1	0	1	0				58
		1	1	0	1	0	1	0	1	1	53,375
1	1	1	1	1	1	1	1				255
								0	1	0	0,25
1	0	0	0	0	0	0	1				129

Для того чтобы перевести десятичное число в двоичное, достаточно знать структуру кода и весовые значения разрядов. Они возрастают справа налево в порядке возрастания номеров разрядов. Их легко запомнить: 1, 2, 4, 8, 16...

Пользуясь этой таблицей, нетрудно найти десятичный эквивалент любого двоичного числа разрядностью не более 8.

Поясним, как это делается на конкретном примере.

Десятичное число 7 представляется суммой трех весовых значений младших разрядов: 4, 2, 1.

$$7D = 4 + 2 + 1$$

В соответствии с этим обозначаем единицы в правых трех разрядах: 7D = 111B. Во всех остальных разрядах (в этом примере – впереди трех единиц) обозначаем нули, т.е. можно было записать: 7D= 00000111B. Это была бы эквива-

лентная запись (ноль всегда ноль!). Нетрудно убедиться, что число 13 в двоичной форме будет иметь вид 00001101.

Отметим, что наибольшее десятичное число, которое можно представить 8-разрядным двоичным числом, 256, а 16-разрядным – 65535 (соответственно $2^8 = 256$, а $2^{16} = 65536$). Следовательно, максимальный объем памяти не может превышать 65536 байт.

Точно так же кодируются и дробные числа. Для этого важно учитывать, что весовые значения разрядов дробной части числа уменьшаются после запятой в следующем порядке: 0,5; 0,25; 0,125; и т.д. Так, дробному числу 7,125 соответствует двоичный код 00000111, 0010000.

Подобно прямому преобразованию чисел (от десятичных к двоичным) выполняется и обратное преобразование. Искомое десятичное число получается как сумма весовых значений единичных разрядов. Например: двоичное число 111010 представляется в десятичной форме по таблице как сумма весов единиц разрядов:

1		1		1		0		1		0		
32	+	16	+	8	+	0	+	2	+	0	=	58D

Поскольку люди в научной и производственной практике пользуются десятичной системой счисления, то возникает задача поиска простых правил (а точнее, алгоритмов) для перевода чисел из десятичной системы в двоичную:

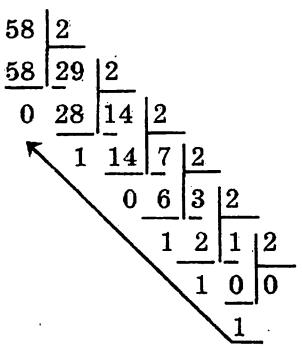
---

1) разделить число на 2. Зафиксировать остаток (0 или 1) и частное;

2) если частное не равно 0, то разделить его на 2, и так далее. Если частное равно 0, то записать все полученные остатки, начиная с первого, слева направо.

Пример: представить число 58D в двоичной системе счисления. Деление осуществляется столбиком так, как показано ниже.

Цифрами искомого числа, равного данному, являются все возникающие остатки. Появление в частном нуля означает конец процесса деления:



Остатки, начиная с последнего, выписанные слева направо, образуют искомое двоичное число:

$$58D = 111010B$$

Все операции прямого и обратного преобразования чисел осуществляются компьютером автоматически с помощью специальных функциональных узлов, в частности регистров, шифраторов и дешифраторов.

Основными воротами для ввода информации в компьютер является его клавиатура. Путем нажатия той или иной клавиши пользователь вводит в компьютер соответствующий знак, символ, букву. Но машина в состоянии "воспринять" только число, его двоичный код. Поэтому все клавиши в соответствии с их обозначениями и положением регистровых переключателей соединены с пронумерованными электрическими цепями таким образом, что ввод каждого символа будет равнозначен вводу определенного числа.

Одним из наиболее распространенных кодов является код ASCII (от англ. *American standart code for information interchage* – американский стандартный код для информационного обмена). В русском расширении этот код получил название АСКОИ – алфавитный код обработки информации. Этот код (его версия КОИ-8) принят в качестве стандарта.

Каждый символ в этом коде представляется восьмиразрядным двоичным числом (байтом). Всего существует 256 разных последовательностей из 8 нулей и единиц – это позволяет закодировать 256 разных символов, напри-

мер, большие и малые буквы русского и латинского алфавитов, цифры, знаки препинания и т.д. Соответствие байтов и символов задается с помощью таблицы, в которой для каждого кода указывается соответствующий символ. Вот фрагмент такой таблицы для кодировки КОИ-8:

Таблица 2

Код	Символ	Код	Символ	Код	Символ	Код	Символ
00100000		00110000	0	01000000	@	01010000	P
00100001	!	00110001	1	01000001	A	01010001	Q
00100010	"	00110010	2	01000010	B	01010010	R
00100011	#	00110011	3	01000011	C	01010011	S
00100100	$	00110100	4	01000100	D	01010100	T
00100101	%	00110101	5	01000101	E	01010101	U
00100110	&	00110110	6	01000110	F	01010110	V
00100111	'	00110111	7	01000111	G	01010111	W
00101000	(	00111000	8	01001000	H	01011000	X
00101001	)	00111001	9	01001001	I	01011001	Y
00101010	*	00111010	:	01001010	J	01011010	Z
00101011	+	00111011	;	01001011	K	01011011	[
00101100	,	00111100	<	01001100	L	01011100	\
00101101	-	00111101	=	01001101	M	01011101	]
00101110	.	00111110	>	01001110	N	01011110	^
00101111	/	00111111	?	01001111	O	...	

Коду 00100000 в этой таблице соответствует **пробел** – пустой промежуток в один символ, который используется для отделения одного слова от другого.

Если, например, пользователь на клавиатуре набрал слово "KIEV", то, учитывая соответствие каждой нажимаемой клавиши определенному коду, в память компьютера поступит следующая последовательность двоичных чисел:

§ 11. Арифметические операции в двоичной системе счисления

Правила выполнения арифметических операций с двоичными числами отличаются большой простотой.

Обратимся сначала к правилу счета в двоичном счислении:.

Поскольку имеются лишь две цифры, правило переноса формулируется так: когда в какой-либо разряд, в котором стоит 1 (самая старшая цифра), добавляется еще одна 1, этот разряд возвышается в 0 и посылает единицу переноса в следующий старший разряд. При непрерывном добавлении единиц в данный разряд его показания все время меняются на обратные: если в этом разряде стоял 0, он заменяется на 0 и при этом выдается единица переноса.

Давайте, руководствуясь этими правилами, сложим самые простые числа 1D + 1D:
записываем:

$$\begin{array}{r} 0001 \\ +\ 0001 \\ \hline 0010 \end{array}$$

Сложим теперь 7D + 4D.

Запись будет выглядеть так:

$$\begin{array}{r} 111 \\ +\ 100 \\ \hline 1011 \end{array}$$

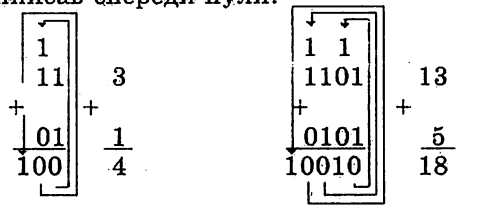
Сложение двоичных чисел проводится в соответствии со следующими правилами:

---

0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 0 и 1 переноса в следующий, старший разряд.

Приведем примеры сложения. Допустим, складываются двоичные числа 11B (3D) и 1B (1D), 1101B (13D) и 101B

(5D). В скобках показаны десятичные числа. Нули, записанные в старших разрядах, перед единицей не меняют значение числа, так же как в десятичной системе, например, 36=036 и т.д. Сначала уравняем количество разрядов у чисел, приписав спереди нули:



Можно проверить результаты, переведя двоичные коды в десятичные.

Вычитание выполняется по следующим правилам:

0 – 0 = 0
10 – 1 = 1 (единица занимается из соседнего, старшего разряда)
1 – 0 = 1 (единица является старшей значащей цифрой двоичного разряда)
1 – 1 = 0

Выполняя в заданном разряде вычитание из нуля единицы, следует занять единицу из старшего значащего разряда. В результате в младшем разряде образуются две единицы.

Проверим правильность выполнения предыдущих примеров:

Таблица умножения имеет очень простой вид:

0 × 0 = 0
0 × 1 = 0
1 × 0 = 0
1 × 1 = 1

Для умножения используем те же числа:

```
  11   3          1101   13
 x01  x 1         x 101   x 5
  11   3          1101    65
+00               + 0000
 011               1101
                  1000001=
```

=1000000(64)+1(1)=64+1=65.

Как видно из приведенных примеров, умножение двоичного числа на другое двоичное число сводится к последовательному сложению кодов множимого, сдвинутых относительно друг друга. Тем самым операция умножения сводится к двум другим операциям – сложения и сдвига.

Деление чисел в двоичной системе похоже на выполнение этой операции в десятичной системе. Оно сводится к последовательному вычитанию делителя из делимого. Приведем пример, позволяющий проверить полученный результат умножения чисел:

```
  ↓↓↓                          _65 | 13
  111                           65 | 5
_1000001 | 1101                  0
 110     | 101
 00011
    110
  _1101
   1101
      0
```

Проверьте свои знания ?

1. Сформулируйте правила сложения двоичных чисел.
2. По каким правилам выполняется вычитание двоичных чисел?
3. Можно ли утверждать, что операция умножения двоичных чисел сводится к операции сложения и сдвига?
4. Прочитайте таблицу умножения двоичных чисел.
5. К чему сводится операция деления чисел в двоичной системе счисления?

Тренировка

I. Произведите сложение двоичных чисел:

а) 111В + 101В;
б) 11011В + 1110В;
в) 0010001В + 1011101В;
г) 11111111В + 11111111В.

Сделайте проверку этих действий в десятичной форме счисления.

II. Выполните вычитания двоичных чисел:

а) 111В – 101В;
б) 11011В – 01110В;
в) 10011010В – 1100101В;
г) 10101010В – 01010101В.

III. Выполните действия:

а) (11011101В + 10101110В) – 1011111В;
б) (10001000В – 11001В) + 1100011В.

IV. Умножьте двоичные числа:

а) 111В х 101В;
б) 11011В х 1110В;
в) 100111В х 1001В;
г) 10101010В х 1010101В.

§ 12. Восьмеричная система счисления и связь ее с двоичной и десятичной

Основной недостаток двоичных чисел – высокая избыточность обрабатываемых чисел, громоздкость записи. Аппаратные средства ЭВМ накладывают известные ограничения на длину двоичных чисел. Вполне очевидно, что использование их для обработки или адресации становится невозможным. Особенно это проявляется при внешнем представлении числовой информации (вне ЭВМ). Поэтому в современных компьютерах помимо двоичной системы счисления применяют и другие, более компактные по длине чисел системы, в частности **восьмеричную**.

В восьмеричной системе счисления числа записываются с помощью восьми цифр:

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

сама восьмерка (как и двойка в двоичной системе) записывается совокупностью цифр "один" и "ноль" (10O, где буква O – обозначает восьмеричную систему счисления)

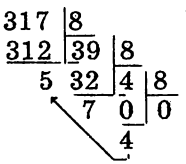
Примеры:

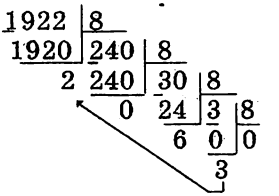
1) $502O=5x8^2 + 0x8^1 + 2x8^0 = 5x64+0+2=320+2=322D$;

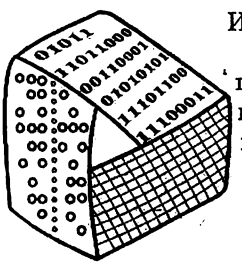
2) $3602O=3x8^3 + 6x8^2 + 0x8^1 + 2x8^0= 3x512 + 6x64+0x8+ 2x1=1536+384+0+2=1992D$.

Для замены десятичного целого числа на равное ему восьмеричное число используется алгоритм последовательного деления этого числа на 8.

Пример: *записать число 317 в восьмеричной системе счисления:*







Итак, имеем: 317D=475O; 1922D= =3602O.

Рассмотрим правило замены двоичного числа на равное ему восьмеричное, предварительно рассмотрев таблицу, в которой каждой восьмеричной цифре поставлено в соответствие трехзначное двоичное число:

Таблица 3

Восьмеричная цифра	Двоичное число	Восьмеричная цифра	Двоичное число
0	000	4	100
1	001	5	101
2	010	6	110
3	011	7	111

Удобство применения восьмеричной системы при работе с машинно-ориентированной информацией заключа-

ется в том, что переход от восьмеричной записи числа к двоичной осуществляется очень просто:

Каждую цифру восьмеричной записи следует заменить ее двоичным представлением (соответствующей двоичной триадой, т.е. трехразрядным числом):

Например: 1) $475O = \underbrace{100}_{4}\ \underbrace{111}_{7}\ \underbrace{101}_{5}B$;

2) $317{,}403O = \underbrace{011}_{3}\ \underbrace{001}_{1}\ \underbrace{111}_{7},\ \underbrace{100}_{4}\ \underbrace{000}_{0}\ \underbrace{011}_{3}B.$

Достаточно прост и обратный переход от двоичного представления какого-либо числа к восьмеричному.

Для этого в двоичной записи числа нужно выделить триаду (влево и вправо от запятой) и заменить каждую триаду соответствующей восьмеричной цифрой. В случае необходимости неполные триады дополняются нулями.

Например: 1) $1010110B = \underbrace{001}_{1}\ \underbrace{010}_{2}\ \underbrace{110}_{6}B = 126D$;

2) $11\ 110\ 100,\ 010\ 111\ 011\ 110\ 000\ 11\ B =$
$\underbrace{011}_{3}\ \underbrace{110}_{6}\ \underbrace{100}_{4},\ \underbrace{010}_{2}\ \underbrace{111}_{7}\ \underbrace{011}_{3}\ \underbrace{110}_{6}\ \underbrace{000}_{0}\ \underbrace{110}_{6}B = 364{,}273606O.$

§ 13. Шестнадцатиричная система счисления. Двоично-десятичный код

Назначение **шестнадцатиричной** системы счисления аналогично восьмеричной – для компактной записи двоичных кодов чисел и команд. В этой системе счисления данные (например, содержимое ячеек памяти – напомним, что это 8-разрядные двоичные числа) представляются уже в виде всего двухразрядных чисел, а адреса – в виде максимум четырехразрядных.

Для записи чисел в этой системе необходимо шестнадцать различных символов, используемых как цифры. В качестве первых десяти шестнадцатиричных 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) используются буквы латинского алфавита – A, B, C, D, E, F.

Переход от шестнадцатиричной системы к двоичной (и обратно) так же прост, как переход от восьмеричной к двоичной. Только на этот раз каждую шестнадцатиричную цифру нужно заменять соответствующей ей двоичной тетрадой. Неполные тетрады дополняются нулями. Разбивку производят для целой части числа – справа налево, а дробной – слева направо от запятой. Каждую из этих тетрад (групп) обозначают символом в соответствии с табл. 4:

Таблица 4

Десятичные цифры	Шестнадцатиричная запись	Двоичная запись	Десятичные цифры	Шестнадцатиричная запись	Двоичная запись
0	0	0000	8	8	1000
1	1	0001	9	9	1001
2	2	0010	10	A	1010
3	3	0011	11	B	1011
4	4	0100	12	C	1100
5	5	0101	13	D	1101
6	6	0110	14	E	1110
7	7	0111	15	F	1111

Примеры:

1) 15D=FH;
2) $31D=1x16^1 + 15x16^0 = 1FH$;
3) $167D = 10x16^1 + 7x16^0 = A7H$;
4) $6CH = \underbrace{0110}_{6}\ \underbrace{1100}_{C}B$;
5) $11111011111B = \underbrace{0111}_{7}\ \underbrace{1101}_{D}\ \underbrace{1111}_{F}\ B = 7DFH$.

Напомним, что буква "H" после числа обозначает шестнадцатиричную систему счисления.

Признаки системы счисления не указывают, если из текста статьи или смысла команд компьютера однозначно ясно, о какой из них идет речь. Но все-таки порой необходимость в этом возникает. Конечно, шестнадцатиричное число, содержащее буквы, не спутаешь с десятичным или двоичным. Но как, например, определить, что за число 100, если не названа система счисления, в которой оно написано? Ведь 100B при переводе в десятичную систему превратится в 4, 100H – в 256, а 100D – останется числом 100. Иногда к шестнадцатиричным числам добавляют в начале еще и цифру 0 (например, пишут не F800H, а 0F800H). Не вникая пока в детали, отметим, что это дополнительный признак шестнадцатиричного числа. Он существен при разработке программ на языке ассемблера.

Двоично-десятичный код предназначен для кодирования десятичных чисел последовательностью нулей и единиц. Для представления каждой из десяти цифр десятичной системы используют первые десять тетрад от 0000 до 1001 (см. табл. 4). Оставшиеся шесть возможных комбинаций нулей и единиц в четырех двоичных разрядах для двоично-десятичного кода являются запрещенными.

Наиболее часто двоично-десятичный код применяется для ввода в машину исходных данных, записанных в десятичной системе счисления. После ввода в ЭВМ двоично-десятичный код самой машиной переводится в двоичную систему счисления.

Правила перевода двоично-десятичного кода в десятичную систему и наоборот аналогичны переводу по тетрадам двоичных чисел в шестнадцатиричную систему счисления с учетом запрещенных комбинаций.

Пример:

1) $\underbrace{0101}_{5}\ \underbrace{0111}_{7}\ \underbrace{0011}_{3},\underbrace{0100}_{4}$ B-D = 573,4D;

2) 362,89D = $\underbrace{11}_{3}\ \underbrace{0110}_{6}\ \underbrace{0010}_{2},\underbrace{1000}_{8}\ \underbrace{1001}_{9}$ B-D.

ПОДВЕДЕМ ИТОГИ

Нередко приходится слышать о том, что пользователю компьютера нет необходимости знать внутренние механизмы обработки информации. Ведь пользуемся мы ежедневно, причем весьма успешно, многими техническими средствами информации – телефоном, радио, телевизором, – не вникая в принципы их действия, выполняя лишь общие правила эксплуатации. С подобной точкой зрения можно согласиться лишь частично. Действительно, для некоторых пользователей, выполняющих на компьютере элементарно простые операции, достаточной является форма общения с компьютером на уровне "черного ящика". Однако для подавляющего большинства лиц, использующих компьютер как инструмент для обработки разнообразной информации, как усилитель своего интеллекта, такой уровень явно недостаточен.

Поэтому повторим кратко сведения об информации и формах представления ее в ЭВМ.

Информация – сведения об объектах и явлениях окружающей среды, их параметрах, свойствах и состоянии. В информатике эти сведения называются данными.

Единицами количества информации являются:

Бит – цифра в двоичной системе счисления (0 и 1); двоичный разряд машинного слова.

Байт – часть машинного слова – поле из восьми последовательных бит. Байт используется для представления в ЭВМ кода одного символа. **Килобайт** (кбайт) содержит 1024 байта, **мегабайт** (Мбайт) – 1024 кбайта, **гигабайт** (Гбайт) – 1024 Мбайта.

Язык компьютера – это язык чисел, причем не обычных (десятичных), а двоичных, алфавит которых состоит всего лишь из двух цифр – 0 и 1.

Двоичная система наиболее проста и удобна для автоматизации. Наличие в системе всего лишь двух символов упрощает их преобразование в электрические сигналы. Так, символ 1 может передаваться импульсом напряжения, символ 0 – отсутствием импульса (паузой).

Десятичное число может быть представлено суммой последовательности весовых значений разрядов двоичного кода. При этом каждое из слагаемых сумм обозначается единицей соответствующего разряда кода. Недостающие слагаемые обозначаются нулями в разрядах кода.

Двоичная система счисления имеет и **недостатки**: ею пользуются только ЭВМ для внутренней, а не внешней работы.

Для компактной записи двоичных кодов чисел и машинных команд в различных периферийных устройствах и устройствах подготовки данных используют **восьмеричную** систему счисления.

Шестнадцатиричная система счисления используется как средство кодирования чисел при составлении адресов команд.

Двоично-десятичный код позволяет представить в ЭВМ каждую десятичную цифру от 0 до 9 тетрадой двоичного кода. Тем самым десятичное число заменяется поразрядно тетрадами двоичных чисел.

Таблица 5

Системы счисления, применяемые в ЭВМ

Система счисления	Основание	Алфавит	Примеры соответствия десятичным числам
Двоичная	2	1,0	A=11011; A=27
Восьмеричная	8	0,1,2,3,4,5,6,7	A=23; A=19
Шестнадцатиричная	16	0,1,2,3,4,5,6,7 A,B,C,D,E,F	A=2B A=43
Двоично-десятичная	2	1,0	A=01000101 A=45

Можно перевести число из одной системы счисления в другую. Например (см. табл.5): из десятичной системы чис-

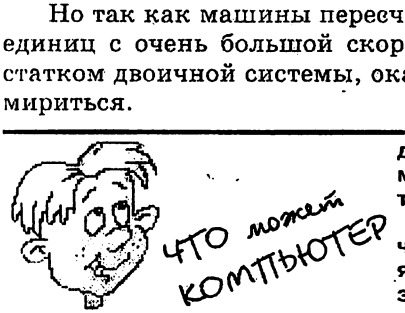
ло 43 переведем соответственно в двоичный, восьмеричный и шестнадцатиричный коды:

43D=101011B;
43D=53O;
43D=2BH.

И самое главное: из любой системы счисления можно перейти к двоичному коду. А, как известно, двоичная система счисления является основной системой представления информации в компьютере. Почти все ЭВМ используют либо непосредственно двоичную систему счисления, либо двоичное кодирование какой-либо другой системы счисления, например, десятичной (двоично-десятичный код).

Все четыре арифметических действия над двоичными числами сводятся к выполнению операций сложения, вычитания и сдвига. В свою очередь, вычитание двоичных чисел можно свести к сложению преобразованных двоичных кодов.

Но так как машины пересчитывают вереницы нулей и единиц с очень большой скоростью, то с главным недостатком двоичной системы, оказывается, совсем нетрудно мириться.



Умный компас

Если вы плохо ориентируетесь на местности, или решили исследовать запутанный лабиринт карстовых пещер, или просто любите собирать грибы – вашим проводником может стать "умный" компас.

В верхней части электронного прибора, умещающегося на ладони, расположен цифровой жидко-кристаллический дисплей, на который, в зависимости от установленного режима, выводится либо азимут вашего движения в градусах, либо время, в течение которого вы идете в выбранном направлении.

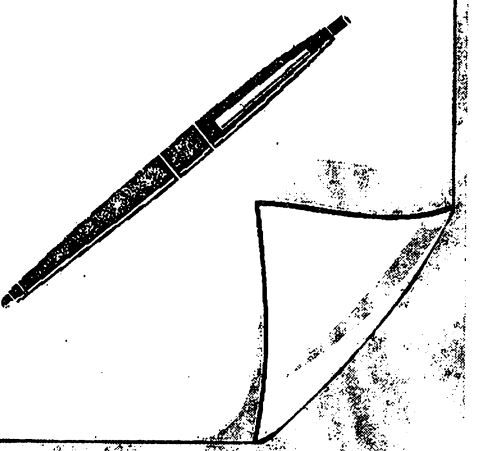
Но самым замечательным качеством электронного компаса является то, что он детально запоминает маршрут движения и может точно воспроизвести его в любой момент. Поэтому, если вы потеряли надежду найти и грибы, и обратную дорогу, обратитесь к компасу, и он напомнит вам весь ваш маршрут, который хранится в его памяти в виде последовательности пар чисел: азимут и время движения в эту сторону. Проделав указанный путь в обратном направлении, вы вернетесь в исходную точку. И пусть это будет не самая короткая дорога к вашему дому, зато она пройдет по уже знакомым вам местам.

Глава III

НАЧАЛА АЛГОРИТМИЗАЦИИ

"Ум человеческий имеет три ключа, всё открывающих: знание, мысль, воображение – всё в этом".

В. Гюго



В предыдущих разделах мы показали структуру и назначение устройств, из которых состоит компьютер, попытались выяснить, как представляется информация в ЭВМ. Но нами не был затронут самый важный вопрос: каким образом на ЭВМ удается решать различные сложные задачи, если она умеет выполнять лишь простейшие операции?

Главная особенность всех вычислений машины состоит в том, что в основе ее работы лежит **программный принцип** управления. Это означает, что для решения как самой простой, так и самой сложной задачи пользователю необходимо использовать перечень инструкций или команд, следуя которым шаг за шагом ЭВМ выдаст необходимый результат.

Таким образом, для того, чтобы решать задачу на ЭВМ, ее необходимо сначала, как говорят, алгоритмизовать. Именно алгоритмический принцип и лежит в основе работы всех ЭВМ.

Не нужно думать, что изучение и построение алгоритмов понадобится только тем учащимся, которые изберут профессию программистов. Умение выделять алгоритмическую суть явления и строить алгоритмы – очень важно для человека любой профессии.

Понятие алгоритма ценно не только практическим использованием, оно имеет также важное общеобразовательное и мировоззренческое значение. Навыки алгоритмического мышления способствуют формированию особого стиля культуры человека, составляющими которого являются: целеустремленность и сосредоточенность, объективность и точность, логичность и последовательность в планирова-

Что может компьютер

Дантист и компьютер

У стоматологов иногда возникает необходимость в определении формы прикуса зубов пациента. Для этого обычно применяется кусочек копировальной бумаги или воска. Но получившуюся картину неопытному врачу бывает порой трудно разобрать и совсем невозможно зафиксировать.

Справедливо решив, что всю малоприятную и трудоемкую работу должен делать компьютер, американский дантист В.Манесс изобрел датчик, который не только кодирует информацию о прикусе и записывает ее на диск компьютера, но и демонстрирует

нии и выполнении своих действий, умение четко и лаконично выражать свои мысли, правильно ставить задачу и находить окончательные пути ее решения, быстро ориентироваться в стремительном потоке информации.

Перед тем как приступить к изучению алгоритмов, необходимо определиться в очень важном понятии – **исполнитель**. В данной главе под исполнителем мы будем понимать человека-исполнителя, который сам разрабатывает алгоритм либо получает его в готовом виде и затем исполняет. В последующих главах под этим понятием мы будем иметь в виду компьютер (или другое устройство: автомат, робот), который может выполнять некоторый, вполне определенный набор действий.

§ 14. Понятие алгоритма

Понятие алгоритма не есть для нас что-то новое и необычное. Встречаются они в нашей повседневной жизни почти на каждом шагу. Так, утром мама перед вашим уходом в школу дает вам такую инструкцию: “Когда придешь со школы, сразу пообедай и не забудь вымыть посуду. После этого подмети пол, купи в магазине молоко и хлеб. Сделав покупки, погуляй часок и начинай выполнять домашнее задание”.

Эта инструкция состоит из последовательности отдельных указаний, которые определяют ваше поведение. Это и есть – алгоритм.

Каждый из нас, не задумываясь, ежедневно использует сотни различных алгоритмов. Например, правила сложения, вычитания, деления, умножения чисел; правила преобразования алгебраических выражений; грамматические правила правописания слов и предложений, а также различные инструкции и правила, рецепты и указания – все это алгоритмы.

Например, вы засняли пленку на фотокамере и собираетесь проявить ее. На упаковке проявителя вы найдете инструкцию по его применению:

"Содержимое малого пакета растворить в 350 мл кипяченой воды при температуре 18-20 градусов Цельсия. Время проявления фотобумаги 1-2 мин, фотопленок 3-4 мин."

Приведенная инструкция есть алгоритм. Она указывает на последовательность действий в приготовлении раствора.

Еще пример. Для приготовления омлета с сыром можно руководствоваться следующей последовательностью действий:

1) 50 г мякоти белого хлеба намочить в 3 столовых ложках молока и размять;

2) разбить в эту смесь 3 яйца, и все это хорошо взбить ложкой;

3) всыпать сюда 50 г тертого сыра;

4) посолить, перемешать и вылить на горячую сковороду с 1 столовой ложкой масла;

5) жарить на сильном огне, слегка помешивая.

Приведенный рецепт тоже относится к алгоритмам.

Много примеров алгоритмов имеется в школьных предметах. Но более

на его экране последовательность смыкания зубов и даже усилие их сжатия.

Датчик представляет из себя пластиковую пластину, внутри которой расположена серебряная сеточка. Сжимая датчик зубами, пациент уменьшает расстояние между ее проволочками, а компьютер по соответствующей программе отслеживает и обрабатывает поступающую с него информацию.

Теперь, если дантисту понадобится внимательно рассмотреть или обсудить с коллегами какую-нибудь фазу сжатия зубов пациента, он может обратиться к памяти компьютера. У самого пациента, рот которого свободен, можно уточнить кое-какие детали.

всего таких примеров в школьном курсе математики. Это и понятно: ведь математика и занимается, по существу, изучением различных алгоритмов и созданием новых.

Происхождение самого термина "алгоритм" тоже связано с математикой. Слово "алгоритм" появилось в результате искаженного перевода с арабского на европейские языки имени узбекского ученого IX века Аль-Хорезми, который изложил правила арифметических действий над числами в позиционной десятичной системе счисления (см. с. 209). Эти правила и назвали алгоритмами (Альхорезми [имя] + Аритмос [число] = алгоритм).

Таким образом, понятие алгоритма возникло значительно раньше появления ЭВМ. Но широкое распространение это понятие получило в эпоху автоматов и роботов.

Рассмотрим еще пример. Для того чтобы тронуться с места на автомобиле, необходимо выполнить ряд действий:

1) сядьте за руль в машину автомобиля;

2) рукой поверните ключ зажигания по часовой стрелке и запустите двигатель;

3) левой ногой нажмите на педаль сцепления, а правой – включите рукоятку (или кнопку) первой передачи движения;

4) одновременно медленно отпускайте левой ногой педаль сцепления, а правой – с той же силой нажимайте на педаль "ГАЗ";

5) вращая рулевое колесо вправо-влево, направляйте автомобиль в нужную сторону движения.

Из приведенных примеров ясно, что алгоритмы, алгоритмические процессы неотделимы от нас и являются составной частью нашей жизни. Почти вся человеческая деятельность связана с алгоритмами.

Это и правила для учащихся (записанные в уставе школы), это и требования воинских уставов и уставов различных общественных организаций и партий. Это наказы избирателей, инструкции по технике безопасности, схемы электронных устройств, чертежи деталей и т.д. и т.п.

А в информатике под АЛГОРИТМОМ понимают понятное и точное предписание исполнителю совершить последовательность действий, направленных на достижение указанной цели или на решение поставленной задачи.

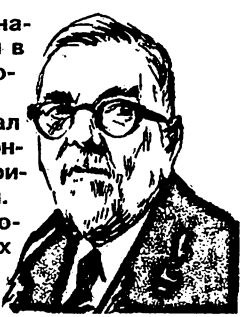
Алгоритм, как правило, формулируется в виде схемы или предложения (текста). Этот текст записывают на бумаге или вводят в память компьютера, используя специальные обозначения.

Понятие алгоритма в информатике является фундаментальным, т.е. таким, которое не определяется через другие, еще более простые понятия. Для сравнения вспомним, что в физике таким фундаментальным понятием является пространство и время, в математике – точка, в химии – вещество, поэтому приведенное выше определение не является всеобъемлющим, а смысл этого понятия будет уточняться нами по мере изучения всего курса.

§ 15. Свойства алгоритмов

Использование вычислительных машин в качестве исполнителей алгоритмов предъявляет ряд требований к алгоритмам. В отличие от людей, компьютер может выполнять только точно определенные операции. Поэтому

Норберт Винер (1894 – 1964)



Выдающийся американский математик, названный "отцом кибернетики". Родился он в Колумбии, работал в США. Отец его по происхождению поляк.

Он был человеком весьма одаренным – знал тринадцать языков. В четырнадцать лет окончил колледж, в восемнадцать стал доктором философии, а затем профессором математики.

Изучая аналогии между процессами, протекающими в электрических и электронных системах и в живых организмах, пришел к идее создания новой науки – кибернетики, которую он представлял как единую науку об управлении. Эта наука объединяет в себе достижения многих наук: математической логики, электроники, физиологии и общественных наук.

Созданная им новая отрасль науки – кибернетика – показала свои огромные возможности, стала необходимой во многих других отраслях науки и техники.

Известно ли вам, что...

существует способ запоминания (ввода информации в человеческую память), называемый гипнопедия. Вспомните знаменитый способ изучения иностранных языков по методу Илоны Давыдовой. Идея эта не нова. Так, еще в Древней Греции учителя нашептывали спящим ученикам то, что трудно усваивалось днем. Его используют при обучении курсантов телеграфному коду в военно-морской школе во Флориде и летчиков во Франции.

Казалось, что появился новый эффективный способ введения информации, даже не требующий расхода времени, т.к. используется во время сна. Однако более глубокие исследования показали, что запоминание возможно лишь в

машинные алгоритмы должны обладать следующими свойствами:

1. Чтобы исполнитель мог решить поставленную перед ним задачу, используя алгоритм, он должен уметь выполнить каждое его указание. Иными словами, он должен понимать суть управления. То есть при составлении алгоритма нужно обязательно учитывать "правила игры", т.е. систему предписаний (или систему команд), которые понимает ЭВМ. Например, при решении какой-то задачи студент использовал обращение к функциям sin x (это тригонометрическая функция) и к функции Бесселя (это цилиндрическая функция), но компьютер (как и читатель, наверное) не понимает последней. Она создателями данного класса машин не предусмотрена. Следовательно, алгоритма (в целом) машина не поймет. Мы будем говорить в данном случае о "понятности" алгоритма.

Под "ПОНЯТНОСТЬЮ" алгоритмов понимают указания, которые понятны исполнителю.

2. Будучи понятным, алгоритм не должен все же содержать предписаний, смысл которых может восприниматься неоднозначно.

Этими свойствами часто не обладают предписания и инструкции, которые составляются для людей. Например.

В приведенном выше рецепте приготовления омлета сказано: "Разбить в эту смесь 3 яйца и все это хорошо взбить ложкой". На бытовом уровне

нам понятно, что речь идет о трех куриных яйцах (а каких еще! – скажете вы). Но яйца могут быть и голубиные, и утиные, и даже страусиные (все резко отличаются по величине друг от друга). Здесь явно "закралась" неоднозначность.

Или указания типа: "посолить по вкусу", "насыпать две-три ложки сахарного песку", "получил оценку 4 или 5", "жарить до готовности", "копать от забора до обеда" не могут встречаться в алгоритмах. Очевидно, что понятные в определенных ситуациях для человека предписания такого типа могут поставить в тупик ЭВМ.

Или вспомним известную всем притчу о царской воле. Царь приказал подчиненным выполнить такой указ: "Казнить нельзя помиловать". Он забыл в указе поставить запятую, а подчиненные не знали, что им делать. Указание "казнить нельзя, помиловать" и "казнить, нельзя помиловать" задают совсем разные действия, от которых зависит жизнь человека.

Кроме того, в алгоритмах недопустимы такие ситуации, когда после выполнения очередного предписания алгоритма исполнителю неясно, какое из них должно выполняться на следующем шаге.

Под ОДНОЗНАЧНОСТЬЮ алгоритмов понимается единственность толкования правил выполнения действий и порядка их выполнения.

3. Как мы уже знаем, алгоритм задает полную последовательность дей-

наиболее поверхностные фазы сна, в период дремотного состояния. По мере углубления сна возможности введения информации резко снижаются. Таким образом, гипнопедия применима в течение довольно короткого отрезка ночи и составляет ту ее часть, когда, по существу, сна еще нет, когда Морфей только протянул свои руки, но еще не обнял нас.

• • •

Количество книг, которое может прочесть человек за всю свою жизнь, не превышает 6-8 тысяч. И это довольно высокая норма. Ее можно выполнить, читая ежедневно приблизительно по 50 страниц. За время "читающей" части жизни будет издано более 20 млн книг. Значит, в среднем можно прочесть только одну из 10000 книг. Выход – в увеличении "упаковки" информации – лежит только через использование компьютеров!

ствий, которые необходимо выполнять для решения задачи. При этом, как правило, для выполнения этих действий их расчленяют (разбивают) в определенной последовательности на простые шаги. Возникает упорядоченная запись совокупности четко разделенных предписаний (директив, команд), образующих прерывную (или, как говорят, дискретную) структуру алгоритма. Выполнить действия следующего предписания можно лишь выполнив действия предыдущего.

Именно программирование – это процесс разложения сложной задачи на ряд простых действий.

Под ДИСКРЕТНОСТЬЮ понимают возможность разбиения алгоритма на отдельные элементарные действия, выполнение которых человеком или машиной не вызывает сомнения.

---

4. Очень важно, чтобы составленный алгоритм обеспечивал решение не одной частной задачи, а мог выполнять решение широкого класса задач данного типа.

Например. Необходимо решить конкретное квадратное уравнение $x^2+ 4x+3=0$. Но ведь можно составить алгоритм решения любого квадратного уравнения вида: $ax^2+ bx+c =0$.

Действительно, для случая, когда дискриминант $D = b^2 - 4ac$ неизвестен, корни квадратного уравнения можно найти по формулам:

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

Если же $D<0$, то действительных корней не существует. Таким образом, этот алгоритм можно использовать для любого квадратного уравнения. Такой алгоритм будет **массовый**.

Часто приходится производить расчеты каких-либо величин при различных исходных данных.

Например. В физике для расчета общего сопротивления двух или более проводников, соединенных параллельно, используют формулу: $1/R=1/R_1+1/R_2+1/R_3+...+1/R_n$, где $R_1, R_2, R_3, ... R_n$ – данные сопротивления, а R – искомое общее сопротивление. Эта формула и есть алгоритм, поскольку она задает действие нахождения общего сопротивления.

Меняя исходные данные сопротивлений R_1, R_2, R_3,..., R_n и их количество, можно находить результаты для новых наборов данных.

И таких примеров универсальных алгоритмов можно привести множество. Все они должны иметь в своей основе функцию, которая каждому начальному значению ставила бы в соответствие результат. И выражается такая зависимость, как правило, формулой.

Под МАССОВОСТЬЮ алгоритмов подразумевается возможность их применения для решения целого класса конкретных задач, отвечающих общей постановке задачи.

---

5. Выполнение действий, заданных алгоритмом, состоит из конечного числа шагов. Это свойство называют конечностью алгоритмов.

Рассмотрим примеры, поясняющие это.

Пусть перед нами стоит требование преобразовать обыкновенные дроби в десятичные 7/4 и 5/3.

1) 7/4=1,75; 2) 5/3=1,6666666...

Для первого случая алгоритм задает конечную, а для второго – бесконечную последовательность действий. Таким образом, известное правило преобразования дробей для одних начальных данных удовлетворяет, а для других – не удовлетворяет свойству конечности. Обычно, чтобы удовлетворить это свойство, накладывают ограничения, т.е. делают преобразования не точно, а с какой-то погрешностью. Задают условие выполнения алгоритма, например, после получения 15 десятичных знаков результата. Тогда алгоритм, построенный таким образом, является выполнимым и заканчивается за конечное число шагов.

Следует заметить, что свойство конечности – это желательное, но не обязательное свойство алгоритмов. В противном случае нужно было бы признать, что многие широко распространенные правила, инструкции, в частности, упомянутое правило преобразования обыкновенных дробей в десятичные, не являлись бы алгоритмами, что было бы неоправданно с любой точки зрения.

Итак, под КОНЕЧНОСТЬЮ алгоритмов понимают завершение работы алгоритма в целом за конечное число шагов.

6. Еще к желательным свойствам алгоритмов нужно отнести **результативность**.

Она предполагает, что выполнение алгоритмов должно завершаться получением определенных результатов.

Подобные ситуации в информатике возникают, когда какие-либо действия невозможно выполнить. В математике такие ситуации называют неопределенностью. Например, деление числа на ноль, извлечение квадратного корня из отрицательного числа, да и само понятие бесконечности неопределенно. Поэтому, если алгоритм задает бесконечную последовательность действий, то в этом случае он также считается результатом неопределенным.

Но можно действовать по-другому. А именно: указать причину неопределенного результата. В таком случае, пояснения типа "на ноль делить нельзя", "компьютер выполнить такое не в состоянии" и т.п. можно считать результатом выполнения алгоритма.

Таким образом, свойство РЕЗУЛЬТАТИВНОСТИ состоит в том, что во всех случаях можно указать, что мы понимаем под результатом выполнения алгоритма.

И последнее общее свойство алгоритмов – их **правильность**.

Мы говорим, что алгоритм ПРАВИЛЬНЫЙ, если его выполнение дает правильные результаты решения поставленных задач. Соответственно мы говорим, что алгоритм СОДЕРЖИТ ОШИБКИ, если можно указать такие допустимые исходные данные или условия, при которых выполнение алгоритма либо не завершится вообще, либо не будет получено никаких результатов, либо полученные результаты окажутся неправильными.

Приведем пример.

Пусть A и B – два равных количества: A=B. Умножим обе части этого равенства на A:

$$A^2=AB.$$

Теперь уменьшим на B^2 и левую и правую части равенства. Получившиеся разности $A^2 - B^2$ и $AB - B^2$ тоже будут равными:

$$A^2 - B^2 = AB - B^2.$$

Разложим на множители:

$$(A + B)(A-B) = B(A-B).$$

Делим обе части равенства на $(A-B)$, после чего получаем:

$$A + B = B.$$

Так как $B = A$, то в последнем равенстве можем заменить B через A, тогда $A+ A = A$ или $2A = A$.

Разделив на 2, получим $A=A/2$, а это означает, что мы получили абсурдный вывод "целое равно своей половине".

Внешне, или, как говорят, "формально", все правильно, а по существу где-то в приведенных выкладках есть дефект. Предлагаем читателю самостоятельно найти изъян такого алгоритма.

Проверьте свои знания

1. Каким человеческим качествам способствуют навыки алгоритмического мышления?
2. Что такое исполнитель?
3. Расскажите о происхождении термина "алгоритм".
4. Объясните понятие алгоритма. Почему нельзя дать строгого определения алгоритма?
5. Каковы основные свойства алгоритмов?
6. Приведите примеры, подтверждающие свойства алгоритмов.
7. Что значит алгоритм правильный? Неправильный?

Тренировка

1. Опишите алгоритм поиска нужного слова в энциклопедическом словаре.

II. Какие действия вы бы добавили, чтобы выполнялся алгоритм измерения объема вилки с помощью мензурки?

1. Привязываю нить к вилке.
2. Наливаю воду в мензурку.
3. ?
4. ?
5. Смотрю, на сколько делений поднялся уровень воды.
6. Зная, что каждое деление мензурки равно 1 куб. см, определяю объем вилки.

III. Составьте алгоритм вычисления площади неправильного треугольника, если известны его стороны a, b, c.

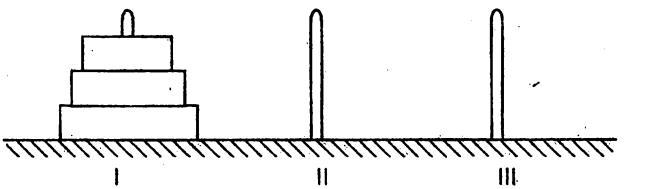
IV. Экспериментальная задача. Определите площадь прямоугольного стола, используя в качестве оборудования гирьку, часы и нитки. Напишите алгоритм, выполняя который можно решить эту задачу.

V. В своей книге "Арифметика" Леонтий Филиппович Магницкий привел следующий способ отгадывания задуманного двухзначного числа:

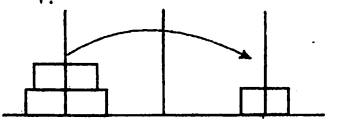
1. Увеличьте задуманное число десятков в 2 раза.
2. Прибавьте 5.
3. Сумму увеличьте в 5 раз.
4. К произведению прибавьте 10.
5. К полученной сумме прибавьте число единиц задуманного числа.
6. Из указанного результата вычтите 35 – получите искомое задуманное число.

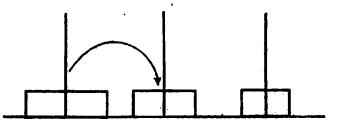
Почему так получается?

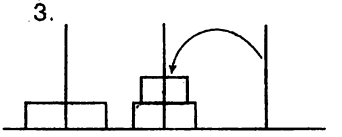
VI. Ханайская башня. На подставке укреплены три стержня, на левый стержень нанизано несколько колец уменьшающегося размера – внизу самое большое кольцо, на нем поменьше, сверху еще меньше и т.п. Надо, перемещая по одному кольцу со стержня на стержень, переместить все кольца на правый стержень, но при этом ни в какой момент времени большее кольцо на меньшее класть нельзя.

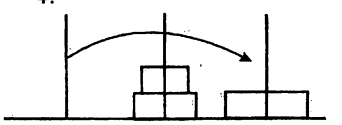


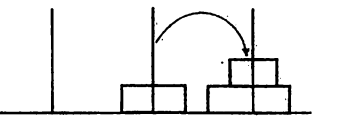
Ученик работает с тремя кольцами по следующему алгоритму (рис. 9):

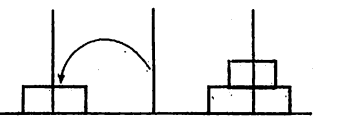
1.

3 →III

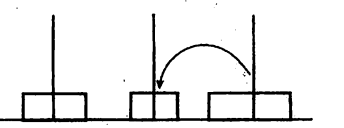
2.

2 →II

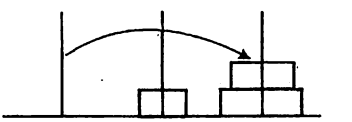
3.

3 →II

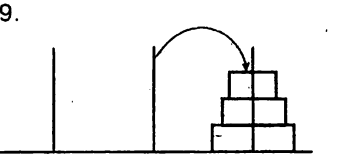
4.

1 →III

5.

3 → III

6.

2 → I

7.

3 → II

8.

2 → III

9.

3 → III

Оказывается, этот алгоритм не самый удачный. Найдите оптимальный алгоритм для перекладывания 3 колец, распространите это правило для 4, 5 и... N колец (где N – число колец на стержне).

Получите общую формулу расчетов.

Рис. 9

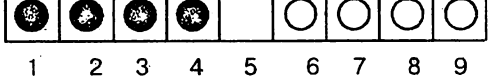
VII. Четыре белых шашки и четыре черных расположены так, как показано на рисунке. Требуется переставить белые шашки на клетки с номерами 1, 2, 3, 4, а черные – на клетки с номерами 6, 7, 8, 9 с соблюдением условий:

1. Каждая шашка может перескочить на ближайшую клетку или через одну клетку, но не дальше.

2. Никакая шашка не должна возвращаться на клетку, где она уже побывала.

3. В каждой клетке не должно быть более одной шашки.

4. Начинают ход белые шашки.

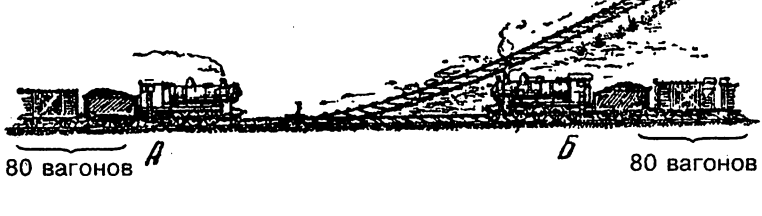


1 2 3 4 5 6 7 8 9

Составьте алгоритм перестановок.

VIII. Два поезда, каждый по 80 вагонов, встретились на одноколейном пути, имеющем небольшую тупиковую ветку (см. рисунок).

Как разойтись этим поездам, если тупиковая ветка может вместить локомотив с 40 вагонами, не более? Составьте алгоритм решения.



§ 16. Графический способ представления алгоритмов

Можно назвать три способа написания алгоритмов:

1) на естественном языке;

2) на языке схем;

3) на алгоритмическом языке.

Все алгоритмы к задачам и примерам в предыдущих параграфах были составлены как раз на естественном языке. Алгоритмическим языком, т.е. языком, ориентирован-

ным на ЭВМ, мы займемся позже. А сейчас разберем графический способ представления алгоритмов.

СХЕМОЙ называют наглядное графическое изображение алгоритма, когда отдельные его действия (этапы) изображаются при помощи различных геометрических фигур (блоков), а связи между этапами указываются при помощи стрелок, соединяющих эти фигуры.

Подобные схемы иногда называют **блок-схемами**. Они отображают шаги, которые должны выполняться компьютером, и последовательность их выполнения.

Для изображения различных шагов алгоритма на блок-схеме используют фигуры различной формы (табл. 6).

Начало и конец алгоритма изображаются с помощью овалов (табл. 6. 1). Внутри овала записывается "начало" или "конец".

Инструкции по выполнению каких-либо действий помещаются внутрь прямоугольников (табл. 6.2), а блоки выбора, определяющего путь, по которому пойдут эти действия (например, вычисления) дальше, в зависимости от результата анализа данных, изображаются в виде ромбов (табл. 6.3).

Само условие записывается внутри ромба. Если проверяемое условие выполняется, т.е. имеет значение "истинно", то следующим выполняется этап по стрелке "да". Если условие не выполняется (ложно), то осуществляется переход по стрелке "нет".

Известно ли вам, что...

первая электронно-вычислительная машина ЭНИАК предназначалась для решения задач баллистики. Она состояла из 18 тыс. радиоламп и 1,5 тыс. реле, за одну секунду производила 300 операций умножения или 5000 сложений многоразрядных чисел. Она была очень громоздкой: масса машины составляла 30 т, для ее размещения требовалась комната длиной 30 м. Потребляла она 150 кВт электроэнергии. Кроме того, много времени затрачивалось для набора и подготовки программ на коммутационной доске.

Ранее созданные и отдельно описанные алгоритмы и программы (а точнее, подпрограммы) изображаются в виде прямоугольника с боковыми линиями (табл. 6.4). Внутри такого "двойного" прямоугольника указываются имя алгоритма (подпрограммы), параметры, при которых он должен быть выполнен.

Ввод исходных данных и вывод результатов изображаются параллелограммом (табл. 6.5). Внутри него пишется слово "ввод" или "печать" и перечисляются переменные, подлежащие вводу или выводу.

Стрелками изображаются возможные пути алгоритма, а малыми кружочками (табл. 6.6) – разрывы этих путей (линий) потока на странице. Комментарии используются в тех случаях, когда пояснение не помещается внутри блока (табл. 6.7).

При записи вычислительных алгоритмов удобно использовать специальный **знак присваивания**: =. Просим не путать со знаком "=" (равно) в математике. Этот знак используется для изображения особой операции – операции присваивания. Поясним ее смысл. Пусть имеется предписание вида

Y=X (читается: "Y присвоить X").

Здесь Y – переменная, а X – некоторое выражение.

Предписание означает следующее: выполнить все действия, предусмотренные формулой X, и полученный результат (число) считать значением (т.е. присвоить) переменной Y.

В левой части команды присваивания всегда должна стоять переменная, а в правой – обычно стоит формула (переменная), но в частном случае может быть и число.

Например:

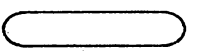
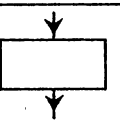
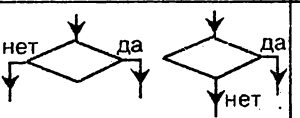
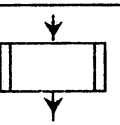
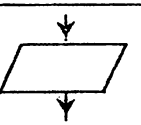
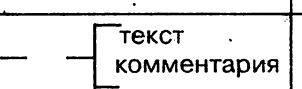
$$Y=K;\quad Y=37;$$

$$I=\frac{U}{R}\qquad Y=x^2+4x+3;\qquad x_1=\frac{-b-\sqrt{D}}{2a};$$

$$F=\begin{cases} 1, \text{ если } X<Y;\\ 0, \text{ если } X=Y;\\ -1, \text{ если } X>Y. \end{cases}$$

Таблица 6

Условные графические обозначения в схемах алгоритмов

Наименование	Обозначение	Пояснение
1. Пуск-останов		Начало, конец алгоритма, останов, вход, выход в подпрограмму
2. Процесс		Действие, вычислительная операция или группа операций
3. Решение		Разветвление в алгоритме, проверка условий
4. Предопределенный процесс		Программа, стандартная подпрограмма
5. Ввод-вывод		Ввод-вывод в общем виде
6. Соединители		Разрыв линий потока на странице, на разных страницах
7. Комментарий		Комментарий

В качестве примера попробуем по уже разобранному у нас алгоритму на естественном языке составить блок-схему.

Возьмем алгоритм для решения квадратного уравнения:

1. Начать.
2. Ввод a, b, c.
3. D присвоить $b^2 - 4ac$.
4. Если D<0, то идти к 6.
5. Если D>0, то идти к 8.
6. Действительных корней нет.
7. Идти к 10.
8. Вычислить: $x_1 = (-b - \sqrt{D})/2a$;
   $x_2 = (-b + \sqrt{D})/2a$.
9. Вывести значение x_1 и x_2.
10. Закончить.

Каждому оператору соответствует стандартный блок (забегая вперед, скажем, что эти условные операторы [1, 2, 3 и т.д.] заменяются конкретными командами языка, на который переводится алгоритм – см. § 30 и решения к тренировке VII):

а) оператор 1 – начало работы;
б) оператор 2 – ввод исходных данных;
в) оператор 3 – начальное присваивание;
г) операторы 4, 5 – проверка условия;
д) операторы 6, 8 – действия;
е) оператор 7 – линия пути выполнения;
ж) оператор 9 – вывод результатов;
з) оператор 10 – окончание работы.

В окончательном виде блок-схема алгоритма изображена на рис. 10.

Блок 4, 5 можно было бы проверить другим образом: D>0. В этом случае последующие блоки просто поменялись бы местами.

Понятно, что для каждого конкретного случая необходимо вводить свои коэффициенты квадратного уравнения a, b, c.

Известно ли вам, что...

покупают?

В общем объеме продаж программного обеспечения в Японии игровые программы занимают наибольшую долю: 30%. На втором месте – учебные программы (20%), затем идут обслуживающие (17%), деловые (13%) и научные программы (10%). Программы обработки текстов, хоть и являются обслуживающими, выделены в особую графу – их продажа достаточно велика: 6%. Оставшаяся доля рынка обозначена словом "прочие".

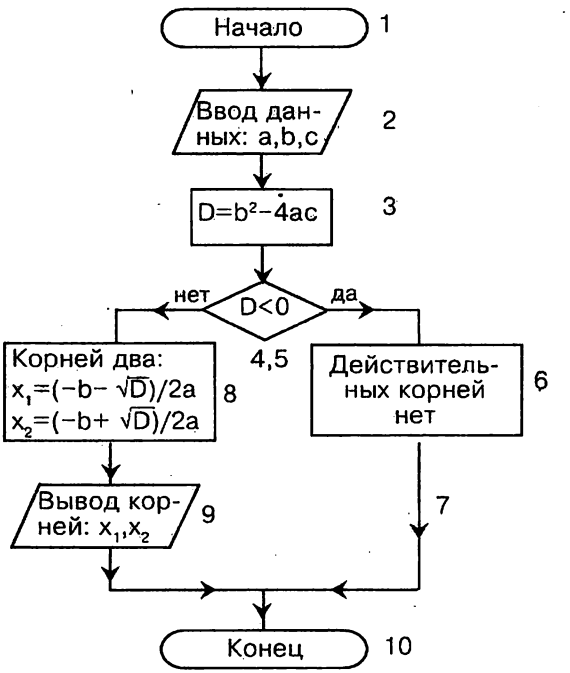


Рис. 10

А теперь покажем алгоритм вычисления наибольшего общего делителя (НОД) двух натуральных чисел M и N (алгоритм Евклида). Сначала запишем его на естественном языке: "Чтобы найти НОД двух чисел, составьте таблицу из двух столбцов и назовите их X и Y. Запишите первое из данных чисел в столбец X, а второе – в столбец Y. Если данные числа не равны, замените большее из них результатом вычитания из большего числа меньшего. Повторяйте такие замены до тех пор, пока числа не окажутся равными, после чего число из столбца X считайте искомым результатом".

А теперь то же самое запишем в виде структурных пунктов:

1. Начать.
2. Ввести натуральные числа M, N.
3. X присвоить значение M, а Y значение N.
4. Если X=Y, то идти к 10.

5. Если X>Y, то идти к 8.
6. Y присвоить значение Y–X.
7. Идти к 4.
8. X присвоить значение X–Y.
9. Идти к 4.
10. D присвоить значение X.
11. Ввести значение HOD: D.
12. Закончить.

Переложим алгоритм на язык блок-схемы:

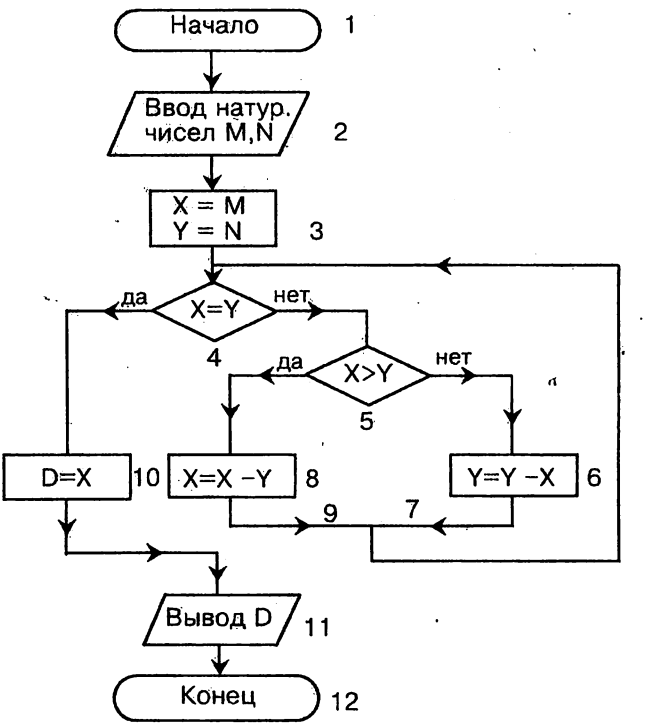


Рис. 11

Таким образом, блок-схема позволяет наглядно представить структуру алгоритма. Она как бы "по полочкам" раскладывает даже сколь угодно сложный алгоритм. Но иногда в графическом виде он становится громоздким и запутанным. Схемы обычно используются для изображения промежуточных вариантов алгоритма. Окончательный

вариант для исполнителя-ЭВМ (программа) должен быть записан на алгоритмическом языке. В настоящее время существует технология разработки алгоритмов и программ без использования схем. Однако независимо от этого на начальном этапе изучения программирования использование схем при разработке алгоритма целесообразно. Это обеспечивает приобретение прочных навыков разработки алгоритмов с использованием их типовых структур, являющихся основой так называемого **структурного подхода**, особенно плодотворного при постановке и решении на ЭВМ сложных задач.

Проверьте свои знания

1. Какие способы написания алгоритмов вы знаете?
2. Что понимают под схемой алгоритма?
3. Поясните условные графические обозначения, используемые в блок-схемах алгоритмов.
4. Почему блок-схемы обычно используются для изображения простых алгоритмов?
5. В чем заключается и как используется в записи алгоритмов знак присваивания?

Тренировка

I. Заданы три числа X, Y, Z. Найти максимальное из этих чисел. Результат записать в переменную S.

Составить алгоритм на естественном языке и записать его в виде блок-схемы.

II. Пусть задана функция знака числа (она называется сигнум: sign x):

$$y \triangleq \begin{cases} -1, & \text{если } x<0 \\ 0, & \text{если } x=0 \\ 1, & \text{если } x>0 \end{cases}$$

Составьте алгоритм на естественном языке и запишите его в виде блок-схемы.

III. Составьте алгоритм решения неравенства $ax>b$ (где a и b – произвольные действительные числа). Напишите его на естественном языке и в виде блок-схемы.

IV. Составьте блок-схему решения уравнения вида $ax^3 + bx=0$.

Примечание:

1. При $a \neq 0$ и $b/a < 0$ три действительные корня:

$x_1=0$; $x_2= \sqrt{-c}$; $x_3 = -x_2$, где $c = b/a$.

2. При a = b = 0: x – любое действительное число.
3. При других значениях a и b – один корень: $x_1 = 0$.

V. Составьте блок-схему решения уравнения вида: $Y=P^n$, где P – любое действительное число; n – натуральное число.

ПОДВЕДЕМ ИТОГИ

Создание алгоритма для решения задач какого-либо типа, его представление исполнителю в удобной для него форме – это творческий акт. Образно говоря, всю нашу жизнь, и в частности историю математики, можно было бы назвать историей открытия алгоритмов и их внедрения в человеческую практику.

☝ **Свойства алгоритма** – это свойства, отличающие алгоритм от любых предписаний и обеспечивающие его автоматическое исполнение. К ним относятся: **понятность; однозначность; дискретность; массовость; конечность; результативность.**

Алгоритм может быть представлен различными способами: на разговорном естественном языке; на языке блок-схем; на языке программирования.

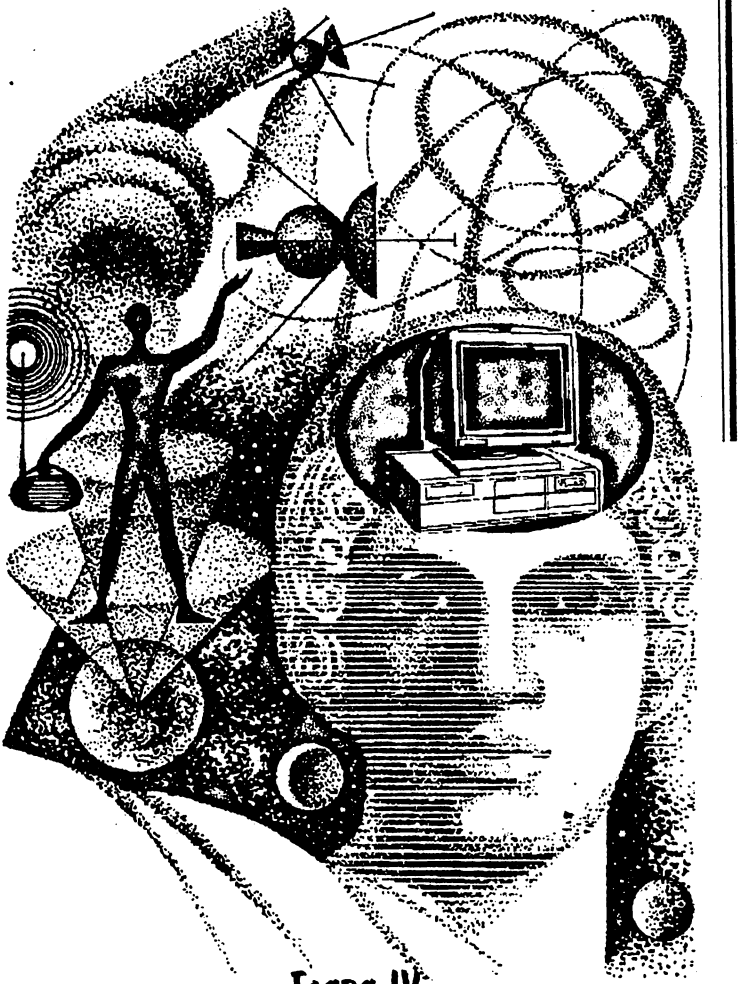
Укажем, что первые понятия языка блок-схем алгоритмов ввели в 1956 г. советские математики А.А.Ляпунов и Ю.Н.Янов. На Украине в 1959 г. Л.А.Калужнин.

В каждой блок-схеме алгоритма последовательность указаний исполнителю изображается с помощью линий и условных обозначений: условия ветвления – ромбы; элементы указания работ (присваивания, вычисления и т.д.) – прямоугольники; элементы ввода информации и ее вывода – параллелограммы.

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

Под АЛГОРИТМОМ понимается конечная система указаний, адресованных исполнителю, четко и однозначно задающих процесс решения задач какого-либо типа во всех деталях.

Сегодня мы наблюдаем, как ширится класс задач, которые удается алгоритмизовать.

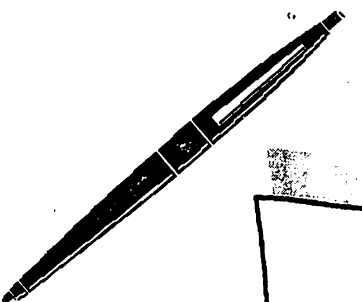


Глава IV

МАТЕМАТИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ЭВМ

"Нам говорят: безумец и фантаст.
Но, выйдя из зависимости грустной,
С годами мозг мыслителя искусный
Мыслителя искусного создаст."

И. Гете



Какими бы совершенными ни были электронно-вычислительные машины, без набора соответствующих программ компьютеры превращаются в бесполезную груду стекла, пластмассы и металла. Успех применения ЭВМ в любой области человеческой деятельности зависит прежде всего от того, есть ли в наличии программы для решения возникающих задач и насколько эти программы совершенны.

Поскольку ЭВМ применяются для решения все новых и новых задач, **программное обеспечение** (ПО) постоянно развивается. Совершенствование ПО можно сравнить с ростом дерева (рис. 12).

"Корень дерева" – это аппаратные средства; именно они определяют возможности ЭВМ и круг задач, которые можно решить с помощью данной машины (см. § 53).

"Надземная часть дерева" – это математическое обеспечение, которое включает в себя:

"Ствол и крона дерева" – программное обеспечение и "листья" – информация, связанная с использованием этих программ.

По мере роста дерева усиливается его корневая система (аппаратные средства), появляются все новые и новые "ветви" – программы нового назначения, а также "листья" – информация к ним.

Современные ЭВМ – это не только комплекс технических устройств, но и приложенный к нему, как его основная часть, заранее разработанный комплект программ.

Этот комплект программ образует так называемое **математическое обеспечение** (МО). По современным оценкам, доля математического обеспечения (т.е. всего того, что не является аппаратурой) составляет в среднем 80% от общей стоимости компьютерного продукта.

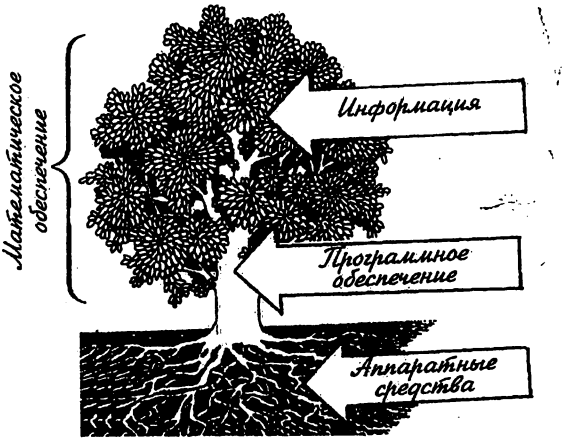


Рис. 12

Иногда бывает ситуация, когда, не зная, как решить ту или иную задачу, человек думает: "Надо заложить в ЭВМ, и пусть она посчитает". Это говорит о том, что не все понимают, что можно, а чего нельзя ждать от компьютера и почему нельзя. А компьютер может делать лишь то, что осмыслил человеческий разум. Необходимо запомнить главное:

"Ум" компьютера – это ум человека, воплощенный в программе.

§ 17. Языки программирования

Мы уже знаем, что алгоритм, записанный на языке программирования, называется **программой**. А что же мы понимаем под языком программирования?

ЯЗЫК ПРОГРАММИРОВАНИЯ – это совокупность набора символов (алфавит) системы, правил образования (синтаксис) и истолкования конструкций из символов (семантика) для задания алгоритмов с использованием символов естественного языка.

Именно для удобного решения задач с помощью ЭВМ искусственно и создавались языки программирования. Естественным же языком, который "понимает" компьютер, является машинный. **Машинный язык** – это такой язык, который компьютер воспринимает непосредственно, то есть это язык машинных команд данной модели компьютера. А мы уже знаем, что ЭВМ "понимают" только язык двоичных знаков: нулей и единиц.

Процесс описания конкретного алгоритма на языке машинных команд называется **программированием в машинных кодах**. Для выполнения этой работы программист должен знать коды всех машинных операций, назначение и особенности применения каждой из них, а также помнить адреса конкретных ячеек памяти, хранящие те или иные операции. Процесс разработки такой программы чрезвычайно трудоемок и непроизводителен. Поэтому для своего облегчения программисты придумали язык, переводящий символические имена в машинные коды. Ведь гораздо легче запомнить какое-то ключевое слово, чем соответствующий ему двоичный код. Такие программы, работающие с помощью мнемонических (символьных) обозначений, называют **ассемблерами**. Они и сейчас находят широкое применение, особенно при разработке эффективных, быстродействующих программ.

Именно такие машинные и полумашинные языки программирования относят к **языкам низкого уровня**. Их еще называют **машинно-ориентированными языками** (сюда относят и автокод).

В 1955 г. появился первый язык высокого уровня. Программы, написанные на таком языке, представляли собой набор уже не отдельных машинных команд, а более крупных элементов, называемых **операторами** дан-

ного языка. На языке высокого уровня исходная программа состояла из последовательности операторов.

Именно такие языки и ориентированы на описание алгоритмов. Поэтому их еще называют **алгоритмическими языками**.

Все алгоритмические языки отличаются прежде всего наглядной формой реализации алгоритма в виде программы, поскольку используют привычную математическую символику и ограниченный набор понятных ключевых слов. Основным достоинством алгоритмических языков является их универсальность, то есть независимость от конкретного типа машин (машинно-независимые языки).

Поскольку машина "понимает" только свой машинный язык, программа на алгоритмическом языке перед выполнением переводится на этот язык с помощью специальной программы – **транслятора**, название которой происходит от английского слова *translator* (переводчик). В программе-трансляторе "заложены" все правила алгоритмического языка и способы преобразования различных его конструкций на машинный язык.

Существуют два способа трансляции:

---

1. ИНТЕРПРЕТАЦИЯ (Interpretation) – метод выполнения в ЭВМ программы, заданной на языке программирования, при котором инструкция исходной программы переводится и сразу выполняется.

---

2. КОМПИЛЯЦИЯ (Compile – собирать) – метод выполнения в ЭВМ программы, но не сразу, а лишь тогда, когда собран перевод всего текста программы.

Разницу между компиляцией и интерпретацией можно пояснить с помощью аналогии. Фармацевты в аптеке приготовляют микстуру по старинному рецепту, написанному на латыни. Есть два пути: можно сначала перевести (скомпилировать) рецепт на родной язык и лишь затем готовить лекарство на родном языке. А можно, по мере чтения перевода рецепта, сразу готовить лекарство, но не записывать сам текст перевода (т.е. только интерпретировать). В последнем случае мы не получим текста рецепта на родном языке, а сразу получим микстуру, прав-

да, если лекарство нужно готовить несколько раз, рецепт придется переводить многократно.

Интерпретация используется в простых языках, где требуется несложная трансляция (Бейсик), или там, где компиляция слишком сложна или даже невозможна (язык ЛИСП). Часто используют оба эти способа совместно: интерпретатор – для отладки и компилятор – для трансляции отлаженной программы.

Работа с программой, написанной на алгоритмическом языке, очень упрощается за счет относительной простоты написания программы, удобной читаемости, возможности ее подкорректировать. Однако при этом всплывают и недостатки: требуется дополнительное время на трансляцию и дополнительная память для размещения транслятора.

Итак, в 1955 году увидел свет первый алгоритмический язык **Фортран.** Он использовался для решения научно-технических и инженерных задач. Слово "Фортран" образовано от начальных слогов английских слов – formula translator (переводчик формул). Он был разработан сотрудниками фирмы IBM под руководством Джона Бэкуса. Основным назначением этого языка является программирование численных расчетов на ЭВМ.

Неутомимый помощник

"Бурное развитие вычислительной техники..." Звучит зуммер...

Примерно так закончится ваша попытка напечатать расхожий штамп на американской пишущей машинке, в которую встроен текстовый процессор.

Помимо предупреждения о попытке напечатать ранее использованную фразу встроенный компьютер предостережет вас от использования противоречивых слов и от синтаксических ошибок. Если же вы захотите более серьезной помощи, ваша пишущая машинка начнет следить за пунктуацией, подбирать определения и подсказывать синонимы из своего словаря для чрезмерно часто употребляемых слов.

Но если и после всего этого вы умудрились наделать ошибок, машинка предоставит вам возможность отредактировать текст в ее памяти. И только потом напечатает его на бумаге.

Как и многие естественные языки (украинский, русский, английский и т.д.), Фортан (и другие языки) имеет много "диалектов" (их называют версиями), различающихся правилами записи некоторых команд, но по сути одинаковых.

За прошедшие годы было много новых версий языка Фортан. Он все время менялся и развивался. Одна из последних версий – Фортран-77. Благодаря простоте и тому, что этим языком написаны большие библиотеки программ, Фортран и в наши дни является одним из самых распространенных в мире языков программирования.

Затем в 1960 г. появился **Алгол** (Algoritmic language – алгоритмический язык), также ориентированный на научное применение, в него было введено множество новых понятий, подхваченных позднейшими языками, например, понятие **блочной структуры**.

Также при поддержке фирмы IBM появился язык **Кобол** (Cobol – сокращенное от английских слов Comnon business oriented language – общепринятый деловой ориентированный язык). Он был ориентирован на решение экономических задач, а точнее – на обработку информации.

Язык **Бейсик** (Basic – beginners all-parpouse sumbólic instraction code, что в переводе с английского означает "многоцелевой язык символических инструкций для начинающих") был разработан профессорами Дартмутского колледжа (США) Т.Куртцем и Дж. Кемени в 1965 году для обучения студентов, незнакомых с вычислительной техникой. Этот язык, напоминающий Фортран, но более простой, быстро стал очень популярным. Особенно его популярность повысилась благодаря "взрыву микроинформатики" – появлению персональных микрокомпьютеров, где Бейсик стал основным языком программирования.

Достоинствами Бейсика являются удобные средства ввода, отладки и испытания программ, а также возможность доступа ко всем основным ресурсам компьютера. Его отличает простота конструкций и возможность осуществления диалогового режима работы с ЭВМ.

Вместе с тем Бейсик имеет и ряд недостатков. Это прежде всего отсутствие явных ограничений на составление запутанных программ (этот недостаток присущ и Фортрану). Его оператор Goto при бездумном применении сильно запутывает программу и порой делает ее совсем непонятной с точки зрения логики выполнения. Кроме того, программы на языке Бейсик обычно выполняются относительно медленно, поскольку ЭВМ применяют, как правило, не компиляторы, а интерпретаторы языка.

Но последние диалекты языка Бейсик все больше устраняют перечисленные недостатки и приближают его к языку Паскаль и другим **процедурным языкам** (т.е. основанным на понятиях алгоритмов, программ, инструкций).

С его помощью можно решать достаточно сложные задачи. Например, так называемая версия расширенного Бейсика, имеющая матричные операции, позволяет с помощью одного оператора преобразовывать большие таблицы (матрицы). Есть версии Бейсика, работающие в режиме компиляции, что значительно ускоряет прогон программ, написанных на этом языке.

Итак, простота, совмещенная с мощными инструментальными возможностями, а также наличие у всех без исключения персональных ЭВМ интерпретатора этого языка, делают его самым распространенным среди начинающих пользователей ЭВМ.

В 1967-1968 гг. появился язык **PL/1** (Programming language – универсальный программно-ориентированный). Он также был создан на

Известно ли вам, что...

Бейсик жил,
Бейсик жив,
Бейсик будет жить...

Отмечая 30-ю годовщину создания языка Бейсик, фирма Microsoft сообщила, что, по ее подсчетам, на Бейсике сейчас программируют около 6 млн человек – в 3 раза больше, чем на всех остальных языках, вместе взятых. Начиная с 1985 года фирмой было продано более 1 млн экземпляров пакета Квикбейсик, и спада популярности этого языка не ожидается.

фирме IBM, но уже в качестве универсального языка программирования. Этот язык, как языки программирования СИ, Ада и Паскаль, может использоваться как для научных задач, так и для задач управления. Он очень мощный, но и очень сложный, используется лишь в высших учебных заведениях и научно-исследовательских центрах.

В 1970 г. профессор Никлаус Вирт создал в Цюрихском политехническом университете язык **Паскаль** (Pascal). Создатель языка назвал его в честь Блеза Паскаля – первого конструктора устройства, которое теперь относится к классу цифровых вычислительных машин. Он создавался как язык, который, с одной стороны, был бы хорошо приспособлен для обучения программированию, а с другой – давал бы возможность эффективно решать самые разнообразные задачи на современных ЭВМ. При создании этого языка Вирт большое внимание уделял хорошему стилю программирования (так называемое **структурное программирование**), благодаря которому конструкции Паскаля позволяют писать надежные, легко проверяемые программы с ясной и четкой структурой.

В 1980 г. появился язык **Ада**. Назван он в память об Аде Лавлейс – дочери английского поэта Лорда Байрона, первой программистки в истории вычислительной техники (см. с. 212). Он был создан во Франции по заказу американского министерства обороны как универсальный язык программирования. Это самый новый и самый мощный из языков программирования, он унаследовал качества языков Паскаль и Алгол-68 и дополнительно приобрел многие другие качества: **системное программирование, параллельность** и т.д.

Языки **Лисп** (List procssing language – язык обработки списков), разработанный американским профессором Джоном Маккарти в 1961 г., и **Пролог** (Prolog – programmation en logique – логическое программирование), разработанный Колмероэ и другими учеными университета Люммини во Франции в 1973 г., – это основные языки для задач, связанных с **искусственным интеллектом**. Лисп оперирует **списками** (цепная последовательность элементов), а Пролог – **деревьями** (логическими разветвлениями).

Работая в среде данных языков, компьютер решает задачу, а программист лишь ее формулирует. ЭВМ пробегает все рабочее пространство в поисках решения специальным способом, но зато полностью исчерпывающим (возможен обратный ход).

Существует огромное множество специализированных языков, позволяющих эффективно решать задачи в некоторых областях: моделирования (языки **Симула, Симкрит** и **GPSS**), управления аппаратурой (**ФОРТ**), для написания системных программ (**СИ**), написания баз данных (**Кодасил**), обучения программированию (**Лого, Робик, алгоритмический язык А.П.Ершова**) и другие.

Но в целом эволюция машинных языков происходит в направлении естественного языка – идеальное решение, состоящее в том, что пользователю надо будет только сформулировать задачу на естественном языке, а все остальное сделает компьютер. Конечно, в настоящее время это всего лишь мечта, но тем не менее языки последних поколений более близки к языку человеческих рассуждений.

Проверьте свои знания

1. Что мы понимаем под языком программирования?
2. Что такое программирование в машинных кодах?
3. В чем состоит различие между языками программирования высокого уровня и низкого уровня?
4. Что такое программа-транслятор?
5. Объясните разницу в выполнении программы с помощью интерпретатора и компилятора.
6. Почему язык Фортран остается популярным языком программирования по настоящее время?
7. Каковы основные недостатки языка Бейсик?
8. Назовите процедурные языки. В чем состоит их основная особенность?
9. В честь кого названы языки программирования Паскаль и Ада?
10. Назовите языки, предназначенные для решения задач, связанных с проблемой искусственного интеллекта.
11. Какие вы знаете специализированные языки программирования? Для каких целей они используются?
12. Куда направлена тенденция эволюции машинных языков?

§ 18. Понятие об операционной системе. Файловая система

В предыдущем параграфе мы выделяли для языков программирования условные уровни иерархии; каждому уровню соответствует свой язык. Самый нижний уровень иерархии, о котором шла речь, – **машинный язык**. Чуть выше уровень, соответствующий полумашинным языкам, – **ассемблеры**. И, наконец, самый высокий уровень соответствует **языкам высокого уровня** (алгоритмическим языкам).

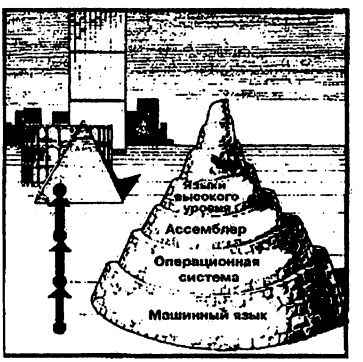


Рис. 13

Однако существует еще один очень важный уровень иерархии в организации ЭВМ: он находится между машинным языком и ассемблером – это уровень **операционной системы** (ОС) (рис. 13):

4-й уровень – языки высокого уровня;

3-й уровень – ассемблер;

2-й уровень – операционная система;

1-й уровень – машинный язык.

ОПЕРАЦИОННАЯ СИСТЕМА представляет собой совокупность программ (их называют СИСТЕМНЫМИ) и является посредником между аппаратными средствами, т.е. непосредственно самой ЭВМ с соответствующим машинным языком и пользователем.

Какие же функции выполняет операционная система? Зачем она нужна? Прежде всего отметим, что всякая ОС – **программная система** – помогает пользователю решать его задачи. Она освобождает его от многих рутинных и довольно нудных операций, связанных с использованием аппаратных средств компьютера – процессора, оперативной памяти, внешней памяти, печатающего устройства и т.д. Именно через операционную систему пользователь полу-

чает доступ к этим ресурсам, управляет работой компьютера, получает информацию, где находятся нужные ему программы и данные, узнает, куда направить полученные результаты.

ОС помогает ответить на многочисленные вопросы, возникающие у пользователя: как ввести или вывести информацию на имеющиеся внешние устройства, как запустить на выполнение другие программы: трансляторы, редакторы, программы пользователя, как приспособить для решения своей задачи программы, имеющиеся в наличии, куда направить готовую или подготавливаемую программу и т.д. и т.п. Таких вопросов и задач во время диалога с компьютером у пользователя возникает множество. И все они разрешаются при обращении пользователя к ОС.

Таким образом, ОС использует все возможности компьютера и весь **сервис**, необходимый для эффективной работы пользователя в диалоговом режиме. При этом важнейшей функцией операционной системы является организация и поддержание того, что называется **файловая система.**

ФАЙЛОМ (от англ. file – досье, набор документов) называют упорядоченный набор записей или элементов данных, выступающих как самостоятельный единый объект.

Например, файл может представлять собой некоторую программу или систему программ, произвольный текст, набор данных, подлежащих обработке программами, и т.д.

Как правило, ОС и вся файловая система записываются на магнитные диски. Наиболее распространенным типом магнитных дисков являются гибкие сменные диски (**дискеты**), которые из-за малых габаритов и массы оказались особо удобными для промежуточного хранения данных и программ в процессе работы на персональных ЭВМ. Гибкий, или **флоппи-диск**, – это майларовый диск толщиной 0,125 мм и диаметром 133 мм (5,25 дюйма), покрытый с двух сторон высококачественным ферролаком, допускающим непосредственный механический контакт с головками чтения записи при вращении его в дисководе.

Гибкий диск находится в защитном конверте, который имеет "окно" для подвода головок чтения-записи к рабочим дорожкам диска, а также небольшое индексное отверстие, обеспечивающее оптический способ отметки начала дорожки (рис. 14). Внутренняя поверхность конверта обладает отталкивающими свойствами, так что при вращении диск как бы пролетает внутри конверта.

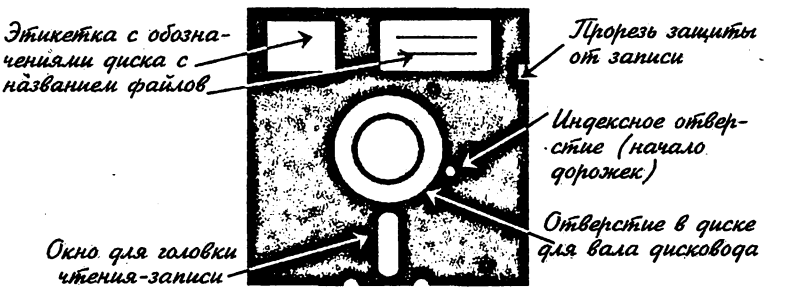


Рис. 14.

Данные всегда записываются на магнитной поверхности в виде концентрических окружностей, называемых **дорожками**. Каждая дорожка, в свою очередь, состоит из нескольких **секторов** (рис. 15).

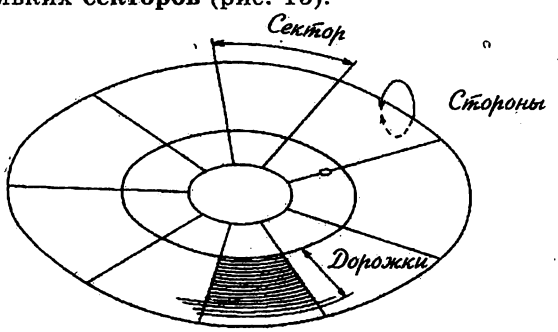


Рис. 15. Физическая структура магнитного диска

Подобную физическую структуру имеют и **жесткие магнитные диски**. Только изготовлены они из алюминиевых пластин, покрытых магнитным материалом, заключенных в защитную герметическую оболочку (ящик). От-

крыть ящик означает безнадежно загрязнить его содержимое, поэтому они, как правило, несъемные и в совокупности с дисководами прозваны **винчестерскими дисками** (из-за совпадения кодового номера дисковода с номером модели знаменитого винчестерского ружья).

Перед первым использованием дискеты ее необходимо специальным образом **инициализировать**, т.е. ее поверхность размечают должным образом, устраняют выявленные дефекты и отводят конкретное место для каталога. Такой процесс инициализации диска называют **форматированием**.

Использование ОС позволяет не учитывать физическую структуру диска. При этом работа с ним организовывается на так называемом **логическом уровне**, более естественном и простом, чем физический.

На логическом уровне гибкий магнитный диск представляет собой последовательность пронумерованных **логических блоков** информационной емкостью, равной емкости четырех секторов (512 байт).

Для обращения к блоку достаточно указать его номер, не вдаваясь в подробности физической организации. Однако емкость блока слишком мала, чтобы использовать его в качестве основной информационной единицы обмена между устройствами ЭВМ. Поэтому в ОС реализовано управление более крупными информационными структурами – **файлами**.

Файлы хранятся на дисках и могут содержать произвольное число блоков.

Каждый файл на диске имеет обозначение, которое состоит из двух час-

Что может компьютер

Точно, правдиво

Известно множество формул для расчета нормального веса человека, учитывающих множество параметров и позволяющих сделать быструю оценку состояния пациента. Кроме того, существуют профессионалы – врачи, портные, члены жюри конкурсов красоты, опыт которых позволяет порой получить и более точные результаты при определении избыточного веса человека.

Однако ни формулы, ни профессионалы, вооруженные ими, не смогут соперничать с прибором, разработанным недавно американскими медиками. В основе его действия лежит тот факт, что подкожный жир наиболее активно поглощает

свет с определенной длиной волны.

Прибор устроен следующим образом: от источника инфракрасного излучения свет поступает в цилиндрический корпус, прижатый открытым концом к бицепсу пациента. Отраженный подкожными тканями свет поступает на светоприемник, расположенный в центре корпуса в светоизолирующем цилиндре, и далее – в компьютер для анализа спектра.

Помимо отраженного света в ЭВМ вводятся рост, вес и другие антропологические параметры пациента, после чего на основе этих данных компьютер вычисляет и распечатывает процентное отношение веса подкожного жира к общему весу человека, указывает нормальный и максимальный вес и, если это необходимо, предлагает специальную диету и комплекс физических упражнений.

Работа с прибором не требует специальной квалификации. Его разработчики считают, что массовое распространение прибора сделает американцев более подтянутыми и жизнерадостными.

тей: **имени** и **расширения**. В имени файла может быть от 1 до 6 (иногда 8) символов латинского алфавита. Расширение начинается с точки, за которой следуют от 1 до 3 таких символов. Например,

Comman.com
имя. расширение

Нельзя в качестве имени файла давать названия операторов и команд (так называемые служебные слова).

Расширение файла предназначено для распознавания файлов, содержащих однотипную информацию. По нему вы можете также узнать, какая программа создала данный файл.

Вот некоторые стандартные типы расширений:

.com, .exe } файлы, пригодные для непосредственного исполнения

.txt, .lst, .doc } текстовые файлы

.dat – файл данных

.asc – файл, написанный в коде ASCII (американский стандартный код информационного обмена)

.bas, .for, .pas } файлы, содержащие исходный текст на соответствующих языках (Basic, Fortran, Pascal)

.bad – файл, содержащий плохие блоки

.obj – файл с объектным модулем

Каждое устройство файловой структуры (жесткий или гибкий диск, лен-

та) называется **томом**. Их **разметка** (форматирование) заключается в нанесении **магнитных меток**, физически разделяющих магнитный диск на блоки. Она производится соответствующими программами ОС (например, программа Format). При этом на томе выделяется место под запись каталога (оглавления или справочника). Каталог тома выполняет функцию, аналогичную оглавлению в обычных печатаных книгах.

В каталог (оглавление) машинного тома **прямого доступа** вносится учетная информация о каждом файле, вновь записываемом на магнитный диск: имя файла, его размер в блоках, дата создания и т.д.

Например:

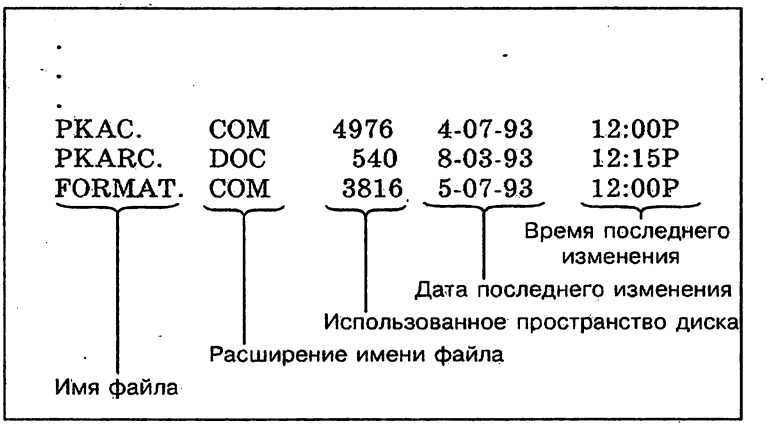


Рис. 16. Разделы каталога

Кроме того, в каталоге имеется номер первого блока, с которого начинаются внесенные в каталог файлы.

Для манипулирования файлами пользователю необходимо знать **язык команд**, с помощью которого он сможет активно вмешиваться в содержимое файлов. Назовем несколько из основных системных команд (общепринятых):

DIR – вывод на экран каталога или его части;
COPY – копирование файла;
RENUM – переименование файла;
ERASE (DELETE) – удаление файла;
TYPE – вывод файла на дисплей.

Эти команды позволяют просмотреть на экране дисплея каталог файлов, записанный на выбранном диске, скопировать их на другие носители (диски, магнитные ленты и т.д.), переименовать содержащиеся в нем файлы, стереть имя в каталоге и т.д. Естественно, что файловая система имеет и защиту от возможного искажения или стирания каких-либо очень важных файлов, например, тех, где хранится операционная система.

Общение пользователя с внешними устройствами (дисплей, накопитель на магнитных дисках, печатающее устройство, графопостроитель, каналы связи с другими ЭВМ и т.д.) осуществляется через специальные программы ОС, называемые **драйверами** (от англ. Driver – шофер, водитель). Так, если вы хотите вывести на экран дисплея содержимое определенного файла, достаточно указать номер диска и имя файла и, разумеется, дать команду его вывода на дисплей, в соответствии с этой командой файловая система по каталогу определит, где именно на диске расположен нужный файл, и сообщит информацию драйверу, который запустит нужный дисковод, переведет считывающие головки на нужную дорожку и считает файл в оперативную память. Далее вступит в работу драйвер дисплея, который переведет эту информацию на экран дисплея. Как видно, драйверы осуществляют взаимодействие компьютера с его внешними устройствами и отражают их специфику.

Операционная система и способствует повышению эффективности такого взаимодействия пользователя и компьютера, оптимально использует оборудование, управляет выполнением программ.

Естественно, возникает вопрос: если все ОС для ПЭВМ фактически выполняют одни и те же функции, то что должно являться критерием при выборе конкретной ОС для использования? Ответ на этот вопрос в общем виде можно сформулировать так: ОС эффективна при использовании, если она удовлетворяет сегодняшним потребностям и достаточно универсальна для того, чтобы обеспечить работу в будущем. Важнейшими характеристиками для окончательного решения вопроса о выборе операционной системы являются ее распространенность и наличие большого коли-

чества прикладных программных средств, реализованных в ее среде.

Однако в целом проблема совместимости ОС как с базовым, так и с прикладным программным обеспечением (ПО) решается с помощью **стандарта** на операционную систему. Назовем лишь два (пожалуй, самых распространенных) стандарта операционных систем:

1. Операционная система RT-11 (Real time systems PDP-11) – родоначальница всех однопользовательских систем для малых ЭВМ (семейство СМ-3, СМ-4, микроЭВМ "Электроника", ДВК и т.д.). Она разработана фирмой DEC в 1972 г. и является основой более поздних версий ОС Рафос, Фодос и др.

2. Операционная система MS-DOS (DOS – дисковая операционная система фирмы Microsoft corp.), разработанная для IBM – совместимых персональных ЭВМ, в начале 80-х годов (ее вариант: PC DOS, распространяется фирмой IBM с 1988 года). Сейчас существует огромное множество аналогов этой ОС.

Большинство ОС строятся по модульному принципу. Модули представляют собой отдельные файлы, содержащие программы, реализующие основные функции системы.

Диск, на котором записано ядро ОС, называют **системным** (сюда входят программы начальной загрузки, модули расширения базовой системы ввода-вывода, поддержания файловой системы, командный процессор и т.д.).

Кроме перечисленных модулей в состав ОС входят **утилиты**, представляющие собой отдельные программы для выполнения сервисных функций системы. Утилиты могут размещаться на системном или рабочих дисках.

Ранние версии ОС предполагали работу с **единственным** каталогом, описывающим все файлы. Однако с увеличением емкости запоминающих устройств количество файлов разрасталось до неуправляемой величины. Поэтому сейчас начали использовать **древовидную структуру каталогов.**

На каждом диске создается один главный **(корневой)** каталог. В нем записаны сведения о файлах и других, подчиненных ему каталогах, называемых **каталогами первого уровня**. Каждый каталог первого уровня наряду с име-

нами файлов может содержать названия подчиненных ему каталогов **второго уровня** и т.д. Имена каталогов формируются по тем же правилам, что и имена обычных файлов. Подчиненные каталоги называют **подкаталогами**. По отношению к подчиненному подкаталогу включающий его каталог называется **родительским** (рис. 17).

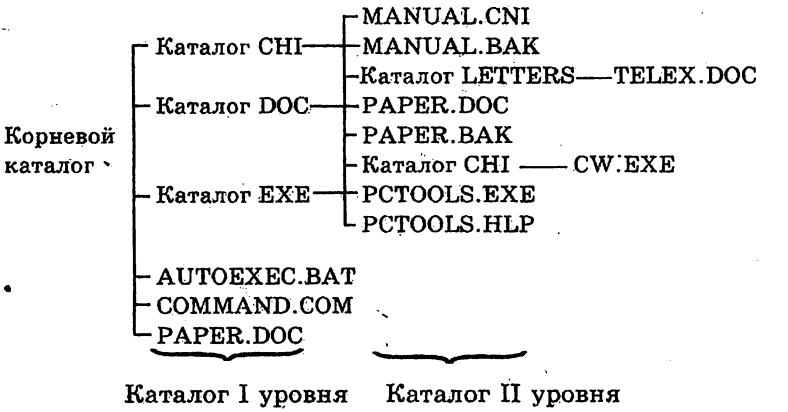


Рис. 17. Пример файловой системы на магнитном диске

Каталог, с которым в настоящий момент работает пользователь, называется **текущим**. Когда пользователь использует файл не из текущего каталога, тогда с помощью специальной символики он должен указать **путь к каталогу**, в каком он каталоге находится.

Проверьте свои знания

1. Какому уровню организации ЭВМ принадлежит операционная система? Объясните содержание понятия "операционная система".
2. В чем заключается процесс инициализации дисков?
3. В чем состоит преимущество логической организации магнитного диска перед ее физической структурой?
4. Поясните понятие "расширение" имени файла.
5. Какие системные команды вы знаете? Что они выполняют?
6. Что такое драйвер?

7. Какие принципы лежат в основе критерия выбора конкретной ОС?

8. Что такое каталог магнитного диска? Какие каталоги называют корневыми, а какие подкаталогами?

Тренировка

I. Научитесь форматировать диск.

II. Научитесь заносить на диск файл информации и считывать его, пользоваться каталогом файлов на диске.

III. На примере конкретных файлов укажите путь к файлу из корневого каталога в каталог какого-то уровня.

IV. Оцените объем использованной и оставшейся части диска. Что означает термин "защита информации на диске"?

V. Скопируйте информацию, хранящуюся на одном диске, на другой диск. Удалите отдельные файлы с диска.

VI. Научитесь копировать систему (ДОС) с одного диска на другой.

§ 19. Программное обеспечение ПЭВМ

В предыдущем параграфе мы ознакомились с основами организации операционной системы.

Если процессор и память выполняют роль мозга, а дисплей – лицо компьютера, то операционную систему считают его душой.

Перечисленные нами функции реализуются **системными** программами, составляющими **ядро ОС**. Ядро ОС постоянно находится в оперативной памяти ЭВМ. Остальное программное обеспечение (ПО) располагается на внешних носителях и чаще всего называется **окружением ядра ОС**. К нему относятся различные утилиты (программные оболочки, архиваторы, антивирусные программы, программы-доктора), электронные таблицы, редакторы, компиляторы, базы данных, графические интерфейсы, средства сетевой обработки и т.д.

Именно таким программным обеспечением мы и займемся, взяв за основу операционную систему MS DOS.

1. Программа-оболочка Norton commander

Практически в арсенале каждого, кто работает с персональным компьютером, обязательно есть программы, которые создал гениальный программист Питер Нортон.

Так что же такое "Нортон командер"? Она представляет собой одну из наиболее популярных программ-оболочек для работы с ОС DOS. Как правило, с ее помощью пользователь просматривает каталоги и файлы; копирует, переименовывает, удаляет файлы; запускает программы и т.д. Конечно, все это можно сделать и непосредственно, используя возможности самой ОС DOS. Однако гораздо удобнее делать это с помощью Norton commander. Зачем же используется, казалось бы, лишнее промежуточное звено при работе с DOS? Ответ прост.

Дело в том, что взаимодействие пользователя с операционной системой построено по принципу диалога: пользователь набирает на клавиатуре команды, а DOS выполня-

---

Питер Нортон

Знаменитый американский программист.

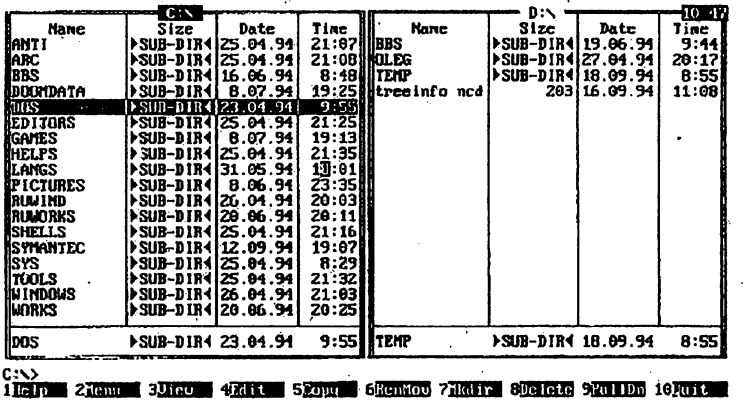
Родился в г.Сиэтле (штат Вашингтон, США), получил образование в Ридоновском колледже (Портленд, штат Орегон) и Калифорнийском университете в Беркли.

Он хорошо известен в современном компьютерном мире как "великий учитель" персональных компьютеров. Более 20 лет работает в области создания программных средств для различных типов ЭВМ. Кто не знает его популярную программу-оболочку "Нортон-командер" или не менее знаменитые "Нортон утилиты", "Нортон интегратор", "Нортон эдитор", разошедшиеся по всему миру в десятках миллионов копий. Он является автором значительного числа получивших одобрение читателей книг, в том числе и вышедших на русском языке.

Сейчас П.Нортон является управляющим процветающей фирмы по производству и распространению программного обеспечение Peter Norton Computing, Inc., расположенной в г.Санта-Моника, штат Калифорния. В этом городе он живет с женой и дочерью.

ет их. Такой режим взаимодействия *недостаточно удобен* и, мало того, *не нагляден*. Действительно, для того, чтобы скопировать файл в другой каталог, надо набрать имя команды, имя файла и имя каталога (такой набор называется **командной строкой**), а для этого надо помнить эти имена и не ошибиться при наборе. Куда проще "ткнуть" с помощью клавиш управления курсором или "мыши" в определенные места экрана, чтобы указать нужный файл, потом каталог, в который надо скопировать файл, а затем требуемое действие.

После запуска Norton commander в верхней части экрана появляются два прямоугольных **окна**, ограниченные двойной рамкой – эти окна называются **панелями**. Ниже этих панелей располагается обычное приглашение DOS. Еще ниже располагается строка, напоминающая значения функциональных клавиш Norton commander.



Мы не будем приводить полный перечень всех возможностей Nornon commander, скажем только, что она имеет очень удобную "сервисную" наполняемость.

Кроме Norton commander (NC) к сервисному набору программ Norton относят:

1. Norton utilités (NU) – утилиты Нортона – пакет, объединяющий ряд важных дисковых функций, просмотр и редактирование содержимого диска, таблицы распределения файлов (FAT), каталогов, таблицы разделов (PT). Кроме того, пакет включает функции поиска данных, ко-

пирования их из одной области в другую, восстановления потерянных и удаленных файлов, получения информации о состоянии диска.

2. Norton integrator (NI) – Нортон интегратор – представляет собой интегрированную среду, позволяющую в режиме меню выполнить ряд команд самого разнообразного назначения – от создания интерактивных пакетных файлов и выдачи звуковых сигналов до операций над дисками, каталогами и файлами, установки атрибутов экрана и получения всякого рода информации.

3. Norton disk doctor (NDD) – Нортон диск доктор – представляет собой простой и компактный меню-управляемый пакет, включающий функции диагностики дисков и некоторые средства восстановления информации на носителях.

II. Архиваторы

Как бы ни была велика емкость "винчестера", она, как и все на свете, конечна. Рано или поздно ресурсы жесткого диска будут исчерпаны и для его освобождения потребуется создать архив программ и текстовых файлов на дискетах.

Кроме того, при эксплуатации компьютера, по самым разным причинам, возможна порча или потеря информации на магнитных дисках. Для того чтобы сохранить нужные программы, следует также иметь архивные копии использованных файлов и систематически обновлять копии изменяемых файлов.

Конечно, можно скопировать обычными командами используемые программы, однако при этом такие копии будут занимать столько же места, сколько занимают оригиналы программ. Мало того, для копирования нужных файлов может потребоваться много дискет. Например, для копирования файлов с жесткого диска емкостью 20 Мбайт необходимо около 60 дискет емкостью по 360 кбайт.

Более удобно использовать для создания архивных копий специально разработанные программы – **архиваторы**. Они обладают тем преимуществом перед обычными программами копирования, что позволяют "сжимать", и довольно значительно (в особенности текстовые), файлы. Эти программы также позволяют объединять группы совместно используемых файлов в один архивный файл, что позволяет гораздо легче разбираться в архиве файлов.

Как правило, программы для архивации файлов позволяют помещать копии файлов на диске в сжатом виде в архивный файл, извлекать файлы из архива, просматривать оглавление архива и т.д. Разные программы отличаются форматом архивных файлов, скоростью работы, степенью сжатия, удобством использования.

Большой популярностью среди архиваторов пользуется комплект программ PKARC FAST (фирмы PKWARE INC). Основу комплекта составляют программы PKARC (архиватор) и PKXARC (реархиватор), а также программы PKZIP, PKUNZIP и PKPAR (той же фирмы). К наиболее распространенным можно также отнести программы PAK (фирма Nogate consulting), LHA (фирма Horuyasu Yoshizaki), PAK (фирма System enhancement associates), ARJ (фирма Robert K.Jung); RAR (фирма Eugene Roshal) и русифицированные: Charc (автор С. Чернивецкий) и Viarchiv (входящий в состав интегратора "Виктория", автор В.Ковалев).

III. Антивирусные программы

Что такое компьютерный вирус?

Компьютерный вирус – это небольшая программа, написанная в машинных кодах, способная внедряться в другие программы, хранящиеся на дисках запоминающих устройств компьютера.

Программа, внутри которой находится вирус, называется "зараженной". Когда зараженная вирусом программа вызывается для выполнения, на одном из этапов ее работы вирус перехватывает управление и реализует функции заложенные в него разработчиком. Обычно таких функций две: заражение других программ и дисков и поражение зараженных систем.

Для маскировки вируса действия по заражению других программ и нанесению вреда могут выполняться не всегда, а, скажем, при выполнении определенных условий. После того, как вирус выполнит свои вредные действия (например, портит файлы или таблицу размещения файлов на диске, "засоряет" оперативную память и т.д.), он передает управление той программе, в которой находится, и она работает как обычно. Тем самым внешне работа зараженной программы выглядит так же, как и незараженной.

Считают, что идею создания компьютерных вирусов подбросил писатель-фантаст Т.Дж.Райн, который в одной из своих книг, опубликованной в США в 1977 г., описал эпидемию, за короткое время поразившую более 7000 компьютеров. Причиной эпидемии стал компьютерный вирус, который, передаваясь от одного компьютера к другому, внедрялся в их ОС и выводил компьютеры из-под контроля человека.

Условно выделяют две фазы жизни вируса: пассивную и активную.

Пассивная стадия – когда он практически не проявляет себя, стараясь оставаться незаметным для пользователей. Получая управление на этой стадии, вирус отыскивает на других дисках компьютера системные или прикладные программы и внедряется в них. Продолжительность этой фазы может быть разной: от нескольких минут до нескольких лет.

Стадию активных действий по поражению зараженных систем называют **атакой вируса**.

Вирусная атака может начинаться одновременно на всех пораженных компьютерах или в разное время. Обычно атака начинается с выполнения некоторого общего для всех компьютеров условия. Например, один из распространенных вирусов "Israel virus" запрограммирован так, что переходит в атаку в пятницу, если это 13-е число (любого года, любого месяца), а коварный вирус "Lehigh" активизируется после каждых четырех циклов заражения других программ. Начало активным действиям вируса может положить также достижение определенного количества вызовов зараженной программы на исполнение и т.п.

Если не предпринимать мер по защите от вируса, то последствия заражения компьютера могут быть очень се-

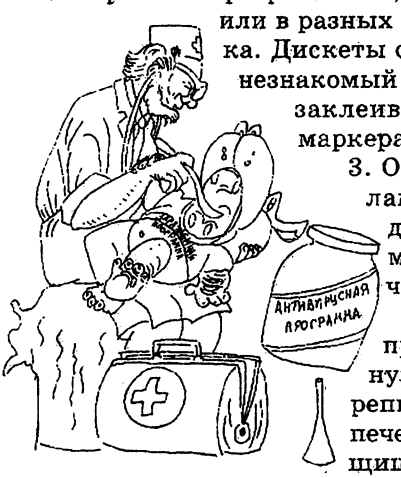
рьезными. Например, в начале 1989 г. вирусом, написанным американским студентом Моррисом, были заражены и выведены из строя тысячи компьютеров, в том числе принадлежащие министерству обороны США. Автор вируса был приговорен судом к трем месяцам тюрьмы и штрафу в 270 тыс. долларов. Наказание могло быть и более строгим, но суд учел, что вирус не портил данные, а только размножался. Некоторые авторы вирусных программ создали их из озорства, не понимая всех последствий распространения вируса. Например, включением компьютера в новогоднюю ночь запускается шуточный вирус с надписью: "Какой дурак работает в такую ночь!". Другие авторы – из стремления "насолить" кому-либо (например, уволившей его фирме) и т.д. Но в любом случае – это высококвалифицированные программисты.

Предотвратить заражение вирусом с высокой степенью эффективности могут как элементарное соблюдение правил "гигиены" пользователей ПЭВМ, так и использование антивирусных программ.

Перечислим основные правила так называемой компьютерной "гигиены":

1. Воздерживайтесь от случайных контактов и связей, выражающихся в использовании чужих компьютеров и дискет, стерильность которых может ставиться под сомнение.

2. Храните программы и данные на разных дискетах или в разных подкаталогах жесткого диска. Дискеты с программами вставляйте в незнакомый компьютер, предварительно заклеив полоской фольги прорезь маркера защиты данных.



3. Ограничивайте доступ к файлам программ, устанавливая для них всегда, когда это возможно, статус "только для чтения".

4. Никогда не копируйте программы для собственных нужд со случайных дискет. Переписывайте программное обеспечение с дисков оригиналов, защищенных от записи.

5. Не загружайте ОС в компьютер со случайных дискет, компьютеры; имеющие жесткий диск, должны загружаться с этого диска.

6. Присваивайте имена всем дискам и привыкайте контролировать эти имена при просмотре каталогов на экране дисплея.

7. При работе в сети, по возможности, не вызывайте программы из памяти других компьютеров.

Наряду с выполнением этих правил можно с достаточно высокой степенью эффективности предупредить заражение, используя специально разработанные для этих целей программные средства – **антивирусы.**

Но предостережем читателя: для такой работы необходимо знать **методику использования антивирусных программ**. Не обладая ею, пользователь может, наоборот, необратимо испортить файлы вместо их лечения.

Существует огромное множество антивирусных программ. Наиболее популярными являются: SCAN (создатель фирма McAfee Associates); AIDSTEST (автор М. Лозинский); Dr.Web (автор И.Данилов, С.-Петербург) – ревизор-программа (следит за изменениями на диске); ADINF (Дм. Мостовой); AVP (фирма КАМИ).

В заключение скажем, что желательно использовать антивирусные программы последних версий, т.к. такого рода программы постоянно переделываются для включения новых типов вирусов, и чтобы одна из них была отечественного производства, т.к. процесс эмиграции вируса в другие страны и иммиграции антивирусных программ занимает довольно большое время (поэтому не выполняется принцип оперативности).

И последнее – не собирайте коллекции антивирусных программ неизвестного назначения – с ними можно получить новый вирус.

IV. Текстовый редактор

Программы обработки текстов называют **текстовыми редакторами** (за рубежом Word processing – текстовые процессоры), они являются одними из основных обязатель-

ных программных средств, которые должен освоить каждый пользователь.

Текстовым редактором пользуются практически все, кто для написания книг, статей, писем и т.д. использует компьютер. Это прежде всего ученые, инженеры, журналисты, писатели, администраторы, клерки и т.д., то есть люди, далекие от вычислительной техники. Для этой категории "пишущих" пользователей нужна только программа редактора и файловая система для хранения написанного текста. Но программе редактора все равно, что редактировать – программу на алгоритмическом языке, текст художественного произведения или научной статьи. Для компьютера все это текст, в котором пользователь хочет сделать какие-то изменения. Зная несложный язык для общения с редактором, пользователь может легко изменять свой текст, как ему захочется, например, убрать слово, предложение или абзац, вставить что-либо, заменить одно или несколько слов и предложений на другие и т.д.

Редактор может также представлять текст на экране дисплея в том виде, в каком его хочет видеть пользователь. Например, определять размер страницы, выровнять поля, перенумеровать страницы текста, определить расположение графиков в тексте, выбрать шрифт. Журналистам и писателям текстовый редактор позволяет обращаться к словарю синонимов и антонимов, исправляет орфографические и синтаксические ошибки, расставляет (в соответствии с правилами разговорного языка) знаки препинания и даже может подстраивать написанный текст под стиль выдающихся писателей и поэтов (или под свой собственный стиль письма).

Широко распространены одно-, двух- и многооконные текстовые редакторы. В настоящее время для широкого класса ПЭВМ создано большое количество пакетов обработки текстов, которые различаются объемом реализуемых функций.

Приведем наиболее популярные: Лексикон (LEXICON – разработчик Е. Н.Веселов), ChiWriter (фирма Horstmann Software Design corporation), Brief (фирма Underware incorporated), Foton, K-report, Multedit (American cibernetics) и др.

Что может компьютер

Телевизор с искусственным интеллектом

Что означают буквы JVC, знают, наверное, все. Высококлассная бытовая техника, выпускаемая на предприятиях этой японской фирмы, не просто соответствует уровню мировых стандартов, но и определяет его. И неудивительно, что именно JVC еще раз подняла планку.

Здесь разработан пульт дистанционного управления телевизором, оснащенный элементами искусственного интеллекта. У владельцев таких видеосистем постепенно появляется ощущение, что телевизор приспосабливается к их привычкам.

V. Графический редактор

Для выполнения чертежных и графических работ на ЭВМ используются программы, которые называются **графическими редакторами**. Использование возможностей графического редактора избавляет от необходимости разрабатывать программу для создания на экране требуемого изображения.

С помощью специальных средств редактора можно нарисовать на экране отрезок, прямоугольник, эллипс и другие геометрические фигуры. Их можно раскрасить, заштриховать, забелить и т.д. Но можно отредактировать готовый (нарисованный пользователем или сканированный с иллюстрации или схемы) рисунок. На нем можно удалить, стереть ту или иную деталь изображения, раскрасить в различные цвета, наложить, видоизменить изображения и т.д. Часто приходится в выбранном месте экрана сделать ту или иную надпись.

Такого рода графические редакторы называются **программами рисования**. Вот некоторые из них: PCpaint, FREELANCE.

Но иногда пользователю приходится построить нужный график. Для этого в памяти программ графического редактора хранятся несколько типичных видов графиков и диаграмм. Это прежде всего обычный график в виде кривой, точнее – ломаной, выражающей зависимость одного фактора от другого. Пользователю следует ввести лишь значения координат точек графика и масштабы по осям,

все остальное выполнит редактор. Есть диаграммы соотношений, с помощью которых удобно иллюстрировать рост или падение какого-то показателя во времени, что любят администраторы. Распределение какого-либо ресурса удобно представлять в виде круговой диаграммы с разделенными секторами.

Такие графические редакторы называются **вычерчивающими программами** (деловая графика). Иногда обе программы совмещены в одном графическом редакторе.

Вот некоторые названия таких редакторов: Graphwriter (фирма Graphics communications), STORYBOARD, PAINTBRUSH, NeoPaint и др.

VI. Электронные таблицы

Вы никогда не задумывались над вопросом составления расписания ваших уроков в учебном заведении? Завучу, составляющему такое расписание, надо учесть большое количество разных фактов: требования учебного плана, пожелания учителей, разделение классов на подгруппы, чередование умственных и физических нагрузок, количество учебных помещений и т.д. Особенно прибавляется забот, когда кто-то из учителей заболел или не закончен ремонт в каком-либо кабинете. При этом завучу приходится заново учитывать большой объем данных и связей между ними.

С подобными задачами сталкиваются не только завучи. Работникам плановых отделов, учреждений и организаций, диспетчерам при составлении графиков движения транспорта,

Микрокомпьютер, встроенный в новую видеосистему, анализирует и запоминает самые популярные передачи в течение суток, выделяя три наиболее часто используемых телеканала. Владельцу достаточно нажать кнопку "Любимые передачи" – и на экране, разбитом на несколько полей, в уменьшенном размере появятся передачи, идущие по этим каналам, и меню, позволяющее выбирать нажатием соответствующей цифры любой из них.

Кроме того, система может запомнить и обеспечить быстрый просмотр еще 25 наиболее популярных каналов, которые разбиваются на пять тематических групп: спутниковые, кинофильмы, спорт, новости и музыка.

Аналогичным образом система следит и за уровнем громкости. Сутки в ее понимании разбиты на 24 часа, для

каждого устанавливается определенная громкость.

Если у владельца этой мудрой системы есть дети, он может ограничить их ежедневные телепорции определенным часом, после наступления которого пропадает звук, а изображение сменяется приятным голубым полем. Но при определенном упорстве с пульта дистанционного управления можно подобрать трехзначный цифровой пароль, который позволит смотреть передачи и после наступления "тихого часа".

Новая система сохранила достоинства предыдущих поколений подобных устройств. С ее помощью можно

бухгалтерам при составлении сметы, организаторам спортивных соревнований (при составлении турнирных таблиц и размещении участников соревнований в гостиницах и кемпингах, организации их питания, отдыха, транспортных перевозок) и т.д.

И вот здесь-то нам на помощь приходит компьютер со специально созданными программами – **электронными таблицами.**

Электронная таблица позволяет хранить в табличной форме большое количество исходных данных, результатов, а также связей (математических соотношений) между ними. Но главное – при изменении начальных данных все результаты автоматически пересчитываются и заносятся в таблицу.

Именно такая программа Visi Calc – программа электронных таблиц (созданная фирмой Software Arts) – олицетворяет начало компьютерной революции. С появлением этой программы пользователь, не владеющий искусством программирования, получил возможность создавать финансовые модели и легко ими манипулировать: сохранять, извлекать из памяти и корректировать за несколько минут, вносить дополнения и уточнения и пересчитывать заново за несколько секунд, выводить путем нажатия некоторых клавиш всю модель или ее блоки на печать или даже представлять информацию графически.

До создания Visi Calc подобного средства на ПК не существовало, вычисления приходилось повторять с самого начала, данные приходилось вводить целиком заново, и такой процесс

продолжался часами и даже сутками. Именно с появлением этой программы *компьютеры стали распространяться повсеместно* как в США, так и в других странах.

В развитых системах такого класса реализована возможность получения ответов на вопросы типа: "Что будет, если..?", – при изменении исходной информации или расчетных формул в таблице. Среди систем этого класса наиболее известны Super-calc-3, Lotus 1-2-3, Excel (Microsoft).

VII. Системы управления базами данных (СУБД)

Представления о том, что такое база данных, вы можете получить, заглянув в свою записную книжку, заполненную однотипными записями, содержащими сведения о фамилиях, именах, отчествах, адресах и телефонах друзей, родственников и знакомых. В необходимых случаях вы обращаетесь к этой "базе данных", чтобы получить те или иные сведения.

Базы данных, создаваемые в запоминающих устройствах компьютеров, подобны картотекам и в отличие от записной книжки могут содержать сотни и тысячи записей, хранящих совокупности взаимосвязанных сведений о тех или иных объектах. Главное преимущество, которое дает переход к автоматизированному ведению базы данных, – быстрый поиск необходимых сведений и представление их в удобной форме. Причем поиск данных в автоматизированной картотеке может

разбить экран на девять равных полей, в каждом из которых будут идти передачи выбранных заранее каналов; или организовать микроэкран, расположенный в нижней части основного экрана, с помощью которого можно контролировать события, происходящие в другой передаче; или разбить экран на два разных поля, в каждом из которых будет изображение, полученное от двух разных источников видеосигнала. Кроме того, видеопроцессор системы позволяет осуществить "заморозку" удачного кадра и цифровое стробирование динамического изображения, при котором экран разбивается на девять областей, в каждой из которых фиксируется очередная фаза движения.

осуществляться не только "по алфавиту" или "по адресу", а по любой совокупности признаков, характеризующих искомые объекты (см. § 36).

СУБД – прикладная программа, позволяющая формировать **базу данных** (БД), производить поиск требуемых данных по информационным запросам, вносить изменения в БД, обрабатывать данные, хранящиеся в БД, и др.

Большинство СУБД, реализованных на персональных компьютерах, поддерживает так называемые **базы данных реляционного типа**. Информация в них хранится сгруппированной в таблицах, каждая из которых имеет свое имя. При задании структуры таблицы определяются количество ее столбцов (полей), имя каждого поля и его тип (числовой, символьный и т.д.). Таблицы информационной базы могут связываться через те или иные поля и обрабатываться совместно.

Иногда СУБД представляют как **информационно-поисковые системы** (ИПС).

Каждая ИПС предназначена для решения определенного класса задач. Для каждого класса характерен свой набор объектов и их признаков. Поэтому соответствующие ИПС будут различными — одна ИПС нужна, чтобы разыскивать книгу по каталогу (в библиотеке), другая – для поиска биографических данных какого-то рецидивиста (в милиции). Впрочем, для разных классов задач объекты могут быть одинаковыми, а наборы нужных признаков – разными. Например, инспектора ГАИ интересует владелец автомобиля, его модель, цвет и государственный номерной знак; налогового инспектора, кроме фамилии, имени, отчества и адреса жительства, – вносимые им налоги и идентификационный код (это так называемый государственный реестр физических лиц).

Каждая ИПС состоит из двух частей: большой, специально организованной совокупности данных (мы ее назвали **базой данных**) и программы, позволяющей оперировать ими. Слово "оперировать" означает находить объекты по заданным признакам, изменять и дополнять сведения об объектах, а также решать иные, самые разнообразные задачи.

В настоящее время наиболее популярны СУБД семейства dBASE III PLUS (фирма – Ashton-Tate), Knowledgeman

(фирма MDBS), R:Base 5000 и CLOUT (фирма Muicrorim), FoxPro (Microsoft), FoxBase (Fox Software), Paradox (Bor-Land), а также русифицированные варианты: Ребус, Карат и Микро-РС-2.

Разговор был бы неполным, если бы в этом разделе мы не упомянули известную программу Clipper.

Компилятор Clipper является независимой от программы СУБД: dBASE III PLUS системой, созданной фирмой Nantucket. Это своеобразный пользовательский интерфейс с элементами операций обработки базы данных, а также специфический язык программирования. Его отличает частичная открытость, т.е. возможность расширения самого языка Clipper специальным образом, посредством собственной системы (Extend System).

VIII. Интегрированные программные средства

При решении многих проблем, например, при создании автоматизированных рабочих мест руководителей и специалистов, требуется использование программных средств нескольких типов. Так как работа с несколькими программными средствами сопряжена со значительными трудностями, прежде всего должно быть их совместное функционирование, а также организация интерфейса (обмена) между ними, то возникла совершенно естественная мысль создать программное средство, объединяющее лучшие достижения в области табличных процессоров и СУБД. Кроме того, существует и другая сторона, подталкивающая к объединению отдельных прикладных систем различного назначения, – сложность при обучении и организации работы пользователей. В связи с этим начали быстро развиваться **интегрированные прикладные системы**, в которых на единой основе объединены текстовые редакторы, электронные таблицы, графические редакторы (чаще всего – деловая графика), СУБД, коммуникационная система связи с другими ЭВМ.

Важнейшими требованиями к новому классу программных средств явились объем используемой памяти ПЭВМ и доступность для конечного пользователя.

Среди систем этого класса наиболее распространены:

Supercalc-3 (разработчик – фирма Computer Associates) (основа его – табличный процессор) позволяет решать многие виды экономических, финансовых, инженерных и научных задач.

Lotus 1-2-3 (создана под руководством Митча Кэйнора на фирме Lotus Develophent). Название этого пакета навеяно мифами Гомера (это описано в его "Одиссее"): поедатели лотоса – народ, который жил в блаженном довольстве и безразличии к окружающему, питаясь только плодами лотоса, современные же "поедатели" – пользователи Lotus 1-2-3, – по замыслу автора, могут весь рабочий день оставаться в среде этого пакета, не испытывая потребности в каком-либо ином программном средстве.

Symphony (те же создатели, что и Lotus) – основным объектом работы остается электронная таблица, но помимо возможностей, имеющихся в Lotus, эта программа предлагает пользователю средства телекоммуникации.

Framework (разработчик – фирма Aston Tate) – интегрированная программа СУБД и текстового процессора.

Samna plus III (фирма Samna corporation) – интегрированный текстовый процессор.

Knowlegmen (фирма Micro Data Base System) – интегрированный пакет СУБД.

Microsoft Office – интегрированный пакет.

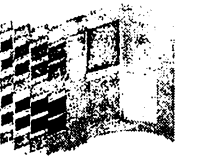
IX. Прикладные среды

Кроме термина "операционная система" введем новое понятие – операционная среда.

ОПЕРАЦИОННАЯ СРЕДА – это весь "фон" системных программ, обеспечивающих взаимодействие пользователя с компьютером: в нее входят ОС, трансляторы с языков программирования и т.п.

Прикладную среду, или, точнее, прикладное окружение, легко принять за часть операционной системы. В действительности же его образуют прикладные программы, которые работают под управлением MS DOS и предоставляют пользователю дополнительные возможности диалога с компьютером, осуществления **мультипрограммирова-**

ния (т.е. многопрограммированного режима), подключения внешних независимых программ и обеспечения графических операций. Единственным существенным недостатком прикладного окружения является то, что для полного использования его возможностей необходимы программы, напрямую взаимодействующие с программами, создающими это окружение.



Windows (фирма Microsoft) – это прикладная среда. Она загружается "поверх" DOS и принимает управление на себя. С помощью Windows можно осуществить одновременное выполнение двух или более программ, а с учетом некоторых, достаточно строгих, ограничений возможен и обмен информацией между программами. Windows – это графический интерфейс, благодаря которому пользователь работает с меню команд и сообщениями системы, а не вводит их с клавиатуры в виде командной строки (термин **"Графический интерфейс"** означает, что информация на экране выводится побитно в графическом режиме). Весь экран в этом случае становится "командной строкой". Вместо того чтобы набирать команды, можно использовать "мышь" или клавиши перемещения курсора для выбора на экране пиктограммы (указателя), обозначающей нужную пользователю прикладную программу.

Преимущество данной среды состоит и в том, что программы, разработанные для этих систем, можно выполнять на компьютерах с самой разной аппаратной конфигурацией. Практически любой пользователь, впервые садящийся за компьютер, без особого труда осваивает первоначальные навыки работы благодаря дружественному диалогу "на языке" картинок и специальных подсказок.

Фирма Microsoff постоянно совершенствует эту среду, добавляет в нее все новые и новые возможности, делая "дружественный интерфейс" с внешней стороны все проще, а по сути – все глубже и серьезнее.

Заключение

Мы привели описание программного обеспечения ПЭВМ **общего назначения**, но ничего не сказали ни о

Известно ли вам, что...

Мысли можно прочесть после смерти

Такую возможность обещают нам английские исследователи на основе нового изобретения в области электроники.

Они утверждают, что узнать о чувствах и мыслях человека на протяжении всей его жизни можно с помощью микрочипа, встроенного в глазное яблоко. По мнению сотрудника компании "Бритиш телеком" доктора Криса Уинтюра, такая конструкция означает ни много ни мало победу человечества над смертью. Вытащив из глаза умершего микрочип, можно будет узнать, о чем думал и что чувствовал некогда живший человек. С помощью подобной техники появится возможность жить жизнью

системах программирования (т.е. трансляторах различных языков, работающих в интерпретирующих или компилирующих режимах), ни об **утилитах базового состава DOS** и **общесистемных утилитах**, ни о **программах обеспечения локальных сетей** ЭВМ и **электронной почты**. Мы также совершенно не коснулись многочисленных специальных прикладных программ: **издательских систем**, **систем автоматизированного проектирования** (САПР), **статистических, бухгалтерского учета** и т. д. и т.п. Мы надеемся, что при необходимости вы сами изучите их.

Программа "Издательская система" заменяет работу всей типографии

Освоив приведенные выше программы общего назначения, вы сумеете создать свою **собственную инструментальную среду** ПЭВМ – это будет ваша среда обитания в общечеловеческом смысле: жилье, магазины, инфраструктура, зоны отдыха, но отнесенная к вашей бытовой или производственной деятель-

ности. Ее можно создать "по кирпичику" – на это уйдет много времени. Под "кирпичиками" понимают готовые программные пакеты, языки программирования и собственные программы.

Если же вы для "города своей мечты" будете брать готовые программы (чтобы успеть в нем пожить), вас не должен интересовать вопрос – как они работают, главное – чтобы они выполняли те функции, которые вам нужны.

Таким образом, собственная ИНСТРУМЕНТАЛЬНАЯ СРЕДА – это совокупность программных средств, работающих под общим управлением на родном языке пользователя и реализующих требуемые функции.

Для создания собственной инструментальной среды после выбора компонентов необходимо решить три проблемы:

1) выполнить при необходимости *адаптацию* выбранных программных компонентов к родному языку;

2) *организовать интерфейс* между программными компонентами;

3) *разработать управляющую программу*, содержащую общее меню функций полученной системы.

В результате такой сверхинтеграции вы достигните следующих целей:

– во-первых – собственная инструментальная среда в такой конфигурации может стать удобным средством для разработки *математических моделей*, описывающих весьма широкий круг объектов и процессов (например, в экономике, планировании и управлении

умершего, используя его опыт. Английские специалисты надеются, что новая технология найдет в будущем самое широкое применение.

. . .

Чемпион мира по шахматам Гарри Каспаров сражался с шахматной программой корпорации IBM, способной просчитывать почти 1 млрд ходов в секунду. Именно такую скорость действий вычислительной машины, носящей название "Дип блю", должна иметь ЭВМ, считают ее создатели, для победы над чемпионом мира, по версии профессиональной шахматной ассоциации. Г.Каспаров никогда не проигрывал компьютерам в серьезных партиях, хотя и уступал иногда машинам. Эта программа прошла "прокатку" на гроссмейстерах в Израиле, кроме того, обыграла шахматистку из Венгрии Юдит Полгар.

И вот теперь новые матчи...

производством, управлении запасами, управлении вашей учебной деятельностью и т.д.);

– во-вторых – построенные с помощью такой среды модели могут стать средством конечного пользователя для решения конкретных задач в составе, например, автоматизированных рабочих мест;

– в третьих – обеспечите будущие расширяющиеся информационные потребности.

Но даже при наличии тысячи прикладных программ, осуществляющих обработку текстов, данных или управление данными, вам все же может потребоваться что-то, чего не делает ни одна из программ коммерческих пакетов. Если вы похожи на большинство других пользователей компьютеров, у вас всегда найдется "еще одна вещь, которую нужно сделать". Часто бывает, что фирменное программное обеспечение не удовлетворяет вашим запросам, и приходится составлять свою программу.

И тем более, если вы хотите создать свою собственную инструментальную среду, вам поневоле приде ся разрабатывать *управляющую программу*. А для этого, как минимум, необходимо знать хотя бы один язык программирования.

Однозначно рекомендовать или спорить о достоинствах того или иного языка программирования – довольно неблагодарное дело. Практически выбор языка программирования определяется исходя из *приобретенного ранее опыта*, имеющейся в наличии *системы программирования* и, конечно, *сложности* поставленной задачи и перспектив дальнейшего ее использования.

Ну а так как мы только приступаем к изучению основ информатики, то все же среди многочисленного количества языков рекомендуем Бейсик. А почему именно этот язык программирования, вы узнаете из следующего параграфа.

Проверьте свои знания ?

1. Назовите программы, которые относят к программному окружению ядра ОС.

2. В чем сущность работы программы-оболочки Norton commander?

3. Зачем нужны программы-архиваторы?

4. Что такое компьютерный вирус? Какова роль антивирусных программ?

5. Каково назначение тестовых и графических редакторов?

6. В чем отличительные особенности электронных таблиц от систем управления базами данных?

7. Что представляют собой интегрированные программные средства?

8. Что такое прикладная среда? Чем она отличается от операционной системы?

9. Как организовать собственную инструментальную среду ПЭВМ?

§ 20. Почему мы избрали Бейсик?

Всех собеседников компьютера мы подразделяем на три большие группы:

1. Системные программисты (их меньшинство) создают программы, облегчающие труд других программистов, например, трансляторы, операционные системы.

2. Программирующие пользователи пишут программы для себя (точнее, для решения своих задач).

3. Программирующие пользователи (их большинство) хотят решать свои задачи на ЭВМ, не желая знать языки программирования; их часто называют конечными пользователями.

Наши подходы в изучении информатики ориентированы именно на вторую группу пользователей. Знание элементов и структуры языка Бейсик необходимо пользователю прежде всего для анализа имеющихся прикладных программ, их корректировки и совершенствования, для организации элементарных вычислений на компьютере в режимах калькулятора и диалога пользователя с программой. Пользователь, владеющий языком Бейсик, не будет довольствоваться только готовыми пакетами прикладных программ. Он всегда сумеет применить компьютер для решения различных задач в своей работе.

Поэтому причины, по которым мы избрали язык Бейсик для изучения основ алгоритмизации и основ программирования, были таковы:

– все современные ЭВМ (в том числе и персональные)

"понимают" этот язык без дополнительных загрузок. Как говорят, язык уже "зашит в сознании", а точнее – в памяти всех типов машин (в ПЗУ). Большинство из них готовы общаться на этом языке при первом включении электропитания;

– этот язык прост по своей структуре и универсален. Он хорош для использования в учебных целях, т.к. обладает такими свойствами, как ясность, простота и согласованность понятий;

– при изучении алгоритмизации нам придется выделять достаточно *общие понятия*, не зависящие от конкретного языка и позволяющие приспособиться к другим языкам. И здесь подходящим языком снова становится Бейсик;

– в нем имеется широкая возможность использовать основные логические управляющие структуры: операторы ветвления и операторы цикла;

– в него, как правило, встраиваются удобные функции для работы с экраном дисплея, клавиатурой, внешними накопителями, принтерами, коммуникационными каналами. Это позволяет относиться к Бейсику, как к "продолжению" аппаратуры ПЭВМ;

– используя этот язык, можно совместить два весьма различающихся между собой этапа: *обучение языку* программирования и *искусство* программирования;

– Бейсик можно рассматривать и как профессиональный язык программирования, подходящий для разработки промышленных программ, позволяющий использовать современные методы и технологию программирования (оговоримся сразу, что в рамках данной книги такая задача нами не ставилась).

Но, пожалуй, сильнейшим аргументом в пользу этого языка является то, что он – один из самых мощных языков, обеспечивающих *доступ* ко всем *возможностям* персонального компьютера – графике, применению цвета и звукового сопровождения, связи с внешними устройствами и манипулированию содержимым экрана.

Бейсик поставляется в придачу к PC-DOS без дополнительной платы (ведь он является собственностью фирмы IBM); неудивительно, что это один из самых популярных языков персональных компьютеров.

ПОДВЕДЕМ ИТОГИ

Программное обеспечение ЭВМ – совокупность всех программ и соответствующей документации, обеспечивающая целесообразное использование ЭВМ в интересах каждого ее пользователя.

Различают внутреннее (системное) и внешнее (проблемное, прикладное) программное обеспечение ЭВМ.

Системное ПО – главными частями его являются операционная система и система программирования.

Операционная система – это важнейшая часть системного программного обеспечения, включающая комплекс системных программ (модулей) управления работой ЭВМ и обработки исходных программ (перевода исходных программ, написанных на языке программирования, в машинную программу, отладки, тестирования и документирования их). ОС включает управляющие и обрабатывающие программы.

Система (автоматизации) **программирования** – совокупность обрабатывающих системных программ и языков программирования, предназначенных для повышения эффективности программистского труда.

Проблемное (прикладное) программное обеспечение предназначено для решения прикладных задач, то есть конкретных задач производственного, научного, управленческого или учебно-тренировочного характера.

К прикладному программному обеспечению относятся библиотеки стандартных программ, пакеты прикладных программ и прикладные программы пользователей.

Транслирующая программа (транслятор) автоматизирует процесс перевода текста программ с одного языка программирования на другой или с конкретного языка программирования на машинный язык.

Компиляция – автоматическое составление машинной программы по исходной программе, записанной на языке программирования, выполняется транслятором-компилятором.

Компилятор – системная программа, осуществляющая трансляцию всей исходной программы в машинную.

Интерпретация – метод выполнения в ЭВМ программы, заданной на языке программирования, без перевода ее в машинную программу. Каждому элементарному действию, описанному на языке программирования, в ЭВМ соответствует своя машинная программа (подпрограмма).

Системная программа-интерпретатор осуществляет последовательное выполнение команд исходной программы без перевода их в машинный код (по мере ввода в ЭВМ).

Интерпретатор – системная программа, осуществляющая синтаксический контроль операторов исходной программы и последовательное выполнение ее команд (операторов).

Ассемблирование – процесс перевода исходной программы, заданной на машинно-ориентированном языке (ассемблере), в машинную.

Эмуляция – автоматическое составление машинной программы для ЭВМ другой архитектуры по исходной программе, заданной на языке программирования, посредством специальной системы (кросс-системы).

Интерфейс – совокупность средств, определяющих логический порядок взаимодействия систем (протокол) и вытекающих из протокола требования к аппаратуре и программному обеспечению, если обмен данными осуществляется под управлением программы.

Языки программирования – искусственно созданные языки для описания алгоритмов решения задач с помощью ЭВМ.

Различают языки низкого и высокого уровня.

Языки низкого уровня, называемые еще машинными языками, – это те языки, которые компьютер воспринимает непосредственно, т.е. это языки машинных команд данной модели компьютера. Программа, записанная на языке высокого уровня, представляет собой набор уже не отдельных машинных команд, а более крупных элементов, называемых операторами данного языка. На языке

высокого уровня исходная программа состоит из последовательности операторов.

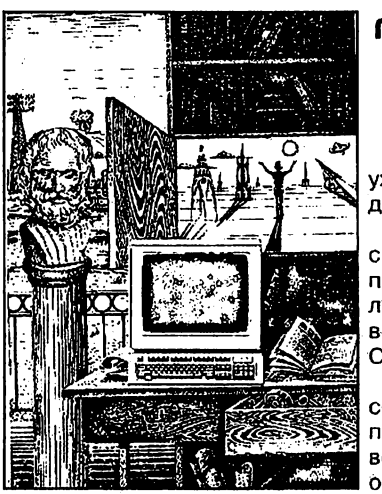
☝ **Набор данных (файл)** – совокупность данных, состоящая из последовательных, логически связанных записей. Файлом может также считаться хранящаяся в памяти ЭВМ программа решения конкретной задачи.

☝ **Пакет прикладных программ** (ППП) – это совокупность сложно организованных машинных программ, дополненная соответствующей технической документацией.

Итак:

☝ **Математическое обеспечение** – средства, которые могут быть представлены пользователю для решения его задач с помощью определенной вычислительной системы или ЭВМ. Оно включает в себя алгоритмическое обеспечение – методы и алгоритмы модели решения задач; лингвистическое обеспечение – языки программирования, программное обеспечение – систему автоматизации программирования; информационное обеспечение – структуру данных и базы данных.

Лицо прошлого с силуэтами будущего



Глядя в лицо прошлому, наметим силуэты будущего

Прошлое... Это бесконечно уходящий в глубь времен фундамент настоящего.

Достижения и открытия, опыт, страдания и радости наших предшественников – все это легло на протяжении тысячелетий в основу нашей цивилизации. Создало ее, сформировало.

Сколько научно-технических сенсаций было отвергнуто и отпало за это время. Сколько утвердилось и стало обыденным, общеизвестным!

Лицо прошлого с силуэтами будущего

Горели на кострах инквизиации бессмертные гиганты, не отрекаясь от своих истин, ставших закономерными сегодня. Сколько прорицательски заглянуло вперед, не в силах осуществить свои замыслы на существующем уровне технологии.

Сколько гениальных открытий, опередивших свой век, ушло в небытие, чтобы, вновь пробившись через столетия, стать ключом к решению тех или иных проблем.

История науки знает немало оракулов грядущего прогресса. Широко известны дерзкие прогнозы Жюля Верна и Герберта Уэллса. В своих утопических романах писатели смело ставили вехи на пути развития науки и техники. Сверхскоростной самолет со стреловидными крыльями – 1970 год. Ошибка в 15 лет. Первые космонавты, покидающие Землю – год 2055-й. Ошибка в 100 лет. Телевидение ("телефот"), пассажирские "аэрокары", летящие со скоростью 600 км/ч, электрические счетно-решающие устройства – всё это должно было появиться не раньше XXIX века. А появилось сегодня. Ошибка ни много ни мало в 900 лет!

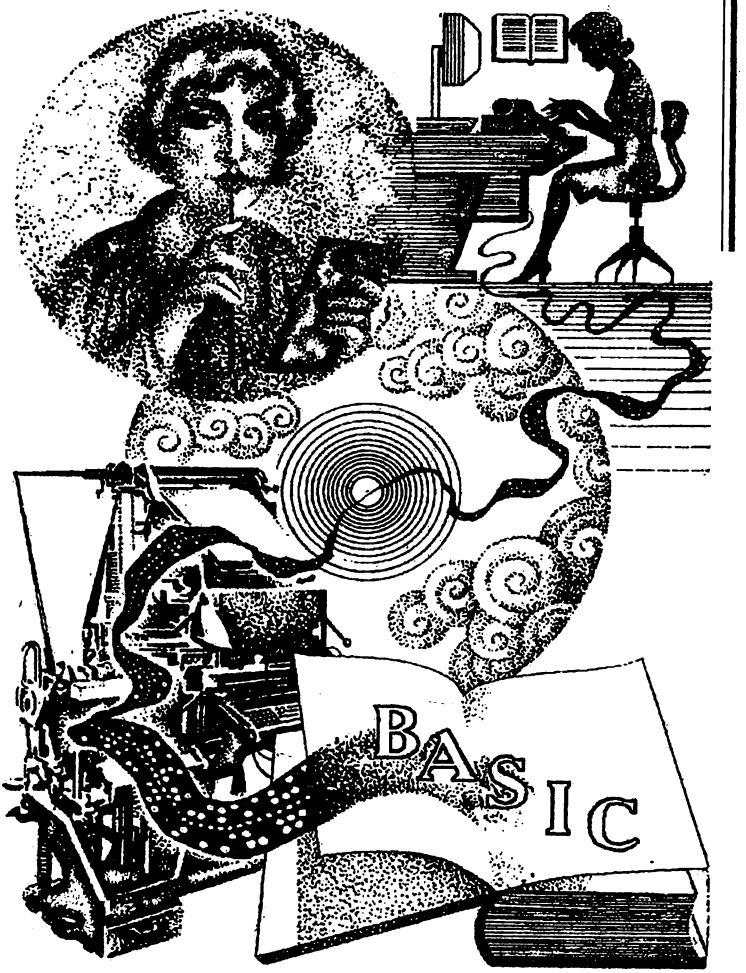
Известный фантаст Артур Кларк в своей книге "Черты будущего" набрасывает контуры прогресса. Он помещает в ней знаменитую таблицу, обошедшую все издания мира. В ней три раздела: свершения науки и техники вчера, сегодня и завтра. В этой книге А.Кларк хорошо проиллюстрировал всемогущество информации. Он высказал возможность мгновенной передачи материальных предметов из одной точки пространства в другую с помощью только информации, да, да, только информации.

Ведь любой предмет есть не что иное, как структурное объединение в определенном порядке определенного числа атомов и молекул. Следовательно, если удастся эту информацию "прочитать" в одной точке пространства и передать в другую, то останется только восстановить по этой информации исходную структуру – и материальный предмет окажется перемещен в пространстве. Такой восстановитель назван репликатором, т.е. восстановителем: он повторяет передаваемый предмет в точке приема. Можно не согласиться с А.Кларком, но сама идея, безусловно, заслуживает внимания, особенно в случае обмена информацией между разными цивилизациями, где другие возможности передачи материальных предметов могут принципиально отсутствовать.

(Продолжение на с.128)

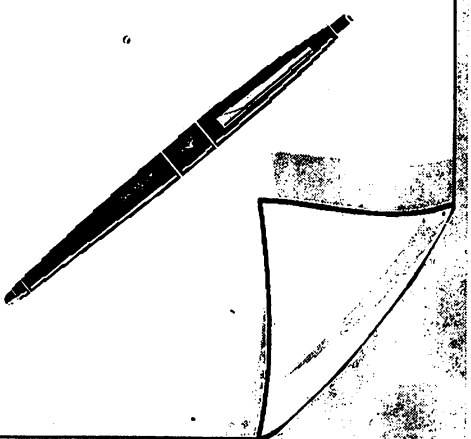
Глава V

НАЧАЛЬНЫЕ ПОНЯТИЯ ЯЗЫКА БЕЙСИК



"Где начало того конца, которым оканчивается начало?"

Козьма Прутков



§ 21. Режим калькулятора

Проще всего с языком Бейсик ознакомиться, непосредственно работая на компьютере.

Но давайте сначала проведем небольшую аналогию с обычным языком, на котором говорят люди. Ведь нам нужно произвести какое-то действие с машиной. Слова, обозначающие действия на естественном человеческом языке, называют глаголами. Например: "высвети", "покажи", "посчитай", "напиши" и т.д. Язык Бейсик устроен по аналогии с обычным языком человеческого общения, но вместо конструкций, указывающих на действия – глаголов, в алгоритмическом языке используется понятие команды. **Команды** – это отдельные фразы языка Бейсик, которые вводятся прямо с клавиатуры и сразу же выполняются ЭВМ. Выполнение команд следует немедленно после нажатия специальной клавиши [↵]. Эта клавиша называется клавишей ввода команд (ВК).

Такой режим, когда введенная в машину команда выполняется сразу, называют **режимом прямого общения** или **режимом калькулятора** (работа с компьютером напоминает работу пользователя с калькулятором).

Например, наберем с помощью клавиш команду

```
PRINT "Я ИЗУЧАЮ ЯЗЫК БЕЙСИК"
```

Затем нажмем клавишу (ВК), и на экране появится фраза:

```
Я ИЗУЧАЮ ЯЗЫК БЕЙСИК
OK
```

Таким образом, по указанной команде машина вывела на экран сообщение, которое было заключено в кавычки. Появившееся после этого сообщения OK (над курсором) свидетельствует о готовности интерпретатора принять следующую команду. Вы, очевидно, догадались, что компьютер при выполнении команды кавычки не выводит, а высвечивает лишь то, что в них заключено.

Команда PRINT может выполнять и арифметические действия и полученные результаты выводить на экран.

Пример:

```
PRINT 17+3 (ВК)
 20
PRINT 2*14-36/3 (ВК)
 16
```

При этом операции сложения и вычитания обозначаются с помощью обычных знаков арифметических действий "+" и "–". Что же касается знаков умножения и деления, то в Бейсике для обозначения этих операций используются соответственно знаки "*" и "/".

А теперь давайте посмотрим, что произойдет, если числа и знаки арифметических действий заключить в кавычки:

Примеры:

```
PRINT "17*4" (ВК)
17*4
```

Лицо прошлого с силуэтами будущего

(Продолжение)

Давайте и мы, используя опыт фантаста, составим таблицу "Лицо прошлого с силуэтами будущего", но возьмем за основу другой подход, ведь речь идет о "Его Величестве Компьютере", а он, как видится автору этих строк, создавался одновременно в двух направлениях.

Первое. Идеи, научные открытия, использовавшиеся потом при создании компьютеров или его программного обеспечения.

Второе. Технические устройства, с помощью которых люди увеличивали скорость счета; машины и механизмы, где использовались принципы программирования или увели-

```
PRINT "744/3+11" (ВК)
744/3+11
```

Как вы уже знаете, кавычки дают компьютеру указание вывести на экран то, что в них заключено. Поэтому он просто печатает **17*4** или **744/3+11**, не выполняя при этом никаких арифметических действий. Следовательно, чтобы такие действия производились, не следует ставить кавычки.

Режим прямого общения полезен тем, что позволяет быстро и легко проверить, как данная команда будет выполняться в конкретных условиях. Он используется для быстрых вычислений, которые не требуют написания целой программы.

§ 22. Организация программ

Вместо того чтобы вводить команды в режиме немедленного исполнения, можно составить из них программу, которая будет выполняться позже в нужный момент. Такой способ работы с ЭВМ называется **программируемым режимом.**

Программа для ЭВМ – это последовательность команд, которые должна выполнять машина. Иными словами, программа для ЭВМ – это алгоритм, записанный на языке, понятном данной ЭВМ.

Лицо прошлого с силуэтами будущего

(Окончание)

чивались возможности обработки, передачи и анализа информации.

Пусть эта таблица даст слабую пунктирную линию пути, по которому прошли люди, и приоткроет завесу неизвестности и таинственности. Думаю, что этот прерывистый, чрезвычайно схематичный след все же не пропадет даром; надеюсь, он проложит тропу к вашему сердцу и вызовет чувство уважения, глубокой признательности и почтения к тем бесконечным искателям, кто, подобрав на тернистой дорожке прогресса камень, дошел до великих откровений, не утомился и продолжает шагать все вперед, все дальше, все выше...

Программа на языке Бейсик представляет собой последовательность команд, записанных в пронумерованных строках. Выполнение программы осуществляется в порядке возрастания номеров этих строк. Чтобы облегчить в процессе составления программы вставление новых строк между уже имеющимися, строки программы обычно нумеруют с каким-либо шагом, например, через 10 номеров: 10, 20, 30 и т.д.

Иногда такие номера называют **метками**. В режиме построчной обработки команды обычно называются **операторами**, хотя часто оба эти термина употребляются как синонимы. Это связано с тем, что большинство команд режима немедленной обработки можно использовать в качестве операторов режима построчной обработки и наоборот. Однако есть такие команды, которые работают только в режиме немедленной обработки, и такие операторы, которые выполняются только в программируемом режиме.

Как и в естественном языке, на котором мы не можем рассказать ничего связного, если не знаем, что такое предложение, так и на языке Бейсик мы не сможем написать программу, пока не поймем, что такое оператор этого языка.

ОПЕРАТОР – конструкция (предложение) для описания величин, организации распределения памяти или для описания логически завершенных этапов процесса обработки информации в ЭВМ. Оператор – это конкретное указание машине, оформленное в виде математической формулы либо в виде обозначения (ключевого слова) какого-то действия, для которого требуется выполнить целый ряд машинных операций.

Одна строка программы может содержать один или несколько операторов. Операторы в строке программы отделяются друг от друга двоеточием. При вводе программы с клавиатуры вы должны заканчивать каждую строку нажатием клавиши ВК (или другой клавиши, принятой для вашего компьютера).

Приведем пример простой программы, состоящей из двух строк:

```
10 PRINT "УМНОЖИТЬ 17 НА 3"
20 PRINT 17*3
```

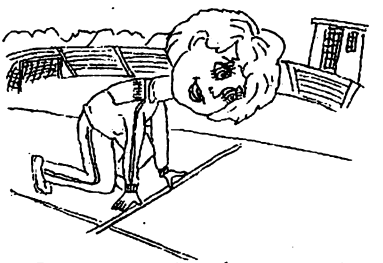
Каждая строка этой программы содержит по одному оператору PRINT и является отдельной инструкцией компьютеру. Номера строк 10 и 20 указывают, в каком порядке выполнять эти инструкции.

§ 23. Основные системные команды языка Бейсик

Системные команды не заносятся в память ЭВМ и выполняются сразу после ввода.

Команда RUN (выполнять)

После набора простейшей программы компьютер записывает введенные операторы в оперативную память машины. Операторы могут вводиться последовательно, начиная с младших номеров, но могут вводиться и в любом порядке. В последнем примере мы могли сначала ввести строку 20, а затем строку 10. Строки выводятся на экран, но компьютер не выполняет программу до тех пор, пока вы не прикажете ему, введя команду Run.



По команде RUN (бежать) программа начинает выполняться

RUN – системная команда, по которой компьютер выполняет инструкции, содержащиеся в программе.

Давайте посмотрим, что произойдет с программой после ввода команды RUN и нажатия клавиши ВК:

```
10 PRINT "УМНОЖИТЬ 17 НА 3"
20 PRINT 17*3

RUN (ВК)
УМНОЖИТЬ 17 НА 3
 51
```

Компьютер выполняет все инструкции программы по одной в каждый момент времени в порядке возрастания номеров строк.

Если требуется "прогнать" программу второй раз, то для этого надо еще раз набрать Run, при этом исключается необходимость вновь вводить программные строки, т.к. они хранятся в оперативной памяти. Например:

```
100 PRINT "ПРОИЗВЕДЕНИЕ 2*3 РАВНО:"
130 PRINT "ЭТО ПРАВИЛЬНО?"
120 PRINT 2*3

RUN (ВК)
ПРОИЗВЕДЕНИЕ 2*3 РАВНО:
 6
ЭТО ПРАВИЛЬНО?
```

Команда LIST (просмотр текста программы)

После того, как программа введена (или только создается), можно потребовать, чтобы компьютер высветил на экране те строки, которые находятся в оперативной памяти. Для этого набирают команду LIST и нажимают клавишу ВК. При этом компьютер выводит на экран полный текст программы, т.е. весь список (отсюда list – список).

Для вывода отдельного оператора программы в команде List указывается номер соответствующей строки. Пример такой команды:

Технические устройства и механизмы

30 тыс. лет до н.э.

Обнаружена в раскопках так называемая "*вестоницкая кость*" с зарубками. Позволяет историкам предположить, что уже тогда наши предки были знакомы с зачатками счета.

```
LIST 120 (ВК)
```

Результатом будет появление на экране строки с номером 120:

```
120 PRINT 2*3
```

Для вывода на экран части программы в команде List нужно указать номера первого и последнего операторов, разделяя их знаком тире. Пример:

```
LIST 100-120 (ВК)
```

Результат:

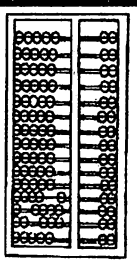
```
100 PRINT "ПРОИЗВЕДЕНИЕ 2*3 РАВНО:"
120 PRINT 2*3
```

Значение верхней или нижней границы в команде можно не указывать, тогда компьютер по умолчанию выводит на экран либо начало программы до какой-то строки (List-120 распечатывает инструкцию с номерами меток до 120), либо конец программы (List 120- – распечатывает инструкцию с номерами меток, начиная со 120).

Команда List полезна в ситуациях, когда в программу надо внести какие-либо исправления или изменения, то есть говорят "нужна для редактирования".

Редактирование внутри строки проводится с помощью курсора. Для этого курсор перемещается на нужную строку, затем на нужную позицию, где и вносятся нужные исправления. По окончании исправлений необходимо обя-

Технические устройства и механизмы



VI-V вв. до н.э.

Появился, пожалуй, первый вычислительный прибор – "*саламинская доска*" по имени острова Саламин в Эгейском море, греки называли его "АБАК". Он был широко известен у китайцев под названием "СУАН-ПАН", у японцев – "СЕРОБЯН".

зательно нажать клавишу ВК. В противном случае ЭВМ не запомнит исправления.

Команда DELETE (удаление строк)

Довольно часто случается, что из программы необходимо удалить одну, а то и несколько строк. Для удаления одной строки есть самый простой способ – это набрать ее номер и нажать клавишу ВК. Но существует для этого и специальная команда Delete. Она управляется точно так же, как и команда List, но производит не распечатку, а удаление указанных строк. Например, если нужно удалить строки с номера 120 по 130, то нужно ввести команду

```
DELETE 120-130
```

Удаление из памяти всего текста программы осуществляется командой

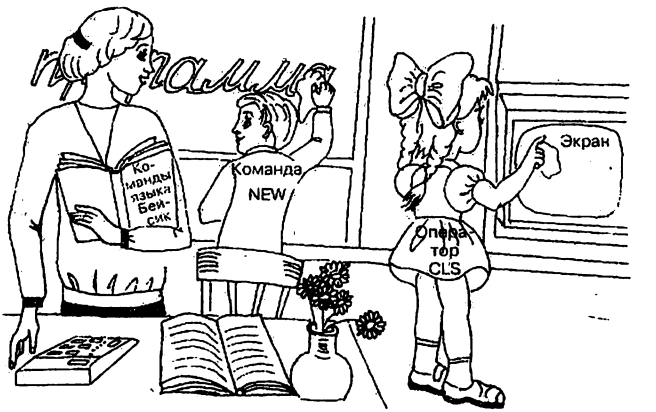
NEW (создание новой программы)

При выполнении этой команды компьютер полностью очищает область оперативной памяти, где ранее хранились программа и другие данные к ней. Эту системную команду обычно применяют в том случае, когда надо начать новую программу. В противном случае может оказаться, что среди строк новой программы попадутся строки старой или будут несуразно использованы какие-либо данные другой программы.

Технические устройства и механизмы

Конец VI в.

Первые упоминания о *механических часах*. Изобретение их приписывают Пацификусу из Вероны (нач. IX века). А лишь в 1276-1277 гг. испанские ученые впервые описывают их. Достоверно известно, что простейшие механические (башенные) часы построены в 1335 г. в Милане.



В отличие от команды очистки машинной памяти, существует команда "очистки экрана"

CLS (очистить экран)

При ее наборе компьютер полностью очищает экран от графической и текстовой информации (но подчеркнем, что только с экрана, а не с области памяти). Эта команда используется и как оператор. Если во время выполнения программы компьютер доходит до строки с оператором CLS, он стирает все, что в этот момент находится на экране. Поэтому необходимо, чтобы в начале каждой программы в первой строке был оператор CLS. Благодаря этому происходит автоматическая очистка экрана перед выполнением программы.

Технические устройства и механизмы

Конец XV-начало XVI в.

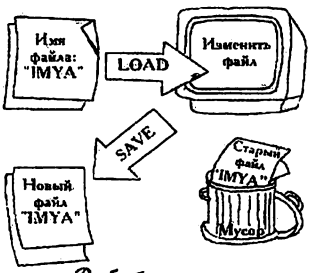
Великим творцом эпохи Возрождения Леонардо да Винчи был дан эскиз 13-разрядного суммирующего устройства с десятизубными колесами. По его чертежам в наши дни американская фирма по производству компьютеров IBM в целях рекламы построила работоспособную машину.

Команда SAVE (записать текст программы) и LOAD (загрузить программу)

Для сохранения текста программы, который мы ввели в область программ интерпретатора, служит команда Save "имя программы". Особенно это важно, когда мы выключаем электропитание ЭВМ или просто заканчиваем работу с подготовленной и отредактированной программой.

Например:

```
SAVE"IMYA"
```



Работа команд LOAD и SAVE

При выполнении этой команды компьютер запишет текст программы на внешний носитель.

А чтобы, наоборот, загрузить эту программу с внешнего носителя в ОЗУ ЭВМ, служит команда Load"имя программы".

Например:

```
LOAD"IMYA"
```

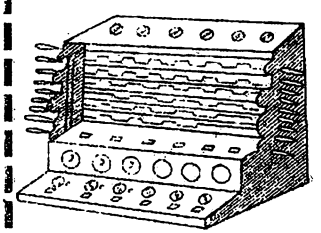
С помощью этой команды можно не только извлечь программу из внешней памяти, но и выполнить ее. Для этого нужно в конце команды поставить запятую и букву R.

Например:

```
LOAD"PROG",R
```

В данном случае говорят, что эта команда становится неявной частью команды Run (отсюда символ R).

Технические устройства и механизмы



1623 г.

Вильгельм Шиккард – профессор Тюбинского университета – в письмах И.Кеплеру описал устройство "часов для счета" – счетной машины. В наше время по его описаниям построена ее модель.

Проверьте свои знания ?

1. В чем состоит суть режима прямого общения? Как его реализовать? Когда его удобно использовать?

2. Что такое программируемый режим работы компьютера?

3. Что такое программа для ЭВМ?

4. В чем состоит единство и различие между терминами "команда" и "оператор"?

5. Что выведет на экран дисплея компьютер, если выдать ему следующие задания:

PRINT "РАБОТАЕМ В РЕЖИМЕ КАЛЬКУЛЯТОРА"
PRINT 12/3
PRINT "5*7"

6. Есть ли ошибки в записи следующих команд; если есть, то в чем они состоят?

PRINT ВЫ ВНИМАТЕЛЬНЫЙ ПОЛЬЗОВАТЕЛЬ
PRINT СУММА ЧИСЕЛ РАВНА A+B"
PRINT 4x8:2

7. Посмотрите на программы

160 PRINT "НОМЕРА СТРОК"
120 PRINT "ИНТЕРПРЕТАТОР БЕЙСИКА"
140 PRINT "ОПЕРАТОРЫ"
150 PRINTT "В ПОРЯДКЕ"
130 PRINT "РАЗМЕЩАЕТ"

и ответьте на поставленные вопросы:

а) Что выведет на экран компьютер при выполнении команды RUN?

б) В каком порядке будут появляться программные строки при выводе их на экран с помощью команды LIST? Команды LIST-140? Команды LIST 130-; LIST 130-150?

в) Что произойдет, если набрать команду CLS?

г) Назовите два способа удаления строки 160.

д) Какую нужно произвести операцию, чтобы удалить из программы текст: "РАЗМЕЩАЕТ ОПЕРАТОРЫ"?

е) Что произойдет, если набрать команду DELETE 140-160?

ж) Что произойдет по команде NEW?

8. Чем отличаются системные команды от обычных команд языка Бейсик? Приведите примеры таких команд.

9. Расскажите о команде записи текста Бейсик-программ на внешний носитель.

10. Какую необходимо дать команду компьютеру, чтобы извлечь программу из внешней памяти и выполнить ее?

§ 24. Алфавит

Как всякий язык, Бейсик имеет свой алфавит, то есть символы, которые составляют его основу. Так как Бейсик является англоязычным языком программирования, все его команды записаны буквами латинского алфавита, причем эти буквы могут быть как заглавными, так и прописными.

В алфавит языка входят:

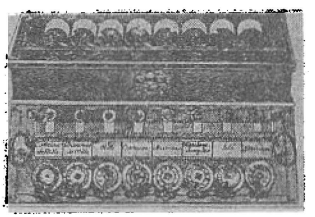
1. 26 букв латинского алфавита от A до Z.
2. Цифры 0 1 2 3 4 5 6 7 8 9.
3. Знаки арифметических операций:
   + сложение;
   − вычитание;
   * умножение;
   / деление (\ целочисленное деление, т.е. деление без остатка);
   ^ возведение в степень.
4. Знаки операций отношения:
   < меньше;
   > больше;
   = равно.

Комбинируя эти знаки, получим:

>= или => больше или равно (в математике $\geq$);
<= или =< меньше или равно (в математике $\leq$);
<> или >< не равно (в математике $\neq$).

5. Знаки:
   . точка;
   , запятая;

Технические устройства и механизмы



1642 г.

Молодой 18-летний французский ученый Блез Паскаль создает *первую модель вычислительной машины*. Эта машина умела выполнять сложение.

; точка с запятой;
‘ апостроф;
“ кавычки;
: двоеточие;
! восклицательный знак;
? вопросительный знак;
(круглая скобка левая;
) круглая скобка правая;
пробел (не имеет начертания).

6. Специальные знаки:
$ знак денежной единицы (иногда он заменен на знак “солнышко” ¤);
& коммерческое “и” – амперсанд;
@ коммерческое “эт”;
# номер (решетка или диез);
% процент.

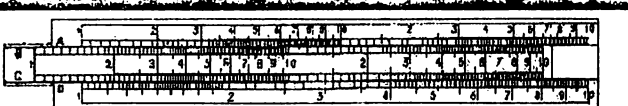
7. Буквы украинского или русского алфавита от А до Я. Отдельные версии языка допускают использование других символов.

В комментариях и текстовых константах допускается использование любых символов клавиатуры, в том числе и украинских (русских) букв.

Букву О легко перепутать с цифрой 0, поэтому в программах и на экранах дисплея ноль, как правило, перечеркивается **0**.

В отличие от обычного языка “пробел” – такой же символ алфавита, как и другие видимые символы. На экране или на бумаге он не отображается, просто одна позиция остается незаполненной.

Технические устройства и механизмы



1654 г.

Англичане Роберт Биссакар, а в 1657 г. – независимо от него – С.Патридж разработали прямоугольную логарифмическую линейку, конструкция которой в основном сохранилась до наших дней.

§ 25. Конструкции языка Бейсик

Из символов алфавита конструируются все его объекты: константы, переменные, массивы, функции и выражения.

Константы: в Бейсике существует два типа констант – числовые и символьные (или литерные).

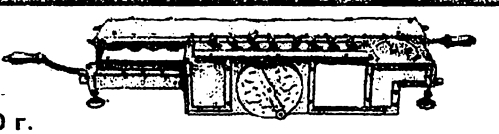
а) **Числовые константы** (или просто **числа**)

В языке Бейсик используются целые и действительные числа. Знак "+ " перед числом не ставится (его можно опускать только у положительных чисел), а для отделения целой части при записи десятичных дробей используется точка. Нулевую целую часть дробного числа можно опускать.

Отличительным признаком целой константы является знак %, который не имеет к исчислению процентов никакого отношения, например: 36% и 7% лишь означают запись целой константы. Создатели умышленно ввели понятия целых чисел. Операции над целыми числами происходят быстрее, чем над действительными, но главное – целое число занимает в памяти в 2 раза меньше места.

Действительные числа могут представляться в экспоненциальной форме. Так, например, одно и то же число 0,0007435 может быть представлено несколькими способами: .7435E–3, или .000007435E2, или 74.35E–5. Здесь латинская буква "E" имеет смысл "возвести 10 в степень". Для положительного порядка степени знак "+" может отсутствовать, для отрицательного знак "–" обязателен.

Технические устройства и механизмы



1670 г.

Замечательный немецкий математик Готфрид Вильгельм Лейбниц сначала описал, а в 1692 г. построил *механическую счетную машину*, которая могла не только складывать и вычитать, но и умножать и делить.

Целые:	Действительные		Десятичные в форме Е
5%	.05	→	5E-02
100 %	-9.083478	→	-.9083478E+1
-834%	37.	→	3.7E1
1275%	+340.033	→	0.340033E+3

б) **Символьные (литерные) константы** – последовательность любых отображаемых символов, заключенных в кавычки. Например: "Информатика", "Результат тестирования", "Temterature 10.03. 1994". В цепочке символов могут быть любые символы языка (запятые, точки, буквы как латинского, так и украинского (русского) языка, пробелы и т.п.), кроме кавычек. Допускается случай, когда литерная константа не содержит ни одного символа; соответствующий текст (пусто) обозначается двумя кавычками, следующими друг за другом " ".

Переменные

ПЕРЕМЕННЫЕ – это величины, значения которых могут изменяться во время выполнения действий.

Как и константы, переменные в Бейсике бывают числовые и символьные. Переменная имеет имя (его называют **идентификатор**).

Обращение к переменной происходит по имени.

Например, мы говорим:

Технические устройства и механизмы

1801-1804 гг.

Часы с боем, шарманки, музыкальная шкатулка. Все эти предметы объединяет одно – они *работают по программе*. Это особенно удивительно, если вспомнить, что во время их создания о программировании никто еще не догадывался. В часах с боем "программа" представляет собой специальное колесо, запускающее

"X имеет значение 4.5".

Хотя на самом деле имеем в виду:

"X – это имя места памяти, где в данный момент хранится значение 4.5".

Во втором высказывании мы четко провели различия между тем, где содержится, и тем, что содержится.

Таким образом, имя переменной – это "где хранится", а значение переменной – "что хранится".

Присваивая новое значение переменной, мы теряем старое.

Рекомендуется имена переменных давать по смыслу содержащихся в них значений. Например, переменную для хранения суммы обозначить S, а переменную скорости – через V.

Имя числовой переменной состоит из одного или двух символов. Первый из них обязательно должен быть буквой латинского алфавита, второй – буквой этого же алфавита или цифрой. Например: A, A8, C7. Символьные переменные после имени содержат знак ¤ (или знак $). Например: A¤, AK¤, B2¤.

Числовые переменные, как и сами числа, могут быть двух типов: целые и действительные. Значениями целых переменных всегда являются целые числа, значениями действительных – действительные. Признаком целой переменной в ее обозначении является присутствие символа %, располагаемого вслед за именем переменной. Например: I%, X5%, K% – простые переменные целого типа.

А вот в следующих выражениях допущены ошибки:

(–B+D) /2A – отсутствует знак операции между 2 и A;

Технические устройства и механизмы

(Окончание)

в определенное время ударный механизм, отбивающий число часов. В шарманке и музыкальных шкатулках "программа" записана в виде штырьков, расположенных на валу. При вращении вала штырьки задевают пластинки, звучание которых сливается в стройную мелодию.

В 1801 году французский изобретатель Жозеф Мари Жаккард создал машину для выработки крупноузорных тканей. Для управления нитями в ней применялись специальные перфокарты с отверстиями.

(R+S*A) (Sin(2.*A) – нарушено соответствие между скобками, отсутствует знак умножения;

Q**5–1,7 – два знака операций стоят рядом, запятую в константе 1.7 употреблять нельзя.

Особое внимание следует обращать на операцию возведения в степень. Возведение нуля в нулевую или отрицательную степень, отрицательного числа в степень с дробной частью воспринимается как ошибка.

Группе переменных одного типа может быть присвоено общее имя, их в этом случае называют переменными с индексами или массивами.

Массивы (см. также § 34)

Для обращения к каждой отдельной переменной в массиве используют один или несколько индексов. Наименьшее значение индекса равно 0, а наибольшее определяется размером массива. Если индекс один, то говорят, что массив одномерный, два – двухмерный. Имена массивов подчиняются тем же правилам, что и имена переменных. Индексы необходимо указывать в круглых скобках после имени массива, разрешено использование массивов как числовых, так и символьных переменных.

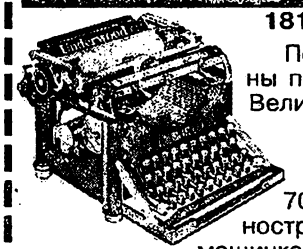
Например:

AC (3) – третий элемент одномерного массива;

LS (5,8) – элемент, стоящий на пересечении 5-й строки и 8-го столбца двухмерного массива LS;

C$(M,N) – элемент, стоящий на пересечении M-й строки и N-го столбца двухмерного символьного массива.

Технические устройства и механизмы



1812-1814 гг.

Первые действующие печатные машины построены Ф.Кенигом и А. Бауэром в Великобритании.

А в 1867 г. американский топограф К.Л.Шоулз изобретает первую практическую пишущую машинку, которую с 70-х годов широко производит машиностроительная фабрика Ф.Ремингтона, и машинка получает наименование "Ремингтон".

• При работе программы для каждого массива в памяти ЭВМ резервируется соответствующая область. Перед использованием массив должен быть описан при помощи оператора DIM.

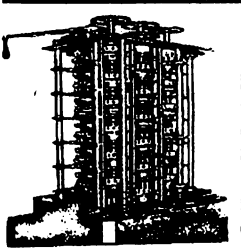
Знаки операций

В Бейсике все операции делятся на арифметические операции, операции отношения и логические операции (табл 7).

Таблица 7

Знаки операций	Операции
Арифметические операции:	
+	сложение
–	вычитание
*	умножение
/	деление
^	возведение в степень
Операции отношения:	
>	больше
<	меньше
=	равно
<>	не равно
>=	больше или равно
<=	меньше или равно

Технические устройства и механизмы



1823 г.

Английский ученый Чарльз Беббидж разрабатывает проект "Разностной машины", предвосхитившей современную программно-управляющую автоматическую машину. Затем он предложил схему "аналитической машины", которая должна была заменить человека в одной из самых медленных операций его ума".

Окончание табл. 7

Логические операции:	
NOT	отрицание ("НЕ")
AND	логическое умножение ("И)
OR	логическое сложение ("ИЛИ")
XOR	исключающее сложение ("ЛИБО")
EQV	эквивалентность
IMP	импликация

При вычислении результата выражения все операции имеют определенный приоритет (табл. 8):

Таблица 8

^
*, /
+, −
=, <>, >, <, >=, <=
NOT
AND
OR
XOR
EQV
IMP

Чем выше в таблице находится знак той или иной операции, тем выше ее приоритет. Для символьных переменных и констант разрешены только операции отношения и **конкатенации** (слияния), обозначаемые знаком "+". При конкатенации один текст без пробела подсоединяется к другому.

Технические устройства и механизмы

(Окончание)

"Аналитическая машина Ч.Беббиджа предполагала в своем устройстве три основные части: "склад" для хранения чисел, набиравшихся с помощью зубчатых колес; "фабрику" для операций над числами, изъятыми из "склада"; устройства для управления операциями с помощью перфокарт.

Например, пусть:

T$ = "ТЕРМО", D$ = "ДИНАМИКА"
C$ = "СИСТЕМА", S$ = "СТАТ"

Тогда:

T1$ = T$ + D$ = "ТЕРМОДИНАМИКА"
T2$ = T $+ C$ = "ТЕРМОСИСТЕМА"
T3$ =T$ + S$ = "ТЕРМОСТАТ"

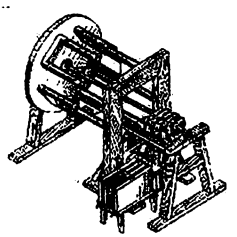
Стандартные функции для числовых величин

В языке Бейсик имеется ряд встроенных функций, которые позволяют значительно упростить написание некоторых программ. Название "встроенная" означает, что в интерпретаторе есть программа обработки данной функции. Для вызова функции достаточно указать ее имя в выражении. В круглых скобках указывают аргумент (табл. 9).

Таблица 9

Запись на языке Бейсик	Математическое определение и результат работы
sqr(x)	$\sqrt{x}$ – корень квадратный из x
exp(x)	e – экспоненциальная функция e
log(x)	lnx – натуральный логарифм от x
abs(x)	$\lvert x \rvert$ – абсолютная величина x $abs(x)=\begin{cases} x, \text{ если } x>0 \\ 0, \text{ если } x=0 \\ -x, \text{ если } x<0 \end{cases}$

Технические устройства и механизмы



1834 г.

Русский ученый Б.С.Якоби построил один из первых практических *электромоторов постоянного тока – электродвигатель* с вращающимся рабочим валом.

Окончание табл. 9

Запись на языке Бейсик	Математическое определение и результат работы
sgn(x)	Знак $\text{sgn}(x)=\begin{cases}1, \text{если } x>0\\0, \text{если } x=0\\-1, \text{если } x<0\end{cases}$
sin(x)	sinx – синус от x (x в радианах)
cosx	cosx – косинус от x (x в радианах)
tan(x)	tgx – тангенс от x (x в радианах)
atan(x)	arctgx – арктангенс x (результат x в радианах)
int(x)	наибольшее целое, не превосходящее x (целая часть от x)
rnd(x)	датчик случайных чисел (случайное число в диапазоне от 0 до 1)
pi	π – число = 3,1415926

Проверьте свои знания

1. Какие группы символов составляют алфавит языка Бейсик?
2. Какие типы констант используются в языке Бейсик?
3. Что является отличительным признаком целых констант?
4. Каково назначение имени переменной?
5. Может ли имя переменной начинаться буквами Ж, З, И, Л, Н, У, Ф, Ц, Ч, Ш, Щ, Э, Ю, Я? Почему?
6. Как по имени определить тип переменной?
7. В каком типе переменных используется операция конкатенация?
8. Перечислите различные типы выражений в языке Бейсик.
9. Какие операции разрешены над символьными константами и переменными?

Тренировка

I. Определите типы констант, входящих в список:

а) 13%;
б) 156.24;
в) "Информатика";
г) "Доход 25%";
д) .134E–1;
е) "100%".

II. Найдите ошибку в записи констант:

а) 2,718282; б) Закон; в) 2O4.57;
г) 0.034E-73; д) "Литерная" константа".

III. Найдите ошибки в записи имен переменных:

а) QЯ; в) TOR; д) Л2;
б) 2R; г) A[X]D; е) 45Z.

IV. Представьте следующие числа в экспоненциальной форме:

а) 0,0038; б) −173,056; в) 10^{-5}; г) $-12,74*10^{4}$;
д) $-0,0489*10^{-5}$

V. В данной цепочке записи числа есть ошибка. Найдите ее:

0.073514=.00073514E2=7.3514E−2=735.14E−3=0.73514E−1=
1 2 3 4 5
.73514E−1.
6

VI. Запишите арифметические выражения на языке программирования:

а) $x^2 + y^2 - z^2$; б) b + 3,14c; в) $a^2 + b^2 - 2abcos\beta$
г) 16: (c−d)k; д) $a_0 + y(a_1 + y(a_2 - a_3y))$
е) $\ln|x-1| - s\left(\frac{1}{2+x} + \frac{\sqrt{b}}{tgx}\right)$; ж) $(0,237 - bsin\sqrt{x})^3 - 4^{cosx}$

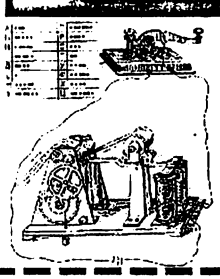
VII. Запишите на языке обычных математических обозначений:

а) X^2/(A/B*COS(X)) − (SQR(Y^2 − 1))/(C*K − LOG(ABS(X + 3)))+
+ 3*B*K^3*SQR(TAN(X^3) − 2);

б) 5*C(I,J) − 3*B/N + EXP(X−1)*SQR(SIN(X−Y)^2−1) +
(1 − ABS(X^2 − Y^2)/EXP(COS(I));

в) X^(Y^Z) + .017*B*C − ATAN(X) + (SIN(X)^2 + COS(X)^2)/
(TAN(X) −1).

Технические устройства и механизмы



1835 г.

Американский изобретатель и художник Сэмюэл Морзе создает и широко внедряет в практику телеграфный аппарат и линии связи. Он же разрабатывает кодирование букв, цифр и знаков препинания набором точек и тире – азбуку Морзе.

VIII. Не изменяя результата, перепишите выражения, убрав лишние скобки:

а) (X*Y)/Z; б) (A+B)*(C/X); в) (K+Y^(1+N)*(K+N));
г) (X*(Y/(A*B/(K*N)))).

§ 26. Оператор LET (присвоить)

Со знаком присваивания ("=") мы с вами уже знакомы. Оператор LET работает по тем же законам, что и знак присваивания.

По словарю Даля "присвоить" – означает "завладеть чем-то, захватить чужое, отбить, отнять". В программировании "присвоить" означает нечто совсем иное, скорее что-то вроде "поделиться", "обменяться", "скопировать".

Например, в строке

```
10 LET X=4.7
```

не обычное равенство, а указание присвоить X значение 4.7 или, другими словами, записать 4.7 в то место памяти ЭВМ (его еще называют ячейкой памяти), которое выделено для хранения численного значения переменной X. Итак:

Оператор LET дает компьютеру указание записать данные в некоторую переменную.

Посмотрим, как работает этот оператор:

```
10 LET X=4.7
20 PRINT X
```

Технические устройства и механизмы

1836-1839 гг.

Французский штабной офицер и изобретатель Жозеф Нисофор Ньепс и парижский художник-декоратор Луи Жак Дагер открыли *фотографию* (светопись). А в 1868 г. чешский художник, ученый и изобретатель Я.Гусник изобретает *фототипию* (репродукция изображений). Независимо от него в 1869 г. в России фотограф В.Я.Рейнгард и физик К.Д.Низовский изобретают фототипию (русское название "светопись").

```
30 STOP
RUN
 4.7
```

По команде LET компьютер записал значение 4.7 в переменную X, и это было проверено с помощью оператора PRINT, который дает компьютеру задание показать, какое значение хранится в X. В результате было выведено число 4.7. (оператор STOP останавливает выполнение программы).

Причем команду RUN можно давать компьютеру многократно, результат будет тот же. Следовательно, содержание ячейки при выполнении оператора LET не уничтожается и его можно читать много раз.

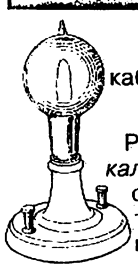
Строка

```
25 LET X=Y
```

предусматривает более сложные действия: найти в оперативной памяти место, отведенное под переменную Y, прочитать содержимое этой ячейки и переслать его туда, где должна храниться переменная X. Причем содержимое ячейки Y, откуда считывается число, тоже не изменяется, не "портится".

Более сложный пример:

Технические устройства и механизмы



1866 г.
Вступили в эксплуатацию две трансатлантические кабельные линии между Англией и США.

1872 г.
Русский ученый А.Н.Лодыгин изобретает *лампу накаливания*, а в 1879 г. американский ученый Эдисон создает лампу накаливания с угольной нитью достаточно долговечной конструкции, получившую широкое распространение.

```
27 LET X=3*A+B/4
```

На первый взгляд кажется, что здесь записано самое обыкновенное уравнение: x=3a+b/4. Но это не так. Чтобы убедиться, можно в этой строке переставить местами левую и правую части. Введите ее в компьютер и дайте команду на выполнение – на экране дисплея появится сообщение об ошибке. А дело в том, что строка 27, так похожая на равенство, почти ничего не имеет с ним общего. Это зашифрованное и краткое указание ЭВМ провести ряд операций: найти и считать в выделенных местах памяти значения переменных a и b, произвести умножение и деление их на константы 3 и 4, сложить полученные результаты и, наконец, полученную сумму присвоить X – записать в то место памяти, где хранится переменная X. Все эти действия производятся автоматически.

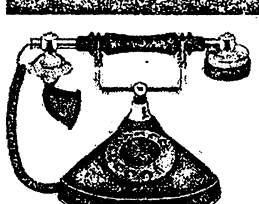
В программировании знак равенства не имеет того смысла, какой он имеет в математике, что хорошо видно на таком примере:

```
50 LET S=S+1
```

Это означает указание ЭВМ взять значение переменной S, прибавить к нему единицу и заслать в то же место, где хранилось старое значение S. Если переменная S имела, допустим, значение 5, то после выполнения этой команды значение S станет равным 6, а его старое значение будет потеряно.

Этот прием широко используют программисты, когда необходимо ввести подсчет каких-то значений или опера-

Технические устройства и механизмы



1875 г.

Лондонский инженер У.Смит изготовил первый в мире полупроводниковый прибор – *фотометр*.

1876 г.

Американский изобретатель А.Г. Белл получает патент на изобретение *телефона*.

ций. Они говорят, что нужно "завести счетчик". А вот с точки зрения математики, операция S=S+1 вообще бессмысленна.

Сформулируем правило оператора LET:

Оператор присваивания должен записываться в строго определенном порядке – после номера строки и LET (что не обязательно) обозначение (имя) переменной, затем знак равенства, а справа от него – числа (константы), переменные и арифметические выражения.

Оператор LET можно также использовать и для присваивания значений строковых переменных. Например, оператор

```
60 LET Z$="ЗНАЧЕНИЕ"
```

предлагает компьютеру запомнить строку из букв "значение" в ячейке памяти с именем Z$. Напомним, что символьные строки всегда должны быть заключены в кавычки, а имена таких переменных должны иметь дополнительный опознавательный символ "$". Необходимо следить, чтобы тип переменной строго соответствовал типу выражения присваивания.

Служебное слово LET писать необязательно. Практически все используемые версии Бейсика имеют эту возможность.

Проверьте свои знания ?

1. Какой вид имеет оператор присваивания?
2. Каково значение символа "=" в записи оператора LET?
3. Каков порядок выполнения команды присваивания?
4. Какое существует соответствие между типом переменной в левой части и значением выражения в правой части равенства оператора присваивания?

Тренировка

1. Присвойте переменной X значение следующих выражений:

а) (a+z) (a−z); б) $y^2 - 3kn$; в) x−10sinx.

II. Найдите ошибки в записи операторов присваивания:

а) 3X = X^3;
б) L+K = Y;
в) A1:=A1/B+K;
г) Q=Q!;
д) R(2.8)= 2*Pi*R;
е) D¤ = L¤; H¤; 2.8;
ж) Л3=45;
з) i%= "ИМЯ";
и) S¤ = S% +1;
к) X = 5Y - COSX.

III. Проследите, как будут меняться значения переменных C и D при выполнении этих программ (устно):

а)
```
NEW
10 LET C=2
20 LET D=4
30 LET C=C+D
40 LET D=1
50 LET C=C*D+D
```

б)
```
NEW
10 C=0: D=5
20 D=D+COS(C): C=SIN(C)
30 D=C+2*D: C=D/6
```

IV. Напишите операторы для выполнения действий:

а) замените текущее значение переменной Y на значение тангенса этой величины;

б) найдите среднее значение величин A и B, результат умножьте на косинус аргумента X и полученный результат присвойте элементу, стоящему на пересечении 5-й строки и 8-го столбца двухмерного массива LS;

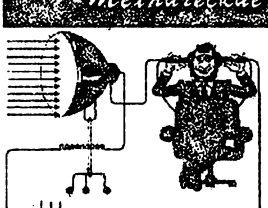
в) найдите символьную величину, равную конкатенации двух величин "АЭРО" и "ДИНАМИКА", и занесите в третий элемент одномерного символьного массива A.

V. Какое значение получит переменная X в результате выполнения таких программ:

а)
```
10 LET X=0
20 LET A=2
30 LET B=3
40 LET C=5
50 LET  Y=A−COS (X)
60 LET X=A*Y^2+B*Y+C
```

б)
```
10 LET X¤ = "ТЕРМОС"
20 LET T¤ ="ТЕРМО": LET G¤
   ="ГИДРО"
30 LET H¤ =T¤ + G¤
40 LET X¤ =H¤ + "СТАТ"
```

Технические устройства и механизмы



1878 г.

А.Г.Белл совместно со своим помощником Тейтером провел первый в мире сеанс *беспроволочной связи* на расстоянии 213 метров с помощью *фотофона* – устройства, соединяющего в себе фотометр и телефон.

§ 27. Оператор PRINT (печать на экран)

Мы уже научились с помощью оператора PRINT выводить на экран числовые константы и переменные, формулы и символьные выражения.

Когда мы используем его для вывода константы, то проблем не возникает, но когда мы в операторе PRINT используем переменные, то должны быть абсолютно уверены, что они заранее определены (либо с помощью операторов LET и INPUT, либо каким-то другим способом). Это значит, что в соответствующих ячейках памяти каждой переменной должны быть записаны какие-то значения.

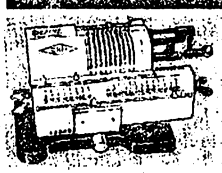
Например, если вы введете:

```
NEW (ВК)
10 PRINT E
RUN (ВК),
```

то компьютер выдаст на экран сообщение, что допущена ошибка (или значение, равное 0). А если:

```
NEW (ВК)
10 LET E=2.71
20 PRINT E;
30 END
RUN (ВК),
```

Технические устройства и механизмы



1878 г.

Русский математик и механик Пафнутий Львович Чебышев создает суммирующий аппарат с непрерывной передачей десятков, а в 1881 г. – приставку к нему для умножения и деления.

В 1880 г. В.Т.Однер создает в России арифмометр с зубчаткой с переменным числом зубцов, а в 1890 г. налаживает их массовый выпуск. Их модификация "Феликс" выпускалась до пятидесятых годов.

В 1904 г. известный русский математик, кораблестроитель, академик А.Н.Крылов предложил конструкцию машины для интегрирования обыкновенных дифференциальных уравнений, которая была построена в 1912 году.

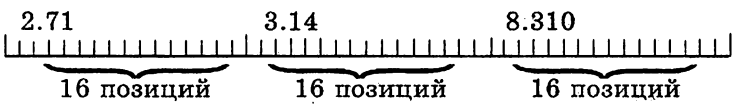
то на экране появится просто число 2.71. Оператор END (от английского – конец) обозначает физический конец программы (оператор STOP представляет собой логический конец программы).

Разрешается в одной строке указывать несколько переменных:

```
60 PRINT E,P,R
```

А точнее:

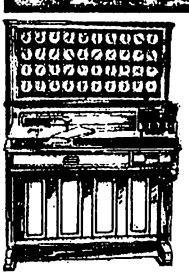
```
NEW (ВК)
30 LET E=2.71
40 LET P=3.14
50 LET R=8.310
60 PRINT E,P,R
RUN (ВК)
```



Если в строке 60 заменить запятые на точку с запятой, то сообщение, выводимое на экран после запуска на выполнение (RUN), будет выглядеть несколько иначе – расстояние между числами уменьшится:

```
RUN
```

Технические устройства и механизмы



1884 г.

Американец Герман Холлерит взял патент "на машину для переписи населения". Изобретение включало *перфокарту* и сортировальную машину. Перфокарта Холлерита оказалась настолько удачной, что без малейших изменений просуществовала до наших дней.

В 1888 г. он создает особое устройство – *табулятор*, в котором информация, нанесенная на перфокарты, расшифровывалась электрическим током.

Лишь в 1907 г. американский инженер Дж.Пауэр сконструировал *автоматический* карточнй *перфоратор*.

2.71 3.14 8.31

В программировании нет мелочей – каждая деталь, даже незначительная, каждый элемент языка Бейсик имеет строго определенные функции, свои правила использования и написания, и этим языки программирования отличаются от естественных языков, в которых есть неоднозначность, отступления, исключения из правил, гибкость.

Так обстоит дело и с разделителями списка переменных – со знаками препинания:

запятой (,);

точкой с запятой (;).

Если использовать разделитель "запятая", то вся выводимая строка делится на 5 зон. Каждая зона, за исключением последней, имеет 16 позиций (в других версиях языка – другое число). Эти 5 зон на экране никак не отмечены, просто надо представить, что экран разделен на 5 зон.

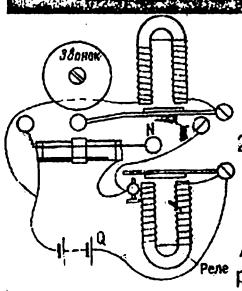
В зоне помещается только одно число, независимо от количества цифр в нем. Поэтому если в списке больше 5 переменных, то 6-я и последующие будут находиться не в той же строке, а в следующей с крайнего левого положения.

Для примера введите строку

```
100 PRINT 1,2,3,4,5,6,7
```

и посмотрите, как ЭВМ будет выводить эти числа на экран. Потом замените запятые на точку с запятой и снова посмотрите результат.

Технические устройства и механизмы



1895 г.

Русский ученый А.С.Попов изобретает радио. 12 марта 1896 г. он демонстрирует радиопередачу на расстояние 250 м. В 1900 г. под его руководством была осуществлена первая практическая радиопередача на расстояние 47 км при спасении броненосца "Генерал-адмирал Апраксин", севшего на камни вблизи острова Гоглана в Финском заливе.

В 1901 г. итальянский физик Маркони установил радиосвязь между Европой и Америкой.

Сделайте то же самое со строками

```
110 PRINT A,B,C,D,E,F,G
120 PRINT "A","B","C","D","E","F","G"
```

Вы, наверное, заметили, что разделитель "точка с запятой" выводит список выражений в так называемом **"плотном формате"**.

Если точка с запятой разделяет в списке PRINT символьные переменные, то они выводятся слитно, без пробела:

```
10 LET K¤="КИНО"
20 LET T¤="ТЕЛЕ"
30 LET P¤="ПРОГРАММА"
40 PRINT K¤;P¤;
50 PRINT T¤;P¤;

RUN (ВК)
КИНОПРОГРАММА
ТЕЛЕПРОГРАММА
```

Если же точка с запятой разделяет в списке числовые переменные, то они выводятся в "плотном формате", но не сливаются в единое число. После каждого числа выводится пробел, а перед положительным числом тоже выводится пробел (в этом месте у отрицательных чисел будет печататься знак "минус"). Следовательно, между любыми двумя числами всегда будет по крайней мере один пробел, а иногда и два. Например:

```
10 LET A=3
20 LET B=7
30 LET X=2
40 LET Y=-9
100 PRINT A*X+B;Y;A

RUN (ВК)
 13  -9  3
```

Каждая команда PRINT, встречающаяся в программе, начинает вывод элементов списка с новой строки, поэтому запись "пустого" оператора PRINT (то есть без текста и списка переменных):

```
200 PRINT
```

вызывает просто пропуск одной строки на экране. А если нам необходимо пропустить несколько строк, то воспользуемся еще одним знаком препинания – двоеточием (:)

```
210 PRINT:PRINT:PRINT
```

Такая запись аналогична:

```
210 PRINT
220 PRINT
230 PRINT
```

Двоеточие используется не только для оператора PRINT, но и для разделения других операторов в одной строке (с целью более рационального использования программного пространства).

И наоборот, для того чтобы не происходил переход на новую строку, необходимо в конце списка оператора PRINT поставить точку с запятой:

```
10 A=15:B=4:C=5
20 PRINT A;
30 Y=A-B*C
40 PRINT Y

RUN (ВК)
 15  -5
```

Чаще всего такой вывод используется для "красивой" печати результатов:

```
10 LET B=4.75
20 PRINT"СРЕДНИЙ БАЛЛ АТТЕСТАТА РАВЕН";B

RUN (ВК)
СРЕДНИЙ БАЛЛ АТТЕСТАТА РАВЕН 4.75
```

Использование запятой как разделителя рекомендуется для печати результатов в виде таблицы, при этом под одну графу таблицы отводится одна зона (16 позиций). Такой вывод списка выражений называют **"зонным форматом"**:

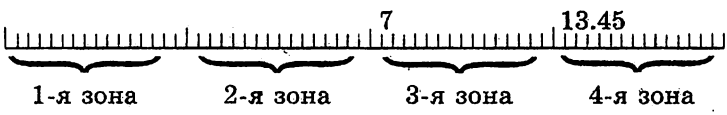
```
10 U1=20: U2=22
20 B1=4.75: B2=4.96
30 K1=0.94: K2=0.98
40 PRINT"10А КЛАСС","10Б КЛАСС"
50 PRINT U1,U2
60 PRINT B1,B2
70 PRINT K1,K2

RUN(ВК)
10А КЛАСС       10Б КЛАСС
 20              22
 4.75            4.96
 .94             .98
```

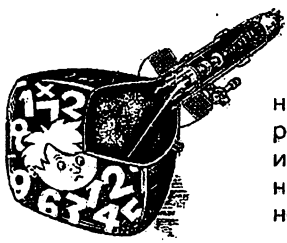
Если в списке стоят рядом две (или более) запятых, то при печати одна (или более) зона будет пропущена. Таким способом при отсутствии данных можно пропустить в таблице несколько граф. Пример:

```
10 CLS
20 X=13.45
30 PRINT,,7,X

RUN (ВК)
```



Технические устройства и механизмы



1897 г.

Английский физик Дж.Томсон (и независимо от него К.Браун) сконструировал электронно-лучевую трубку и с ее помощью исследовал отклонение катодных лучей в электромагнитном поле.

После выполнения оператора CLS экран очистится и курсор установится в начало зоны 1. Первый элемент списка выражения – запятая, поэтому курсор передвинется в начало зоны 2. Следующее данное в списке – тоже запятая. Курсор передвинется в начало зоны 3. На экране печатается число 7. Потом опять запятая – курсор передвинется в начало следующей, четвертой зоны. Далее в списке стоит переменная X. Печатается значение этой переменной – 13.45. Поскольку на этом строка оператора кончается и больше нет никаких разделителей, то оператор PRINT передвигает курсор в начало следующей, то есть второй строки.

Если бы в конце оператора (после X) стояла запятая, то курсор передвинулся бы в начало следующей, пятой зоны и остался бы там.

Еще одним способом вывода таблиц является использование в операторе PRINT функции табулирования. Она имеет вид

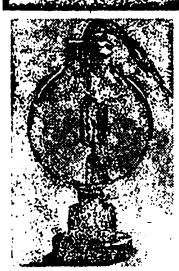
TAB (*позиция*),

где *позиция* – арифметическое выражение.

Зонная печать задает стандартную (16 позиций) ширину граф таблицы. Использование же функции TAB позволяет формировать графы произвольной ширины.

Если в команде PRINT перед элементом списка стоит функция TAB(A), то значение этого элемента будет напечатано с позиции N, где N – целая часть значения арифметического выражения A. Таким образом, функция TAB задает номер позиции при печати. Например:

Технические устройства и механизмы

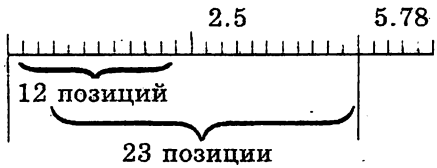


1905 г.

Английский физик Д.А.Флеминг получил патент на электронный прибор – диод для выпрямления колебаний. А в 1906 г. американские физики Л. де Форес и Р.Либен сконструировали вакуумный триод.

1) **10 X=2.5: Y=5.78**
20 PRINT TAB(12);X;TAB(23);Y

RUN (ВК)



2) **10 P1=18.43: P2=4.21**
20 X(1)=7.35: Z¤ ="ЗНАЧЕНИЕ–"
30 PRINT Z¤; TAB((P1+P2)/2);X(1)

RUN (ВК)

А так как $\frac{P1+P2}{2} = \frac{18.43+4.21}{2} = \frac{22.64}{2} = 11.32,$

то TAB((P1+P2/2)=11

RUN

ЗНАЧЕНИЕ- 7.35

11 позиций

Задание определенного количества пробелов между выводимыми значениями производится **функцией пробелов:**

SPC (*количество пробелов*),

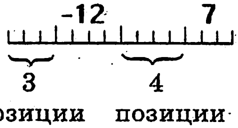
где *количество пробелов* – арифметическое выражение.

Если в команде PRINT перед элементом списка стоит функция SPC(A), то значение этого элемента списка будет отделено от значения предыдущего элемента N пробелами, где N – целая часть значения арифметического выражения A.

Приведем пример использования функции SPC:

10 D=-12: B=7: C=8
20 PRINT SPC(3);D;SPC(C/2);B

RUN (BK)



И последнее. Для удобства вывода текста в произвольных местах экрана служит функция AT (от слова "автомат", т.е. автоматическое смещение) в операторе PRINT. Ее формат:

PRINT AT(X,Y);"ВЫВОДИМЫЙ ТЕКСТ" ,

где X – целое число, которое указывает колонку экрана (меняется от 0 до 63);

Y – целое число, которое указывает строку экрана (меняется от 0 до 23).

Итак, значение X задает абсциссу (или колонку), значение Y – ординату (или строку). Тем самым задается позиция на экране, с которой начинается вывод последующих данных. Колонки нумеруются от 0 до 63 слева направо, строки от 0 до 23 сверху вниз, т.е. начало координат находится в верхнем левом углу экрана (в других версиях языка – в нижнем левом углу) (рис. 18):

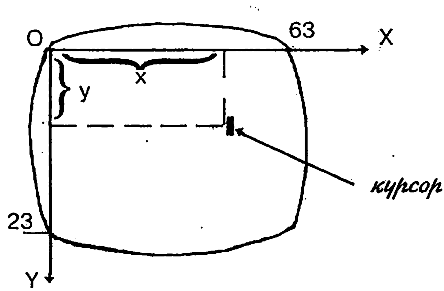


Рис. 18

Если значение аргументов (X или Y) выходит за допустимые пределы, то выдается сообщение об ошибке.

Приведем пример такого вывода сообщения: вывести в центре экрана запись "ВНИМАНИЕ!" (рис. 19):

```
10 CLS
20 PRINT AT(27,11);"ВНИМАНИЕ!"
30 GOTO 20
```

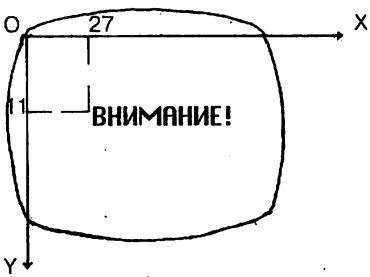
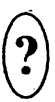


Рис. 19

Проверьте свои знания



1. Каково основное назначение оператора Print?
2. Из чего может состоять список вывода?
3. Чем разделяются элементы списка вывода?
4. Что значит вывести список выражений в "плотном формате"; "зонном формате"?
5. В чем главная особенность оператора Print без списка?
6. В чем состоят различия в выводе пробелов при использовании функций TAB и SPC в операторе Print?
7. Каково назначение функции AT в операторе Print?

Тренировка

I. Не выполняя приведенные ниже фрагменты программы, объясните, как будут выводиться элементы списков:

```
10 CLS
20 PRINT "A";"B";"C";"D";"E"
30 PRINT "A","B","C","D","E"
40 PRINT "A" "B" "C" "D" "E"
50 PRINT "1";"2";"3";"4";"5"
60 PRINT 1;2;3;4;5
70 PRINT 1,2,3,4,5
80 PRINT "A"
```

```
90 PRINT "B"
100 PRINT "C"
110 PRINT "D"
120 PRINT "E"
130 PRINT "A";
140 PRINT "B";
150 PRINT "C";
160 PRINT "D";
170 PRINT "E"
180 PPRINT "A",
190 PRINT "B",
200 PRINT "C",
210 PRINT "D",
220 PRINT "E"
230 PRINT "A": PRINT "B": PRINT "C": PRINT "D": PRINT "E"
240 PRINT "A";: PRINT "B";: PRINT "C";: PRINT "D";: PRINT "E"
250 PRINT "A",: PRINT "B",: PRINT "C",: PRINT "D",: PRINT "E"
```

Затем свои выводы проверьте, запустив фрагменты командой RUN.

II. Объясните, почему при выполнении, казалось бы, одинаковых "по грамматике" программ получают различные результаты вывода?

```
10 LET I¤ ="ИНФОР"          10 LET I=41237
20 LET K¤ ="КИНЕ"           20 LET K=9012
30 LET M¤ ="МАТИКА"         30 LET M=112658
40 PRINT I¤; M¤             40 PRINT I; M
50 PRINT K¤;M¤              50 PRINT K;M

RUN (ВК)                    RUN (ВК)
ИНФОРМАТИКА                  41237  112658
КИНЕМАТИКА                   9012  112658
```

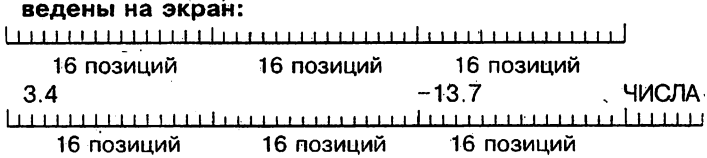
III.

```
10 CLS
20 LET X=7.2
30 LET Y=3
40 PRINT X,,,Y*11,22
RUN (ВК)
```

Что на экране дисплея будет выведено? Объясните полученный результат.

IV. Составьте программу, чтобы она печатала сведения о классе: "КЛАСС", "КОЛ-ВО УЧ-СЯ", "СРЕДНИЙ БАЛЛ", "ПОСЕЩАЕМОСТЬ", используя в качестве разделителя списка данных оператора PRINT лишь запятую. Количество и значение переменных выберите произвольно.

V. Значения трех величин A=3.4, B=—13.7, C ="ЧИСЛА" выведены на экран:



Какой вид имел оператор PRINT в программе?

VI. Каков будет результат работы программ?

1)
```
10 A=3: X=2
20 Y=3^X-4.5*X
30 X=A+5*COS (Y)
40 PRINT "ОТВЕТ:X=";X
50 STOP
```

2)
```
10 A=2:X=-2
20 A=A+X
30 Y=X*TAN(A)+3
40 PRINT "При A=";A;"X=";X;"Y=";Y
50 PRINT
60 PRINT X; "в квадрате равно";
X^2;",";Y;"в кубе равно"; Y^3;"."
```

VII. Составьте программу вычисления и вывода на экран значения функции:

а) $y=a^{x}-bcosx$; б) $z=\sqrt{b}+ax$; при a=2; b=9; x=0.5.

VIII. Сделайте схематический чертеж, поясняющий результат работы функции в операторе PRINT

1)
```
10 K¤ ="10A"
20 F1 ¤="МОИСЕЕНКО": F2 ¤ ="ХМАРА"
30 M1=4: M2=3: M3=3: M4=4: H1=5: H2=4: H3=4: H4=5
40 PRINT TAB(36);"ТАБЕЛЬ"
50 PRINT TAB (20);"УЧЕТА УСПЕВАЕМОСТИ ";
60 PRINT "УЧАЩИХСЯ " ;K¤;" КЛАССА."
70 PRINT: PRINT
80 PRINT "ЧЕТВЕРТЬ", "I","II","III","IV"
90 PRINT
100 PRINT F1¤,M1,M2,M3,M4
110 PRINT F2 ¤,H1,H2,H3,H4
120 STOP
```

2)
```
10 F1¤="ГОЛДА": F2 ¤="КАМЕНСКИЙ"
20 F3 ¤="СЕРЕДА": F4 ¤="ШЕВЧЕНКО": D ¤="ДА"
30 PRINT SPC(32);"ГРАФИК ДЕЖУРСТВА."
40 PRINT: PRINT
50 PRINT SPC(3);"ФАМИЛИЯ";SPC(7);"ПОН.";SPC(6);
"ВТОР.";SPC(6);"СРЕД. ";
60 PRINT SPC(6);"ЧЕТВ.";SPC(6);"ПЯТН.";SPS(6)"СУБ."
70 PRINT
80 PRINT F1 ¤;SPC(13);D ¤;SPC(42);D ¤
```

```
90 PRINT F2 ¤;SPC(19);D ¤;SPC(43);D ¤
100 PRINT F3 ¤;SPC(33);D ¤
110 PRINT F4 ¤;SPC(42);D ¤
120 STOP
```

IX. Используя строку:

10 T ¤ ="ТАВРИЯ": H ¤ ="ШАХТЕР" : D ¤ ="ДИНАМО",

наберите строку 20 с оператором PRINT, чтобы на экране дисплея появилась надпись:

```
RUN
ТАВРИЯ    ШАХТЕР    ДИНАМО
```

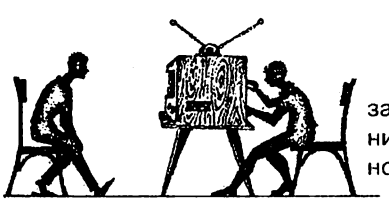
Причем сначала используя функцию TAB, а затем, с тем же результатом, функцию SPC.

X. Напишите программные строки вывода общений "КОМПЬЮТЕР РАД ОБЩАТЬСЯ С ВАМИ!" (в верхней части экрана) и "ДЛЯ ПРОДОЛЖЕНИЯ НАЖМИТЕ ЛЮБУЮ КЛАВИШУ..." (в самой нижней строке экрана).

§ 28. Оператор INPUT (Ввод с клавиатуры)

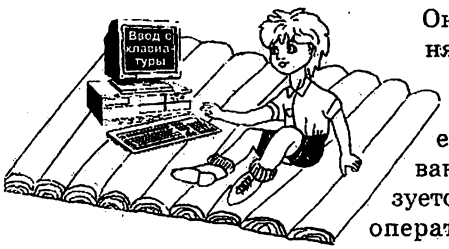
Мы уже знаем, как можно заносить данные в переменную с помощью оператора LET. Но это не самый удобный способ, т.к. в случае, когда нужно изменить значение переменной, приходится заново редактировать строки, в которых стоят операторы LET.

Технические устройства и механизмы



1907 г.

Русский ученый Б.Л.Розинг заявил патент на использование в телевидении электронно-лучевой трубки.



Оказывается, можно изменять значения переменной при каждом новом прогоне программы, избегая ее повторного редактирования. Для этого используется оператор INPUT. Этот оператор действует подобно оператору LET, но только значения переменных вводятся с клавиатуры.

Посмотрим, как работает оператор INPUT с числовыми данными.

Например:

```
400 INPUT X
RUN (ВК)
?
```

На экране появится знак вопроса, и машина остановится: она будет ждать ответа человека, ответа с клавиатуры. Пользователь должен набрать на клавиатуре нужное число, например, 25.3, и нажать клавишу (ВК). Это число будет записано в ячейку оперативной памяти, выделенную под переменную X, т.е. переменной X будет присвоено значение 25.3. После этого ЭВМ приступит к выполнению следующих в программе строк.

Нужно вводить сразу несколько чисел, тогда имена этих переменных в операторе INPUT записывают через запятую.

Например:

Технические устройства и механизмы

В 1929 г. русский инженер А.И.Волков получил патент на электронную систему цветного телевидения.

В середине 30-х годов в результате разработок В.К.Зворыкина и Ф.Франсуорта в США, К.Свинтона в Великобритании, В.П.Грабовского, С.И.Катаева, А.П.Константинова, Б.Л.Розинга в СССР появляются первые системы электронного телевидения.

```
410 INPUT X1,X2,X3,X4
RUN (ВК)
? 17.3,22.0,7779.35 (ВК)
```

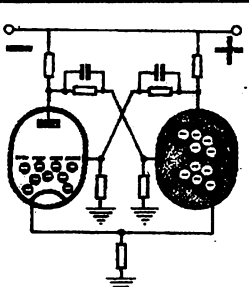
При выполнении такой строки машина на экране выдаст знак вопроса, а после ввода всех четырех чисел присвоит их переменным в том порядке, в каком они записаны в списке. А что будет, если список еще не исчерпан, а клавиша (ВК) уже нажата, скажем, после третьего числа? Тогда машина снова высветит знак вопроса и будет ожидать очередного числа. Что будет, если вы набрали больше чисел, чем переменных из списка? Тогда лишние значения просто отбрасываются, машина их игнорирует.

Надо сказать, что при использовании этого оператора возникает множество ошибок: то между числами пользователь забудет поставить запятую, то в числе – десятичную точку или поставит ее не туда, то пропустит знак (особенно знак "минус" – ведь он нужен обязательно), то ошибется с порядком ввода чисел. Поэтому для контроля вводимых чисел перед строкой с оператором INPUT выдают сообщения-подсказки о том, каких данных ждет от пользователя компьютер. Для этого используют оператор PRINT.

Например:

```
380 PRINT "ВВЕДИТЕ КОЛИЧЕСТВО
ПРИСУТСТВУЮЩИХ В КЛАССЕ"
400 INPUT K
```

Технические устройства и механизмы



1918 г.

Советский ученый М.А.Бонч-Бруевич, а в 1919 г. независимо от него американцы У.Икклз и Ф.Джордон изобретают ламповый триггер.

В 1920 г. американский исследователь Ю.Лименфельд высказал идею создания полупроводниковых усилителей электрических сигналов.

Такое напоминание очень полезно, ведь иногда в память компьютера вводят разнородные величины. Кроме того, с помощью оператора INPUT можно вводить не только числа, но и символьные переменные. Тогда без подсказки просто обойтись невозможно, ведь если тип введенной константы не совпадает с типом соответствующей переменной, то ввод считается ошибочным.

Например:

```
580 PRINT "ВВЕДИТЕ ДАТУ В ВИДЕ ЧЧ,МЕСЯЦ,ГГ"
600 INPUT  Н,М¤,G
```

Это значит, что для каждой переменной надо ввести:
для числа даты – двухразрядное число;
для месяца – символьную переменную;
для года – двухразрядное число.
К примеру, можно дату набрать так:

```
13,ФЕВРАЛЬ,96
```

Во многих версиях Бейсика оператор INPUT как бы совмещен с оператором PRINT:

```
600 INPUT"ВВЕДИТЕ ДАТУ В ВИДЕ ЧЧ,ММ,ГГ";Н,М,G
```

Обращаем ваше внимание, что в такой записи набор месяца производится тоже цифрой. К примеру:

```
13,02,96
```

В списке переменных может присутствовать переменная с индексом. Например:

Технические устройства и механизмы

1930-1935 гг.

Немецкая электротехническая фирма "АЕГ" и химическая "ИГ Фарбениндустри" изготовили магнитную ленту. Она вызвала сенсацию. Для использования этой ленты был создан новый звукозаписывающий прибор, имевший фирменное название *магнитофон*, которое впоследствии стало общим наименованием приборов такого типа.

Однако сама идея не механической, а магнитной записи звука была сделана в 1898 г. датским физиком и инженером Вольдемаром Паульсеном в его приборе "Телеграфон".

```
610 INPUT"ВВЕДИТЕ ЗНАЧЕНИЕ ЧЕТВЕРТОГО ЭЛЕМЕНТА";T(4)
```

Вводимая константа будет занесена в четвертый элемент массива Т.

С помощью оператора INPUT производится обмен информацией между пользователем и компьютером. На его основе строится не только однократный обмен информацией, но и диалог. Со стороны создается впечатление, что пользователь и ЭВМ "разговаривают", "общаются" между собой. Они друг другу задают вопросы и получают на них ответы. Например:

```
10 PRINT "ЗДРАВСТВУЙТЕ, КАК ВАШЕ ИМЯ";
15 INPUT I$
20 PRINT "РАД ВИДЕТЬ ВАС,";I$;"! КОМПЬЮТЕР
ПРИВЕТСТВУЕТ ВАС!"
40 PRINT"ГДЕ ВЫ ЖИВЕТЕ"
50 INPUT P$
60 PRINT "В КАКОМ КЛАССЕ ВЫ УЧИТЕСЬ"
70 INPUT K$
80 PRINT "СКОЛЬКО ВАМ ЛЕТ"
90 INPUT L
100 PRINT"КАКОЕ СОВПАДЕНИЕ, ";I$; "! ВЧЕРА СО
МНОЙ ТОЖЕ ОБЩАЛСЯ ПОЛЬЗОВАТЕЛЬ, ЖИВУЩИЙ В
";P$; ",УЧАЩИЙСЯ ";K$;" КЛАССА,И ЕМУ ТОЖЕ БЫЛО";
L;"ЛЕТ!"
```

Запустив командой RUN эту программу, вы будете отвечать на вопросы компьютера:

```
ЗДРАВСТВУЙТЕ,КАК ВАШЕ ИМЯ? ВОЛОДЯ (ВК)
РАД ВИДЕТЬ ВАС,ВОЛОДЯ! КОМПЬЮТЕР ПРИВЕТСТВУЕТ
ВАС!
ГДЕ ВЫ ЖИВЕТЕ? ДОНЕЦКЕ (ВК)
В КАКОМ КЛАССЕ ВЫ УЧИТЕСЬ?
10А (ВК)
СКОЛЬКО ВАМ ЛЕТ? 16 (ВК)
КАКОЕ СОВПАДЕНИЕ, ВОЛОДЯ! ВЧЕРА СО МНОЙ ТОЖЕ
ОБЩАЛСЯ ПОЛЬЗОВАТЕЛЬ, ЖИВУЩИЙ В ДОНЕЦКЕ,
УЧАЩИЙСЯ 10А КЛАССА, И ЕМУ ТОЖЕ БЫЛО 16 ЛЕТ!
```

Вы заметили, что мы не поставили вопросительные знаки в строках 10, 40, 60, 80, хотя они как бы напра-

шиваются в эти вопросительные предложения. А вот при запуске программы они появляются, и именно там, где необходимы. Понятно, что оператор INPUT выставляет их автоматически. Если же мы поставим этот знак в эти строки, то при прогоне программы в тексте появится два вопросительных знака, что синтаксически будет неправильно.

Приведем еще примеры программ, использующих команду ввода:

А)
```
10 INPUT"КОЛИЧЕСТВО УЧЕНИКОВ ПО СПИСКУ";S
20 INPUT"КОЛИЧЕСТВО ПРИСУТСТВУЮЩИХ";P
30 PRINT"ПРОЦЕНТ ПОСЕЩАЕМОСТИ";P/S*100;"%"
40 STOP

RUN (ВК)
КОЛИЧЕСТВО УЧЕНИКОВ ПО СПИСКУ? 32 (ВК)
КОЛИЧЕСТВО ПРИСУТСТВУЮЩИХ? 30 (ВК)
ПРОЦЕНТ ПОСЕЩАЕМОСТИ 93.75%
```

Б)
```
20 INPUT"ВВЕДИТЕ РАДИУС ОСНОВАНИЯ";R
30 INPUT"ВВЕДИТЕ ВЫСОТУ КОНУСА";H
40 LET V=(1/3)*PI*R^2*H
50 PRINT "V=";V
60 STOP
RUN (ВК)
ВВЕДИТЕ РАДИУС ОСНОВАНИЯ? 10 (ВК)
ВВЕДИТЕ ВЫСОТУ КОНУСА? 15 (ВК)
V=1570.796
```

Проверьте свои знания

1. В чем состоит сходство ввода данных в переменные с помощью операторов LET и INPUT?

2. Какое соответствие должно быть между вводимыми константами и переменными из списка?

3. Чем разделяются элементы списка в операторе ввода?

4. Как "поступит" компьютер при выполнении оператора INPUT, если:

а) пользователь набрал констант больше, чем стоит переменных в списке оператора?

б) не набирая констант, сразу нажали клавишу (ВК)?

Тренировка

I. Найдите ошибки в программе:

```
10 LET P=3,1415926
20 INPUT "Введите диаметр;D
30 LET S=P · D^2/4
40 PRINT "S=",S
50 INPUT "Еще"
60 END
```

II. Напишите программу, которая запрашивает ваше имя, а потом печатает слово "ПРИВЕТ", ваше имя и какое-либо сообщение для вас.

III. Перед вами рифмованные шутки. Они придуманы для малышей, чтобы легче запомнились слова с удвоенными согласными:

а) Если в доме много сора,
В доме может вспыхнуть ссора;

б) Не придет на школьный бал,
Кто получит низкий балл.

Используя их, предлагаем вам составить программу так, чтобы одно из рифмующихся слов вводилось с клавиатуры.

IV. Вам представлена программа, которая пишет стихи:

```
5 CLS
10 INPUT "ПОДБЕРИТЕ РИФМУ К СЛОВУ 'Я'";S$
20 PRINT "СЛОЖИЛИ СТИХ МЫ, - ЭВМ И Я"
30 PRINT "КОМПЬЮТЕР БОЛЬШЕ, - МЕНЬШЕ Я"
40 PRINT "ТЕПЕРЬ Я СЧАСТЛИВ, КАК ";S$
50 END
```

Используя оператор INPUT, подставьте в нее строки так, чтобы она выводила авторство этого стихотворения в юмористической форме. Например: Сергеев-Ценский, Иванов-Питерский, Джордж Гордон Белкин и т.д.

§ 29. Переходы в программе

Обычно программа, написанная на Бейсике, начинает исполняться с первого оператора первой строки. Далее операторы выполняются в естественном порядке:

Последовательно слева направо в строке и сверху вниз по строкам.

Нарушить последовательность выполнения операторов можно с помощью операторов GOTO, IF...THEN, ON, GOSUB.

Оператор GOTO (перейти к...)

Предположим, что вы хотите написать программу с оператором INPUT и серией вопросов, ответы на которые будут давать разные пользователи. Когда один из них ответит на все вопросы, каким образом вы могли бы опять вернуться к началу программы?

Один из способов вы знаете: это повторно дать команду RUN. А если таких пользователей будет много, то каждый раз для повторения выполнения этой программы снова и снова набирают команду RUN. Однако гораздо удобнее научить компьютер возвращаться к начальной строке и вновь задать вопросы при помощи оператора GOTO.

Давайте возьмем в качестве примера диалоговую программу с предыдущего параграфа и добавим в нее строку

```
110 GOTO 10
```

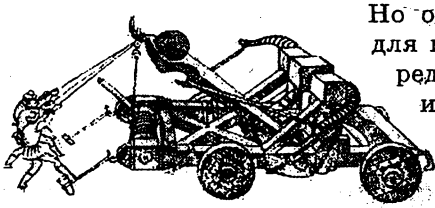
Компьютер при первом "прогоне" программы дойдет до строки 110 и затем вернется к строке 10 и снова начнет выполнять эту программу. Выполнение операторов далее будет происходить последовательно и аналогично первому прогону:

```
RUN (ВК)
ЗДРАВСТВУЙТЕ, КАК ВАШЕ ИМЯ? ВОЛОДЯ (ВК)
РАД ВИДЕТЬ ВАС, ВОЛОДЯ! КОМПЬЮТЕР
ПРИВЕТСТВУЕТ ВАС!
ГДЕ ВЫ ЖИВЕТЕ? ДОНЕЦКЕ
В КАКОМ КЛАССЕ ВЫ УЧИТЕСЬ? 10А (ВК)
СКОЛЬКО ВАМ ЛЕТ? 16 (ВК)
КАКОЕ СОВПАДЕНИЕ, ВОЛОДЯ! ВЧЕРА СО МНОЙ ТОЖЕ
ОБЩАЛСЯ ПОЛЬЗОВАТЕЛЬ, ЖИВУЩИЙ В ДОНЕЦКЕ,
УЧАЩИЙСЯ 10А КЛАССА, И ЕМУ ТОЖЕ БЫЛО 16 ЛЕТ!
```

ЗДРАВСТВУЙТЕ, КАК ВАШЕ ИМЯ? ОКСАНА (ВК)
РАД ВИДЕТЬ ВАС ОКСАНА! КОМПЬЮТЕР ПРИВЕТСТВУЕТ ВАС!
ГДЕ ВЫ ЖИВЕТЕ! КИЕВЕ (ВК)
В КАКОМ КЛАССЕ ВЫ УЧИТЕСЬ? 11 (ВК)
СКОЛЬКО ВАМ ЛЕТ? 17 (ВК)
КАКОЕ СОВПАДЕНИЕ, ОКСАНА! ВЧЕРА СО МНОЙ ТОЖЕ ОБЩАЛСЯ ПОЛЬЗОВАТЕЛЬ, ЖИВУЩИЙ В КИЕВЕ, УЧАЩИЙСЯ 11 КЛАССА, И ЕМУ ТОЖЕ БЫЛО 17 ЛЕТ!

Прерывание в строке 20

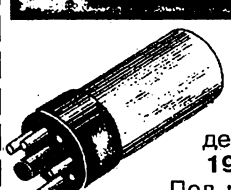
Если нажать кнопку прерывания, то на экране появится сообщение "Прерывание в строке 20" или что-то подобное. Это говорит нам о том, в каком месте программы мы прервали ее выполнение, нажав на эту кнопку. Если мы не будем дотрагиваться до этой кнопки, компьютер будет работать снова и снова, производя прогон в третий, четвертый и т.д. раз. Такой цикл, который никогда не кончается, называется **бесконечным**, а о программе в подобном случае говорят, что она **зацикливается**.



Работа оператора GOTO напоминает работу камнебросалки

Но оператор GOTO служит для перехода не только вперед по ходу программы, но и назад. Вообще, оператор GOTO называют оператором **безусловного перехода**, так как он может перейти на выполнение любой стро-

Технические устройства и механизмы



1938 г.
Американец Р.Риш демонстрирует механическое говорящее устройство. А в 1939 г. он совместно с Додли и Уоткинсом демонстрирует *синтезатор речи*.

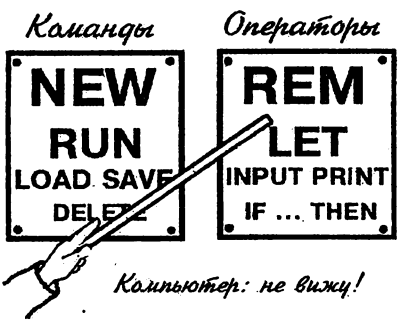
1940 г.
Под руководством Джона фон Неймана разработан компьютер MANIAC, а затем, в 1945-м, – EDVAC. В 1943 г. под руководством Бистчли создается первый электронный компьютер COLOSSUS-1.

ки, номер которой указан в этом операторе. Например, GOTO 20 означает "Перейти на строку с номером 20", а GOTO 1560 – "Перейти на строку с номером 1560". Разумеется, в программе должна быть и строка 20, и строка 1560, чтобы было куда перейти. В противном случае при запуске программы на экране появится сообщение об ошибке.

Иногда в операторе GOTO указывают номер строки, в которой нет ничего, кроме комментария. В качестве оператора комментария используется оператор

REM
(от английского REMARK – примечание, пояснение)

Компьютер игнорирует все, что набрано в строке после ключевого слова REM. Всякий раз, когда он наталкивается на этот оператор, он пропускает его и переходит к следующей программной строке. Этот оператор не влияет на процесс выполнения программы, а используется лишь для того, чтобы написать заголовок или объяснения к программе либо к ее отдельным частям. Вместо командного слова REM допустимо ставить просто кавычку (апостроф – '): интерпретатор воспринимает ее как сокращение для REM. Эти операторы появляются на экране каждый раз



Технические устройства и механизмы

1942 г.

Американские инженер Д.П.Эккерт и физик Д.У.Моучли сконструировали в Пенсильванском университете первую ЭВМ "ЭНИАК".

при выводе текста программы (с помощью команды LIST), но на ее прогон они не влияют.

Но вернемся к оператору GOTO. Итак, оператор REM не исполняется, и поэтому в операторе GOTO лучше передавать управление следующему за ним исполняемому оператору. В этом случае программа будет выполняться правильно и после удаления строк с комментариями. А это иногда приходится делать, чтобы уменьшить объем занимаемой программной памяти. Например:

```
5 ' ПРОГРАММА ПО ОТГАДЫВАНИЮ ЧИСЕЛ
10 GOTO 50
20 REM ВЫПОЛНЕНИЕ УСЛОВИЙ
30 PRINT "РАЗДЕЛИТЕ НА 4 И ОТНИМИТЕ 11"
40 GOTO 110
50 PRINT "ЗАДУМАЙТЕ ЧИСЛО,"
60 GOTO 90
70 PRINT "УМНОЖЬТЕ НА 8, ВЫЧТИТЕ 12"
80 GOTO 20
90 PRINT "УДВОЙТЕ ЕГО, ПРИБАВЬТЕ 4"
95 PRINT "РАЗДЕЛИТЕ НА 2, ПРИБАВЬТЕ 7"
100 GOTO 70
110 REM ЗАПРОС ПОСЧИТАННОГО РЕЗУЛЬТАТА
120 INPUT "СКОЛЬКО У ВАС ПОЛУЧИЛОСЬ";K
130 PRINT
140' ВЫВОД РЕЗУЛЬТАТОВ
150 PRINT "ВЫ ЗАДУМАЛИ ЧИСЛО:";(K-4)/2
```

После запуска программы на экране появится:

```
RUN (ВК)
ЗАДУМАЙТЕ ЧИСЛО,
УДВОЙТЕ ЕГО, ПРИБАВЬТЕ 4
РАЗДЕЛИТЕ НА 2, ПРИБАВЬТЕ 7
УМНОЖЬТЕ НА 8, ВЫЧТИТЕ 12
РАЗДЕЛИТЕ НА 4 И ОТНИМИТЕ 11
СКОЛЬКО У ВАС ПОЛУЧИЛОСЬ? 8 (ВК)
ВЫ ЗАДУМАЛИ ЧИСЛО: 2
```

Теперь мы хотим удалить программные строки, имеющие операторы REM. В приведенном примере сообщение об ошибке не появится, если оператор GOTO в строке 40 заменить на оператор GOTO 120 и, соответственно, в строке 80 – на оператор GOTO 30.

Номер строки задается в GOTO в виде константы. Если в константе встречается десятичная точка, то интерпретатор Бейсика игнорирует ее и все следующие за ней десятичные числа. Недопустимо в качестве номера строки использовать число в экспоненциальном представлении, переменную или выражение.

Заметим, что, по возможности, следует избегать применения оператора GOTO, т.к. частое его использование приводит к увеличению вероятности задать неправильный номер строки, что фактически приводит к неправильному выполнению всей программы. Да и сам текст (листинг) программы становится очень запутанным или, как говорят, "трудночитаемым".

Проверьте свои знания

1. Каково назначение оператора GOTO?
2. Почему оператор GOTO называют оператором безусловного перехода?
3. Как с помощью оператора GOTO организуется неоднократная работа одной и той же программы?
4. Может ли в качестве номера строки в операторе GOTO использоваться переменная или выражение?
5. Каково назначение оператора REM?
6. Почему не рекомендуют передавать управление в операторе GOTO на строку с операторами комментария?

Тренировка

I. Напишите две программные строки, образующие бесконечный цикл, таким образом, чтобы при каждом повторении цикла компьютер выводил на экран слова: "И ДЕНЬ ИДЕТ, И НОЧЬ ИДЕТ" (Т.Г.Шевченко).

II. Используя оператор GOTO и строки из стихотворения Анны Ахматовой:

1. И вовсе я не пророчица,
2. Жизнь моя светла, как ручей,
3. А просто мне петь не хочется
4. Под звон тюремных ключей

напишите программу так, чтобы в ней сначала стояли нечетные строки, а затем четные, но чтобы при выполнении ее на экране появлялся текст оригинала.

III. Найдите значение переменной Y после выполнения программ (устно):

а)
```
10 X=3
20 Y=2
30 GOTO 60
40 Y=Y*X-12
50 GOTO 80
60 Y=X+Y^2
70 GOTO 40
80 Y=SQR(Y)
90 PRINT Y
100 END
```

б)
```
10 X=4.5: Y=X
20 GOTO 50
30 Y=SQR (Y)
40 GOTO 70
50 Y=Y+X
60 GOTO 30
70 Y=2*X
80 PRINT Y
90 END
```

IV. Проследите за выполнением программы:

```
10 A$="АВТО": B$="БИО"
20 G$="ГРАФ": I$="И": Q$="Я"
30 PRINT Q$: PRINT I$
40 PRINT G$ : PRINT B$: PRINT A$
50 GOTO 100
60 PRINT A$+B$+G$+I$+Q$
70 END
80 PRINT B$+G$+I$+Q$
90 GOTO 60
100 PRINT G$+I$+Q$
110 GOTO 80
```

Что будет выведено на экране?

§ 30. Оператор IF...THEN (Если... тогда)

Оператор IF...THEN имеет одно из самых полезных свойств компьютера – сравнивает данные и выполняет различные действия в зависимости от результата.

На месте многоточия в операторе IF...THEN (то есть сразу после слова IF – если) стоит "условие". Под условием понимают любое логическое выражение. В зависимости от этого условия (его

истинности или ложности) компьютер выполняет ту или иную ветвь программы.

Вот так выглядит оператор IF...THEN в программе:

```
10 INPUT "СКОЛЬКО БУДЕТ 2*2";X
20 IF X=4 THEN "ПРАВИЛЬНО!"
30 IF X<>4 THEN PRINT "НЕПРАВИЛЬНО!"
40 END
```

Строка 20 содержит оператор IF...THEN, который состоит из двух ключевых слов IF и THEN (напомним, что изученные нами операторы PRINT,INPUT и т.д. состоят из одного ключевого слова). Выполняя эту строку, компьютер должен проанализировать значение, записанное в переменной X. Если оно равно 4, то он продолжит выполнение оставшейся части строки (THEN – тогда) и выведет на экран слово "ПРАВИЛЬНО!".

В противном случае, если значение переменной X не равно 4, компьютер выполнит строку 30 и на экране выведет слово "НЕПРАВИЛЬНО!".

```
RUN (ВК)                    RUN (ВК)
СКОЛЬКО БУДЕТ 2*2? 4        СКОЛЬКО БУДЕТ 2*2? 5
ПРАВИЛЬНО!                  НЕПРАВИЛЬНО!
```

Ни в коем случае не следует путать "=" из логического условия с "=", используемый в операторе присваивания LET, поскольку первый означает "равно", а второй "присвоить". "Равно" не вызывает изменения ни одной переменной (и поэтому все равно, как писать: X=4 или 4=X); "присвоить" же, вообще говоря, всегда вызывает изменение одной переменной. Той, имя которой находится слева от "=", и поэтому "присвоить" – несимметрично.

Кроме "=" и "<>" условие может содержать и другие операции отношения:

< – меньше;

> – больше;

<= – меньше или равно;

>= – больше или равно.

Оператор IF...THEN может работать не только с числовыми переменными, но и со строковыми. При включении в оператор IF...THEN строковых переменных используются только два знака сравнения:

= – равно;

<> – не равно.

Компьютер в этом случае просто определяет, совпадает ли одна строка символов с другой.

Подобный оператор будет выглядеть так:

```
IF V$ ="ВИНЕР" THEN PRINT "ПРАВИЛЬНО"
```

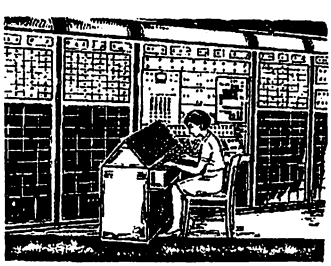
А теперь посмотрите, как такой оператор может использоваться в программе:

```
10 PRINT "НАЗОВИТЕ УЧЕНОГО, СФОРМУЛИРОВАВШЕГО
ОСНОВНЫЕ ПОЛОЖЕНИЯ КИБЕРНЕТИКИ"
20 INPUT V$
30 IF V$="ВИНЕР" THEN PRINT "ПРАВИЛЬНО"
40 IF V$<>"ВИНЕР" THEN PRINT "НЕПРАВИЛЬНО"
50 END

RUN (ВК)
НАЗОВИТЕ УЧЕНОГО, СФОРМУЛИРОВАВШЕГО
ОСНОВНЫЕ ПОЛОЖЕНИЯ КИБЕРНЕТИКИ
? ВИНЕР
ПРАВИЛЬНО
```

В операторе IF...THEN строчка, содержащая в переменной V$, сравнивается, символ за символом, со строчкой, записанной непосредственно в программной строке. Компьютер будет считать две строчки равными только тог-

Технические устройства и механизмы



1947 г.

Под руководством академика Лебедева С.А. и Глушкова В.М. разрабатываются отечественные ЭВМ: сначала МЭСМ – малая электронная счетная машина (1951 г., Киев), затем БЭСМ – быстродействующая электронная счетная машина (1952 г., Москва). Параллельно с ними создавались "Стрела", "Урал", "Минск", "Раздан", "Наири".

да, когда совпадут все соответствующие друг другу символы. Так, если ввести слово "Винер", или "Н.Винер", или " Винер ", то ни одна из них не будет считаться равной строке "ВИНЕР".

Особый интерес представляет сочетание IF с двумя операторами – с GOTO и самим IF.

Конструкции типа IF...THEN GOTO... (и IF...THEN GOSUB) столь распространены в программах, написанных на Бейсике, что пришлось ввести сокращение: вместо GOTO можно писать просто THEN или просто GOTO. Например:

```
10 LET P¤="БАНАН"
20 INPUT "ВВЕДИТЕ ПАРОЛЬ";S¤
30 IF S¤=P¤ THEN GOTO 60
40 PRINT "НЕПРАВИЛЬНО! ПОПРОБУЙТЕ ЕЩЕ"
50 GOTO 20
60 PRINT "ВСЕ В ПОРЯДКЕ! КОМПЬЮТЕР ПУСКАЕТ
ВАС В ПРОГРАММУ"
70 REM ОСТАЛЬНАЯ ПРОГРАММА
180 END
```

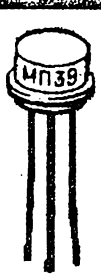
Строку 30 можно записать и по-другому:

или
```
30 IF S¤=P¤ THEN 60
30 IF S¤=P¤ GOTO 60
```

Результат выполнения не изменится.

Сочетание THEN IF позволяет "вкладывать" условия друг в друга. Например:

Технические устройства и механизмы



1948 г.

Американские физики Уолтер Браттейн, Джон Бардин и Уильям Шокли сконструировали *транзистор*.

В 1954-1957 гг. фирмой NCR (США) создается *первый компьютер на транзисторах*.

1957 г.

В СССР осуществлен первый запуск космического корабля. Соответственно с ним велась первая космическая радиосвязь.

```
500 IF X>2.3 THEN IF X<=9.7 THEN 440
```

посылает на строку 440, если к моменту выполнения X находится в интервале (2,3 ; 9,7].

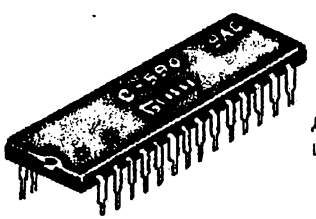
Действие в операторе IF...THEN можно задавать последовательностью нескольких операторов, разделенных двоеточием:

```
10 GOTO 30
20 PRINT"УЧАЩИЙСЯ:";N;".";F¤;" - ИМЕЕТ СРЕДНИЙ
БАЛЛ ПО ПРЕДМЕТУ - ";B;", КРУЖОК - ";P¤
30 INPUT"ВВЕДИТЕ ПОРЯДКОВЫЙ НОМЕР
УЧАЩЕГОСЯ";N
40 IF N=1 THEN F¤="АНДРЕЙЧЕНКО": B=5: P¤=
"ПОСЕЩАЕТ.": GOTO 20
50 IF N=2 THEN F¤="БОНДАРЕНКО": B=4: P¤="НЕ
ПОСЕЩАЕТ.": GOTO 20
60 IF N=3 THEN F¤="СЕРЕДА": B=3: P¤="НЕ ЗАПИ-
САН.": GOTO 20
70 PRINT"ДАННЫЕ ОБ УЧАЩЕМСЯ ПОД НОМЕРОМ";
N;"ОТСУТСТВУЮТ": GOTO 30

RUN (ВК)
ВВЕДИТЕ ПОРЯДКОВЫЙ НОМЕР УЧАЩЕГОСЯ? 2 (ВК)
УЧАЩИЙСЯ: 2. БОНДАРЕНКО - ИМЕЕТ СРЕДНИЙ БАЛЛ
ПО ПРЕДМЕТУ - 4, КРУЖОК - НЕ ПОСЕЩАЕТ.
ВВЕДИТЕ ПОРЯДКОВЫЙ НОМЕР УЧАЩЕГОСЯ?
```

В развитых версиях Бейсика оператор IF...THEN обладает расширенными возможностями и представляется в следующем виде:

Технические устройства и механизмы



1967 г.
В США создана первая быстродействующая ЭВМ на БИСах (больших интегральных схемах).

IF (УСЛОВИЕ) THEN (СЕРИЯ 1) ELSE (СЕРИЯ 2)

Здесь ELSE (ИНАЧЕ) – ключевое слово языка;

(СЕРИЯ 1), (СЕРИЯ 2) – последовательности операторов, разделенных двоеточием, или номера строк;

(УСЛОВИЕ) – любое логическое выражение.

При выполнении этого оператора сначала вычисляется значение логического выражения. Если оно истинно, т.е. условие справедливо, то выполняются операторы СЕРИИ 1.

Если логическое выражение ложно, т.е. условие нарушено, то выполняются операторы СЕРИИ 2.

Поясним работу этого оператора на блок-схеме (она называется базовой структурой "РАЗВИЛКА").

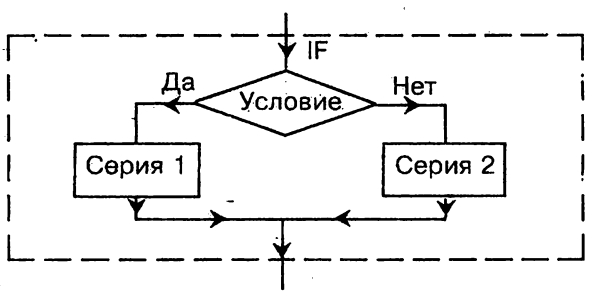
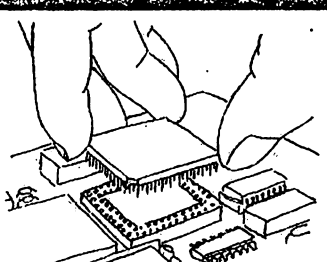


Рис.20. Базовая структура РАЗВИЛКА (полная)

Для сравнения приведем пример изученной нами выше неполной условной конструкции:

Технические устройства и механизмы



1971 г.

Фирмой "Интел" (США) создан первый микропроцессор (МП) – программируемое логическое устройство, изготовленное по технологии СБИС.

IF (УСЛОВИЕ) THEN (СЕРИЯ)

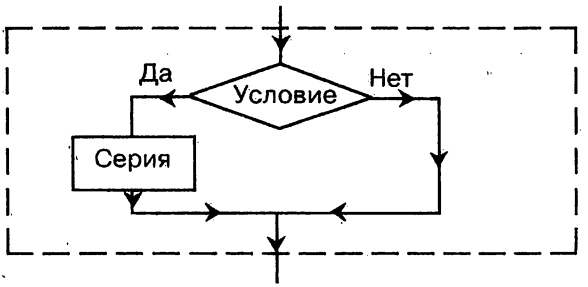


Рис. 21 Базовая структура РАЗВИЛКА (неполная)

Например:

Составьте программу для вычисления значения функции:

$$Y=\begin{cases}\frac{\sin x}{x}, & \text{если } x \neq 0\\ 1, & \text{если } x=0\end{cases}$$

Программа будет выглядеть очень просто:

```
10 INPUT"ВВЕДИТЕ АРГУМЕНТ X";X
20 IF X<>0 THEN Y=SIN(X)/X ELSE Y=1
30 PRINT"ПРИ ЗНАЧЕНИИ X=";X;"ФУНКЦИЯ БУДЕТ
РАВНА ";Y
40 END
```

При выполнении строки 20 компьютер буквально будет делать следующее: ЕСЛИ X не равно нулю, ТОГДА функции Y присваивается значение SIN(X)/X (и заодно считается это значение с учетом введенного аргумента X), ИНАЧЕ функции Y присваивается значение, равное единице.

Операторы IF...THEN..ELSE тоже могут находиться в программе один внутри другого. Их в этом случае называют **вложенными.**

Например:

Составьте программу вычисления значения функции

$$y=\begin{cases}-x, & \text{если } x<1,\\ -1, & \text{если } 1\le x\le 3\\ x-4, & \text{если } x>3\end{cases}$$

Программу используем ту же, заменив строку 20 на такую:

```
20 IF X<1 THEN Y=-X ELSE IF X<=3 THEN Y=-1 ELSE Y=X-4
```

Заметим, что перед командным словом ELSE двоеточие не ставится.

Количество вложенных друг в друга операторов IF (глубина вложения) при расположении их в пределах одной логической строки ограничено максимальным размером строки (255 символов). Для увеличения глубины вложения можно использовать раздельную запись соответствующих серий операторов. Например, блок-схема и программа для расчета этой функции будет представлена в виде:

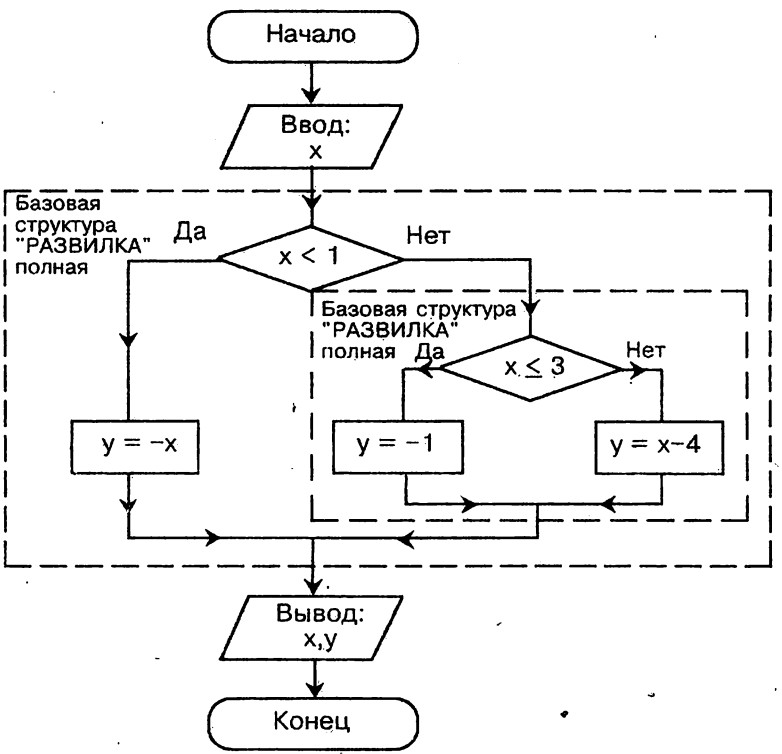


Рис. 22

```
10 INPUT "ВВЕДИТЕ АРГУМЕНТ X";X
20 IF X<1 THEN 30 ELSE 40
30 Y=-X: GOTO 70
40 IF X<=3 THEN 50 ELSE 60
50 Y=-1: GOTO 70
60 Y=X-4
70 PRINT "ПРИ ЗНАЧЕНИИ X=";X;"ФУНКЦИЯ БУДЕТ
РАВНА ";Y
80 END
```

Еще пример:

Разделить натуральное число A на натуральное число B (A>B). Результат представить в виде частного от деления K и остатка L (операцию деления не использовать).

Решение

Операцию деления можно представить как последовательность вычитаний делителя из делимого. Тогда количество вычитаний будет частным от деления K. При этом последовательные вычитания нужно проводить до тех пор, пока результат вычитания не станет меньше делителя. Тогда эта последняя разность и будет остатком от деления L.

Блок-схема и программа этого решения имеют вид:

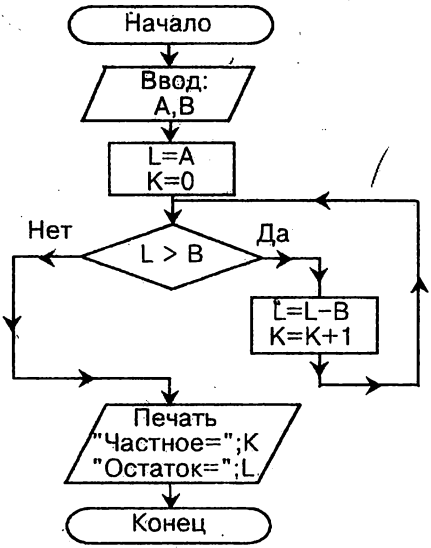


Рис. 23

```
10 INPUT "ВВЕДИТЕ НАТУРАЛЬНЫЕ ЧИСЛА A, B";A,B
20 L=A: K=0
30 IF L>B THEN 50
40 GOTO 70
50 L=L-B: K=K+1
60 GOTO 30
70 PRINT "ЧАСТНОЕ=";K;"ОСТАТОК=";L
80 END
```

Такую блок-схему оператора IF...THEN можно отнести к базовой структуре ЦИКЛ-ПОКА, которую условно рисуют так:

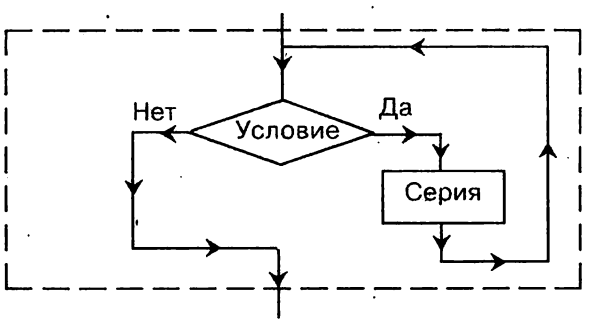


Рис. 24. Базовая структура ЦИКЛ-ПОКА

А теперь, когда накоплено определенное количество знаний, попытаемся составить более сложные программы.

Для примера возьмем составленные нами в §16 блок-схемы решения квадратного уравнения и нахождения наименьшего общего делителя двух натуральных чисел.

Переложим эти алгоритмы с языка блок-схем в программы на языке Бейсик.

Пример 1

```
10 REM РЕШЕНИЕ КВАДРАТНОГО УРАВНЕНИЯ ВИДА
AX^2+BX+C=0
20 INPUT"ВВЕДИТЕ КОЭФФИЦИЕНТЫ A,B,C";A,B,C
30 LET D=B^2-4*A*C
40 IF D<0 THEN 80
```

```
50 LET X1=-B-D: LET X2=-B+D
60 PRINT"КОРНЕЙ ДВА: X1=";X1;" X2=";X2
70 GOTO 90
80 PRINT "ДЕЙСТВИТЕЛЬНЫХ КОРНЕЙ НЕТ"
90 END
```

Пример 2

```
10 REM НАХОЖДЕНИЕ НАИМЕНЬШЕГО ОБЩЕГО
ДЕЛИТЕЛЯ ДВУХ НАТУРАЛЬНЫХ ЧИСЕЛ
20 INPUT"ВВЕДИТЕ ДВА НАТУРАЛЬНЫХ ЧИСЛА";M,N
30 X=M: Y=N
40 IF X=Y THEN 90
50 IF X>Y THEN 70
60 Y=Y-X
65 GOTO 40
70 X=X-Y
80 GOTO 40
90 D=X
100 PRINT"НАИБОЛЬШИЙ ОБЩИЙ ДЕЛИТЕЛЬ";D
110 END
```

Проверьте свои знания

1. Какое основное свойство использует компьютер при выполнении оператора IF ...THEN?

2. Можно ли при записи условия в операторе IF...THEN менять местами левую и правую части равенства? Почему?

3. Как происходит сравнение символьных значений условия в операторе IF...THEN?

4. Поясните сходство и различие в работе неполной и полной условных конструкций оператора IF...THEN.

5. Что значит "вложенные друг в друга сочетания THEN IF"? Приведите примеры. Какова глубина их вложения?

6. Поясните работу базовой структуры "РАЗВИЛКА".

Тренировка

I. Поясните, в чем схожесть двух приведенных ниже программ:

```
10 INPUT "СКОЛЬКО БУДЕТ 3+5";K
20 IF K=8 THEN PRINT "ПРАВИЛЬНО!"
30 IF K<>8 THEN PRINT "НЕПРАВИЛЬНО!"
40 END
```

```
10 INPUT "СКОЛЬКО БУДЕТ 3+5";K
20 IF 8=K THEN PRINT "ПРАВИЛЬНО !"
30 IF 8<>K THEN PRINT "НЕПРАВИЛЬНО !"
40 END
```

II. Составьте программу определения кислотности раствора с помощью лакмусовой бумажки. Напомним, что если бумажка окрашивается:

в красный цвет – раствор кислотный;
синий цвет – раствор щелочной;
не меняет цвета – раствор нейтральный.

III. Даны два числа A и B. Напишите программу определения максимального из этих чисел. Сначала воспользуйтесь полной условной конструкцией, а затем неполной.

IV. Напишите одной программной строкой, используя вложенные друг в друга условия :

ЕСЛИ $0 \le X \le 10$, ТО ТОЧКА ПРИНАДЛЕЖИТ ЭТОМУ ИНТЕРВАЛУ.

V. Заданы два целых числа X и Y. Напишите программу проверки, делится ли одно из этих чисел на другое нацело?

VI. Составьте программу игры в угадывание чисел, выполнив следующие условия:

1. Задуманное число должно быть целым и не больше 100.
2. После ввода задуманного числа экран должен очищаться от любой информации.
3. Заставьте компьютер сообщать вам, когда вводимое вами число слишком мало или слишком велико;
4. Добавьте переменную для подсчета числа сделанных вами попыток, затем с помощью оператора IF...THEN задайте допустимое число попыток (например, пять).
5. Предоставьте игрокам возможность выбора меню: продолжить эту игру или закончить ее.

VII. Составьте программы к заданиям из "Тренировки" к §16.

§ 31. Оператор цикла FOR...NEXT (Для... Потом)

С понятием бесконечного цикла мы с вами уже встречались. Например:

```
20 PRINT"ПРИВЕТСТВУЕМ ВАС!"
50 GOTO 20
```

Программно выйти из такого цикла можно, используя оператор IF...THEN (ссылаясь на "заведенный" счетчик цикла):

```
10 S=0
20 PRINT"ПРИВЕТСТВУЕМ ВАС!"
30 S=S+1
40 IF S=5 THEN 60
50 GOTO 20
60 END
```

Но есть другой, более мощный и эффективный способ организации цикла. Он основан на использовании оператора повторения FOR...NEXT. Его по-другому называют **контролируемый цикл**. Когда вы его используете, компьютер выполняет задание только определенное количество раз. А поскольку вы сами определяете это число раз, то вам незачем беспокоиться о поведении этого цикла. Компьютер сам остановится, выполнив указанное число циклов.

Оператор FOR...NEXT является парным (один из таких двойных операторов мы знаем: IF...THEN). Первый оператор FOR ("для", "от") начинает цикл, второй NEXT ("потом") – завершает его, закрывает. Они используются всегда вместе, отсутствие одного вызовет сообщения об ошибке.

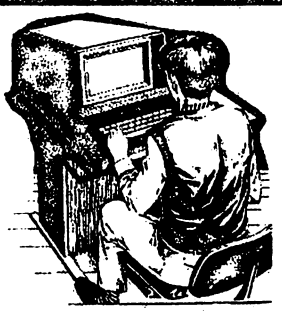
Вышеописанную программу с его помощью можно написать так:

```
10 FOR I=1 TO 5
20 PRINT"ПРИВЕТСТВУЕМ ВАС!"
30 NEXT I
40 END

RUN (ВК)
ПРИВЕТСТВУЕМ ВАС!
ПРИВЕТСТВУЕМ ВАС!
ПРИВЕТСТВУЕМ ВАС!
ПРИВЕТСТВУЕМ ВАС!
ПРИВЕТСТВУЕМ ВАС!
```

Первую строку можно перевести с алгоритмического языка на обычный так: выполнить все нижеследующее для значений переменной I от 1 до 5 ("ТО" означает "ДО"). Таким образом, в строке 10 содержится указание повторить то, что находится внутри цикла, внутри обрамляющих его рамок FOR-NEXT (**Тело цикла**), ровно 5 раз. В строке 30 проверяется, завершен ли цикл, т.е. достигло ли значения I пяти, повторен ли цикл 5 раз, и если нет, то программа снова переходит на строку 10. Эта проверка осуществляется по переменной I, называемой **управляющей переменной** или **параметром цикла**. При первом выполнении ей присваивается начальное значение, указанное в этой строке, т.е. I=1 (FOR I=1 TO 5 читается "от I=1 до 5"). Далее выполняется строка 20. На экране печатается первый раз сообщение "Приветствуем вас!", строка 30 NEXT I – это возвратная часть цикла FOR...NEXT (не зря

Технические устройства и механизмы



1975 г.
Появилась первая персональная электронно-вычислительная машина (ПЭВМ).

читается "потом вернись на начало цикла"). Когда компьютер встречает эту строку, он как бы "слышит" следующее: увеличь значение переменной I на единицу и вернись назад к строке FOR I (строка 10).

Когда компьютер возвращается к строке 10, он проверяет, чему равно значение переменной I. Если это значение меньше 5, то компьютер опять проходит весь цикл, поместив в переменную I следующее число из последовательности.

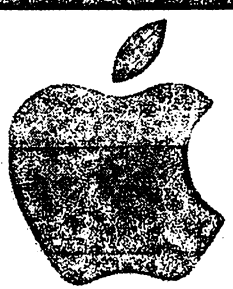
Во второй раз на экране выводится "Приветствуем вас!". И так происходит по кругу столько раз, сколько задано максимальных значений параметра цикла (у нас TO 5, т.е. "до 5").

Если значение параметра I больше 5, компьютер переместится в строку, следующую за NEXT I, если таковая существует (у нас это строка 40). В этом случае NEXT I (строка 30) является последней строкой в цикле, так что, когда значение больше 5, программа заканчивается.

Параметр цикла можно обозначить любым допустимым в Бейсике именем переменной: A,B,...X1,X2,...X9... Его начальное и конечное значения не обязательно должны быть, как в приведенном примере, целыми положительными числами.

Пусть нам нужно посчитать значения функции $y=3x^2$ при изменении аргумента x от —10 до +10. Для этого достаточно ввести в ЭВМ следующую программу:

Технические устройства и механизмы



1977 г.

Молодые американцы С.Джобс и В.Возняк организовали предприятие по изготовлению персональных компьютеров "Apple" ("Яблоко"), предназначенных для большого круга непрофессиональных пользователей.

```
10 FOR X=-10 TO 10
20 Y=3*X^2
30 PRINT X,Y
40 NEXT X
50 STOP
```

И выполнить программу:

```
RUN (ВК)
-10          300
-9           243
-8           192
-7           147
-6           108
-5           75
-4           48
-3           27
-2           12
-1           3
 0           0
 1           3
 2           12
 3           27
 4           48
 5           75
 6           108
 7           147
 8           192
 9           243
 10          300
```

Технические устройства и механизмы

1982 г.

Американская фирма по производству вычислительной техники IBM, занимавшая до этого ведущее положение по выпуску больших ЭВМ, приступила к изготовлению профессиональных персональных компьютеров IBM PC.

А если нужно проследить изменения Y в другом интервале, скажем, от –17 до –3?

Тогда достаточно переписать строку 10 иначе:

```
10 FOR X=–17 TO -3
```

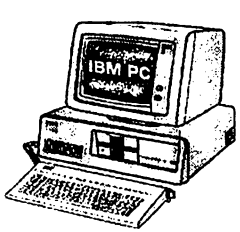
А все остальные строки сохранить неизменными.

В этом примере параметр цикла играет две роли: не только управляет ходом выполнения циклических операций, но и сам участвует в вычислениях. Не зря оператор цикла в начале этого параграфа был назван мощным. Именно потому, что он может многое. С его помощью выполняются многие работы, вычислительные и невычислительные процедуры. Но перед тем, как перейти к выяснению его "способностей", разберем еще одно важное обстоятельство.

Компьютер не обязательно должен подсчитывать циклы, прибавляя по единице. Он может считать их и через 2, 3, 4 или используя любое другое приращение.

Функция STEP (**шаг**), включенная в цикл FOR...NEXT, указывает, каким должно быть это приращение. Тогда оператор начала цикла будет состоять не из двух слов: FOR – TO, а из трех: FOR – TO – STEP. Первая форма является частным случаем, вторая – общим. При шаге, равном 1, слово STEP можно не писать: компьютер сам "понимает", что шаг равен 1, его размер задается, как говорят программисты, по умолчанию. Как раз все, что мы рассматривали выше, было частным случаем оператора FOR...NEXT.

Технические устройства и механизмы



1985 г.

Фирма IBM, совершенствуя компьютеры IBM PC, выпускает совместимые с ними модели IBM PC/XT (1985 г.), а затем IBM PC/AT и IBM PS/2 (1987 г.).

1993 г.

Фирма IBM продолжает выпускать IBM-совместимые модели, но на основе нового процессора – "Pentium".

Например:

```
10 FOR J=1 TO 13 STEP 3
20 PRINT J
30 NEXT J
40 END

RUN (ВК)
 1
 4
 7
 10
 13
```

Каждый раз, когда компьютер добирается до строчки 20, он выводит на экран значение, записанное в J. И всякий раз, попадая на строку 30, содержащую оператор NEXT, он прибавляет к значению J число 3.

С помощью функции STEP переменную цикла можно как увеличивать, так и уменьшать:

```
10 FOR K=16 TO 8 STEP -4
20 PRINT K
30 NEXT K
40 PRINT "ПОЕХАЛИ!"
50 END

RUN (ВК)
 16
 12
 8
ПОЕХАЛИ!
```

Значение переменной К сначала равнялось 16, а потом уменьшилось до 8 с шагом −4.

И опять есть еще дополнительные возможности – шаг может быть не только целым положительным или отрицательным – он может быть и величиной дробной.

Дополним объяснения работы оператора цикла блок-схемой. Но сначала сделаем обозначения:

X – параметр цикла;

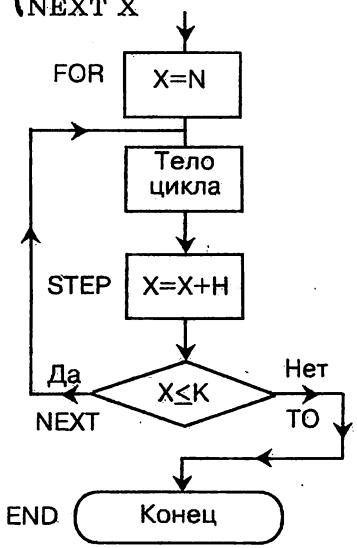
N – начальное значение;

K – конечное значение;

H – шаг.

Тогда условно оператор FOR...NEXT запишется так:

```
FOR X=N TO K STEP H
ТЕЛО ЦИКЛА (ОПЕРАТОР
ИЛИ СЕРИЯ ОПЕРАТОРОВ)
NEXT X
```




```
10 X=N
20
.
.
.
90
100 X=X+H
110 IF X<=K THEN 20
120 END
```

Рис. 25

Такую блок-схему оператора FOR...NEXT (рис. 25) можно отнести к базовой структуре ЦИКЛ–ДО, которую условно рисуют так:

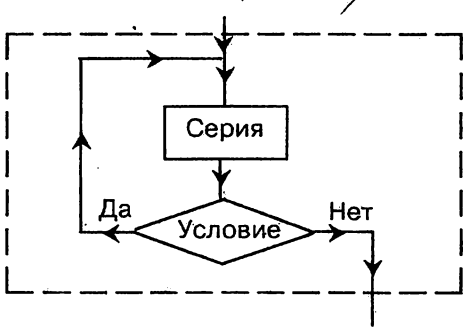


Рис. 26. Базовая структура ЦИКЛ–ДО

Все рассмотренные нами программы с использованием оператора FOR...NEXT делали печать текущего значения переменной (параметра) на экране. И мы видели эти значения. Однако компьютер будет помещать числа в переменную и в том случае, если мы не дадим команду PRINT. Такой вариант невидимого счета может быть использован для того, чтобы сделать в программе временную задержку. Например, введем:

```
10 CLS
20 PRINT"ПРОВЕРЯЕМ ВАШУ ЗРИТЕЛЬНУЮ ПАМЯТЬ"
30 PRINT"АКЦЕНТУИРОВАННАЯ ЛИЧНОСТЬ"
40 CLS
50 PRINT"ВСЕ СТРОКИ ИСЧЕЗЛИ"
```

При выполнении программы вы не успеете прочитать на экране первые две фразы, т.к. после ее вывода компьютер выполнит строку 40 и сотрет их. Поэтому удобно использовать в этой программе **паузу**, чтобы задержать эти строки на экране для прочтения.

Для организации паузы используют пустой цикл:

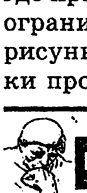
```
32 FOR X=1 TO 3000
34 NEXT X
```

Такой цикл называется также **циклом ожидания**. Пауза всегда может быть увеличена или уменьшена путем изменения значения конечного числа цикла.

Напомним, что вместо двух строк 32 и 34 вы можете написать только одну:

```
32 FOR X=1 TO 3000: NEXT X
```

Обычно паузы используют в таком месте программы, где при выполнении пользователю необходимо небольшое, ограниченное время для запоминания какого-то текста, рисунка и т.д., а затем экран стирался бы и автоматически продолжался ход выполнения этой программы.



Проверьте свои знания ?

1. Как записываются и как действуют операторы цикла FOR ...NEXT?

2. Разъясните понятия: "тело цикла", "параметр цикла", "шаг цикла".

3. Будет ли выполнена серия операторов тела цикла, если начальное значение параметра цикла больше конечного?

4. Какие значения может принимать константа STEP?

5. Можно ли использовать в вычислениях текущее значение параметра цикла?

6. Что такое пустой цикл? Для каких целей он используется?

Тренировка

I. Используя оператор FOR...NEXT, напишите программу расчета и вывода на печать значений функции Y=5/X при изменении аргумента X от —5 до +5 с шагом 2.

II. В нижеприведенной программе измените функцию STEP таким образом, чтобы на экран выводились числа: а) 2, 6, 10, 14; б) 2, 2.5, 3, 3.5,... 14.

```
10 FOR K=2 TO 14 STEP 2
20 PRINT K
30 NEXT K
40 END
```

III. Напишите программу, которая будет выводить на экран текст: "КЛАРА У КАРЛА УКРАЛА КОРАЛЛЫ, А КАРЛ У КЛАРЫ УКРАЛ КЛАРНЕТ" в течение 5 секунд, а затем стирать экран и проверять правильность записанной вами скороговорки.

Примечание. Чтобы задать в цикле задержки подходящее число, предварительно проведите эксперимент с его различными значениями.

IV. Покажите результат выполнения программы (устно):

```
10 FOR X=1.5 TO 0 STEP -0.5
20 PRINT X, X^2
30 NEXT X
```

V. Напишите программу, которая с помощью цикла FOR... NEXT вычисляет сумму первых 10 целых чисел и печатает результат.

VI. Используя оператор FOR...NEXT, составьте программу подсчета суммы 12 членов арифметической прогрессии, в которой $a_1=1$, $d=-3$.

VII. Составьте программу подсчета и печати результата суммы 10 членов геометрической прогрессии, в которой $U_1=2$, $q=3$. Предложите два способа решения:

а) линейный способ (используя формулу суммы геометрической прогрессии);

б) используя цикл FOR...NEXT (без использования формулы).

VIII. Составьте программу, вычисляющую n! (факториал).

IX. Составьте программу определения суммы вклада на сберегательной книжке через заданное количество лет. Начальная сумма и процент начисления также задаются.

§ 32. Структурная организация алгоритмов

Среди свойств алгоритмов (см. §15), пожалуй, главным является **правильность алгоритма**. Для этого и составляем алгоритм, чтобы он правильно решал поставленную задачу. Но дело как раз в том и состоит, что для удовлетворения этого естественного требования алгоритм должен быть:

– легок для понимания;

– прост для доказательства правильности;

– удобен для модификации.

Для удовлетворения этих требований и была предложена специальная методика организации алгоритмов, получившая название **структурного подхода**. Соответственно программирование таких алгоритмов было названо **структурным программированием**.

Суть структурного подхода в конструировании блок-схем алгоритмов предполагает:

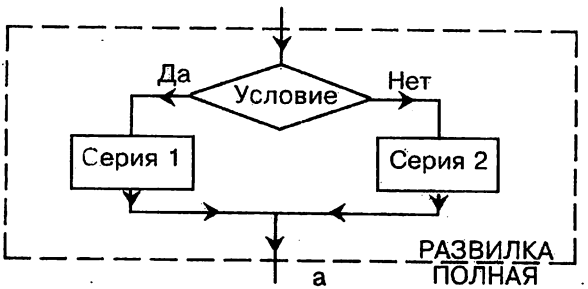
1. Алгоритм как бы "собирается" из трех основных (базовых) структур: развилка, цикл, следование, каждая из которых имеет один вход и один выход (рис. 27).

2. Разработка алгоритма "сверху вниз" (нисходящая разработка).

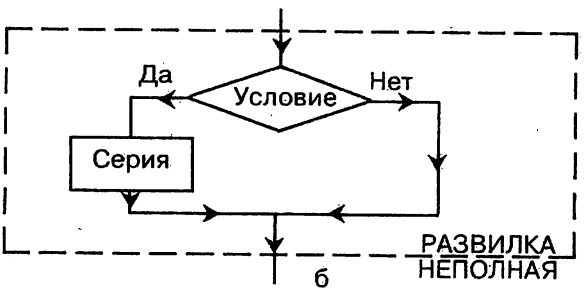
3. Сквозной структурный контроль.

Поясним вышеизложенное:

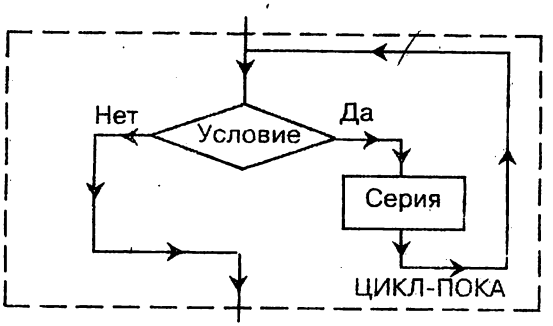
1. Мы с вами уже разобрали две базовые структуры: РАЗВИЛКА и ЦИКЛ. Причем каждая имеет по две модификации: РАЗВИЛКА ПОЛНАЯ и РАЗВИЛКА НЕПОЛНАЯ (см. рис. 27 а,б); ЦИКЛ-ПОКА и ЦИКЛ-ДО (см. рис. 27 в,г).



а

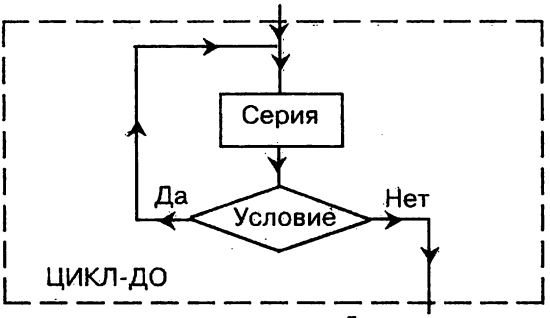


б

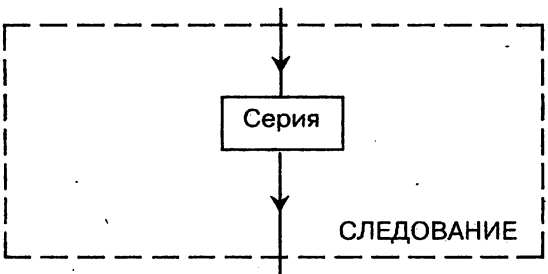


в

Рис. 27. Базовые структуры: а, б – РАЗВИЛКА; в – ЦИКЛ



г



д

Рис. 27 (окончание). Базовые структуры: г – ЦИКЛ; д – СЛЕДОВАНИЕ

Базовая структура РАЗВИЛКА (см. рис. 27 а, б) применяется, когда в зависимости от условия нужно выполнить либо одну, либо другую серию действий. Расстановка значений истинности условия – "да" и "нет" – на приведенных рисунках условна, на практике она вытекает из смысла конкретной программируемой задачи. В отдельных случаях для удобства записи условие заменяется своим отрицанием, и тогда значения "да" и "нет" меняются местами.

Базовая структура ЦИКЛ может быть двух видов. ЦИКЛ-ДО (см. рис. 27 г) применяется при необходимости выполнить какое-либо вычисление несколько раз до выполнения некоторого условия. Особенность этого цикла в том, что он всегда выполняется хотя бы один раз, так как первая проверка условия выхода из цикла происходит после того, как серия действий (тело цикла) выполнена. ЦИКЛ-ПОКА (см. рис. 27 в) отличается от ЦИКЛА-ДО тем, что

проверка условия проводится до выполнения серии действий и, если при первой проверке условие выхода из цикла выполняется, то серия действий не выполняется ни разу. Расстановка значений истинности "да" и "нет" в изображении структур циклов, вообще говоря, может быть произвольной. Однако способ их расстановки, принятый на рис. 27 в, г, не случаен, а является выражением определенного стандарта, объясняющего в том числе и сами названия структур (ЦИКЛ-ПОКА и ЦИКЛ-ДО). В первом случае (см. рис. 27 в) серия действий исполняется, ПОКА – условие истинно ("да"), условие в этой базовой структуре называют **условием продолжения цикла**. Во втором случае (см. рис. 27 г) серия действий исполняется ДО истинности условия; условие в этой структуре называют **условием окончания цикла**.

Разберем базовую структуру СЛЕДОВАНИЕ (см. рис. 27 д). Эта структура состоит из одного или нескольких функциональных блоков (серии действий), каждый из которых в простейшем случае может быть арифметическим элементом. Структура следования означает, что два или несколько блоков размещены друг за другом. В программе это реализуется последовательным размещением операторов.

2. **Принцип нисходящей разработки** заключается в следующем. На начальном этапе дается крупноблочное описание алгоритма. Для этого подсоединяют одну структуру к другой, образуя последовательность структур (реализуемость такой методики очевидна, поскольку каждая из структур имеет один вход и один выход). Затем отдельные блоки детализируются, т.е. любой функциональный блок (серия действий) может быть заменен блоком любой базовой структуры. В свою очередь в любом базовом блоке его функциональный блок (серия) вновь (при необходимости) может быть заменен на другой базовый блок. Таким образом, сколько угодно сложный по структуре алгоритм развивается не только "вширь", но и "вглубь". При этом, разумеется, нужно так

складывать блоки, чтобы полученный в итоге алгоритм правильно решал поставленную задачу, – сам по себе структурный подход не дает автоматического разрешения этой проблемы.

Конструирование блок-схемы из базовых структур напоминает игру в детские кубики. Нужно построить изображение из разрозненных кубиков, причем каждый кубик сам может складываться из более мелких кубиков (рис. 28).

В информатике же любая задача представляется в виде серии подзадач, затем каждая подзадача подразделяется на более простые и т.д. Детализация может быть доведена до элементарных действий, которыми могут оказаться команды машины или операторы языка программирования.

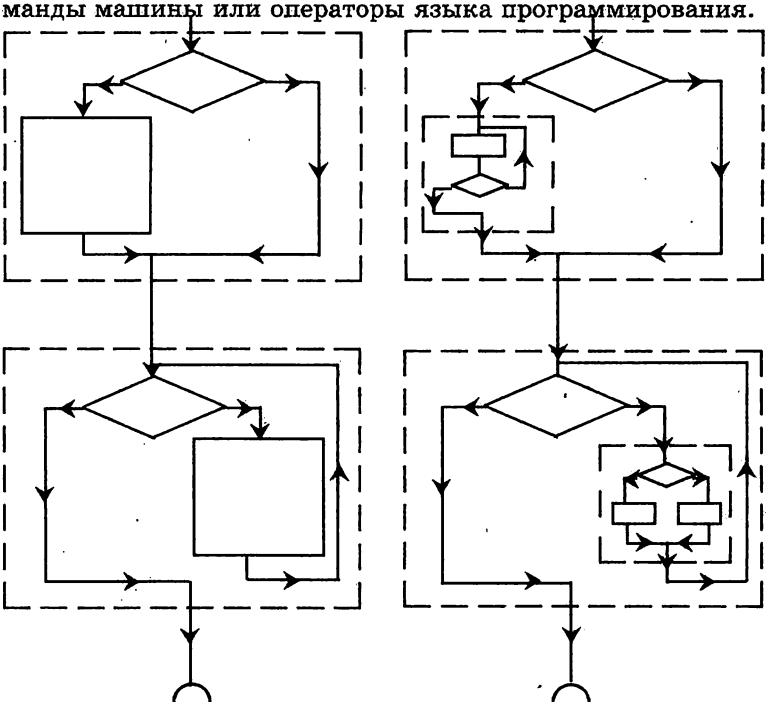


Рис. 28. Замена функциональных блоков базовыми структурами

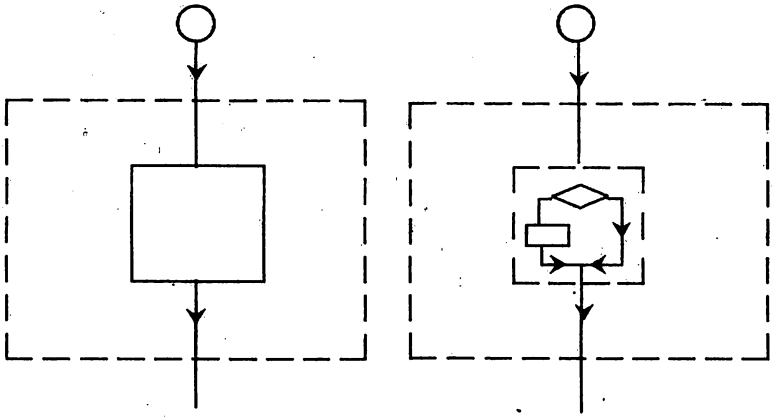


Рис. 28 (окончание). Замена функциональных блоков базовыми структурами

3. Объясним **принцип сквозного структурного контроля.**

Алгоритм, организованный по этой методике, всегда легко подвергнуть структурному анализу и на нем можно выделить составные элементы. Так, например, на рис. 22 изображен структурный алгоритм, который представляет собой базовую структуру РАЗВИЛКА – ПОЛНАЯ, при этом "серия 2" (тело цикла) является тоже базовой структурой РАЗВИЛКА – ПОЛНАЯ (обведено пунктиром).

Но использование структурного подхода не самоцель. В некоторых случаях стремление во что бы то ни стало остаться в рамках структурного подхода приводит к необоснованному усложнению алгоритма и потере его наглядности и естественности. Если учесть, что структурное программирование имеет цель не подчинить программы каким-то правилам, а сделать их более удобными для восприятия, то в ряде случаев приходится отказываться от строгого соблюдения правил структурного подхода.

Но в целом структурный подход в разработке алгоритмов дает больше положительных моментов, чем отрицательных.

Мы исходим из того, что каждый алгоритм может быть представлен в структурном виде. Однако в процессе поиска алгоритма это может удаваться не сразу. Иногда при-

ходится использовать специальные приемы, помогающие преобразовывать неструктурные алгоритмы в структурные. Один из таких приемов – **размножение блоков:**

Например: дана блок-схема неструктурного алгоритма (рис. 29 а):

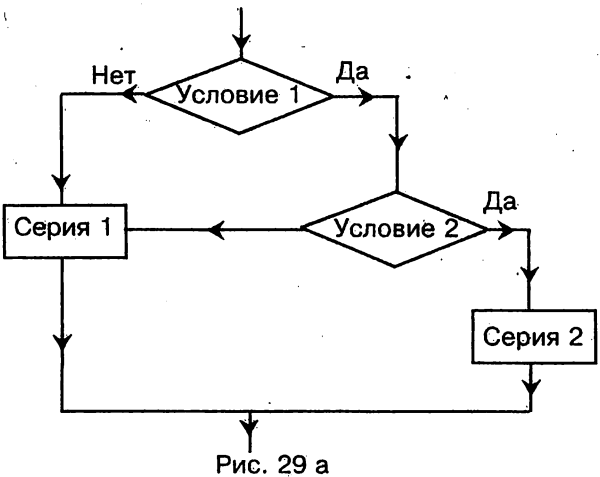


Рис. 29 а

Чтобы привести этот алгоритм к структурному виду, необходимо продублировать блок "серия 1" (рис. 29 б):

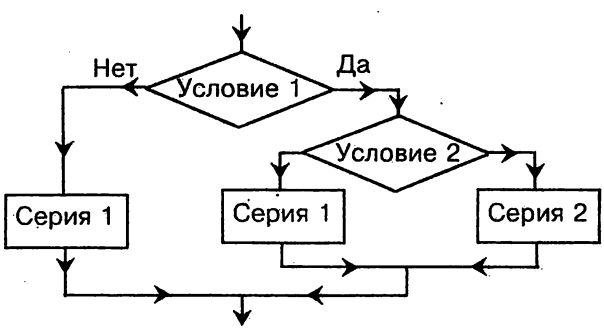


Рис. 29, б

Другие, более сложные приемы преобразования неструктурных алгоритмов в структурные мы разберем в последующих параграфах.

Проверьте свои знания

1. В чем заключается структурный подход к разработке алгоритмов? Какие основные требования обеспечивают разработку алгоритмов с использованием этого подхода?
2. В чем состоит суть структурного подхода в конструировании блок-схем алгоритмов?
3. В чем состоит суть базовой структуры РАЗВИЛКА?
4. В чем состоит основное отличие базовых структур ЦИКЛ-ДО и ЦИКЛ-ПОКА?
5. Поясните основное содержание базовой структуры СЛЕДОВАНИЕ.
6. В чем заключается принцип нисходящей разработки алгоритмов?
7. В чем состоит принцип сквозного структурного контроля?

Тренировка

I. На рис. 30 представлена блок-схема алгоритма решения функции:

$$Y=\begin{cases} e^{\sqrt{x+1}}, \text{ если } \sin x>0{,}4 \\ \operatorname{ctg} x, \text{ если } \sin x<0{,}4 \end{cases}$$

при изменении аргумента x от 1 до 7.

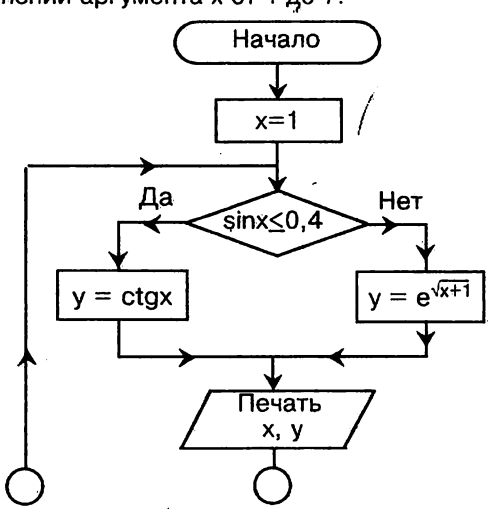


Рис. 30

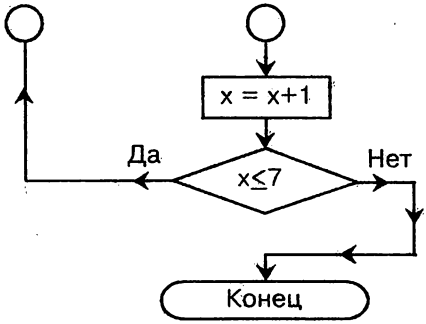


Рис. 30 (окончание)

Составьте структурную схему, выделив в ней известные вам базовые структуры.

II. Выделите базовые структуры в блок-схемах решения заданий к тренировке §16, номера I, II, III.

III. На рис. 31 представлена модульная блок-схема решения уравнения вида $ax^3 + bx = 0$ (см. тренировку §16, номер IV). Модули 1 и 2 не являются структурными. Используя прием размножения блоков, преобразуйте эту блок-схему в структурную.

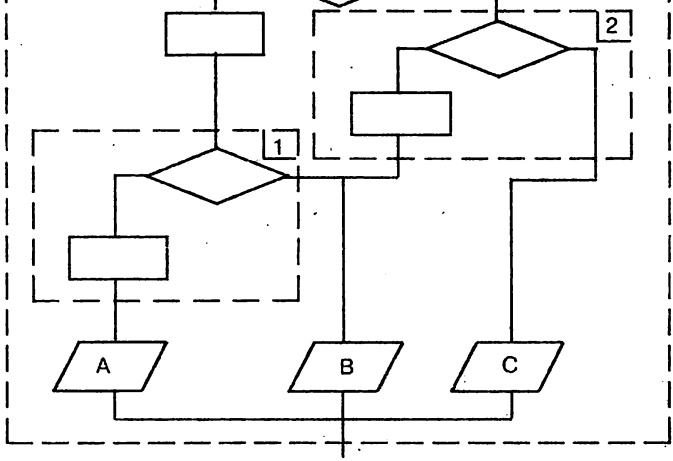


Рис. 31

§ 33. Дополнительные возможности при организации циклов

Рассматривая операторы FOR...NEXT, мы оставили без внимания один очень важный момент: у нас есть возможность поставить один цикл внутри другого. Например:

```
10 FOR X=1 TO 10
20 FOR Y=1 TO 5
30 PRINT X,Y
40 NEXT Y
50 NEXT X
```

Рис.32. Вложенные циклы

Получилась конструкция, похожая на матрешку, – цикл вложен в цикл (рис. 32). Такие циклы называются **вложенными циклами**.

Внешний цикл ограничен строками 10 и 50. Внутренний цикл – это строки 20-40. Обратите внимание на важную деталь: сначала в программе записывается NEXT Y (для закрытия внутреннего цикла), а затем NEXT X (для закрытия внешнего цикла). Если их поменять местами, то программа не заработает и будет выведено сообщение об ошибке. Как же работает такая программа?

Сначала принимается значение X=1 (первый столбец) и начинает работать внутренний цикл. По его окончании, после стро-

Развивающие идеи

Конец V–начало IV в. до н.э.

В произведениях Гомера и Аристофана упоминается *пальцевый счет.*

IV в. до н.э.

Древнегреческий ученый Аристотель основал *дедуктивную логику.*

ки 40, в строке 50 проверяется, достиг ли параметр внешнего цикла X своего конечного значения. Если нет, то компьютер отсылается на строку 10, принимается новое значение X=2, внутренний цикл повторяется уже при этом значении. И все повторяется 10 раз.

Для того чтобы рассмотреть это яснее и подробнее, можно попросить компьютер помочь. Изменим строку 30:

```
30 PRINT "X РАВНО";X;"...ТЕПЕРЬ Y РАВНО";Y
```

и добавим строку:

```
35 IF Y=5 THEN PRINT "ЗАКОНЧИЛИ ВНУТРЕННИЙ
ЦИКЛ ДЛЯ ЗНАЧЕНИЯ X, РАВНОГО";X
```

Если дать команду RUN, то на экране получим различные значения, которые принимают одна за другой переменные X и Y.

Начальные и конечные значения цикла FOR...NEXT не обязательно должны быть представлены программистом. Они могут быть введены с помощью оператора INPUT пользователем.

Например:

```
10 INPUT A
20 INPUT B
30 INPUT C
40 FOR X=A TO B STEP C
50 PRINT X
60 NEXT X
70 END
```

Запустите программу на выполнение и дайте переменным A, B и C значения, какие вы хотите, например, 0, 20

Развивающие идеи

Первая половина IX в.

В трудах аль Хорезми обобщены достижения арабской математики, введены термин "алгебра" и понятие "*алгоритм*", который означал решение задач с помощью уравнений на основе установленных правил.

и 3. Вы получите список чисел, начинающихся от 0, через три единицы.

В случае необходимости можете выйти из цикла FOR... NEXT. Добавьте в эту программу такие строки:

```
45 IF X=9 THEN GOTO 100
100 PRINT"Я ВЫШЕЛ ИЗ ЦИКЛА, ПОТОМУ ЧТО X ПРИНЯЛ ЗНАЧЕНИЕ 9"
```

Значит, с операторами типа GOTO или IF...THEN вы можете выйти из цикла, и цикл прервется. Единственное, что вы не должны делать, – это входить в цикл не через его начало, так как в этом случае компьютер зафиксирует ошибку.

И еще пример: вы можете написать программу для печати таблицы умножения, которая будет спрашивать пользователя, какие числа он хочет поставить в качестве начального и конечного значений:

```
10 ' ПЕЧАТЬ ТАБЛИЦЫ УМНОЖЕНИЯ
20 PRINT "ОТКУДА Я ДОЛЖЕН НАЧАТЬ";
30 INPUT S
40 PRINT "И ГДЕ Я ДОЛЖЕН ОСТАНОВИТЬСЯ";
50 INPUT F
60 FOR I=S TO F
70 FOR J=S TO F
80 Z=I*J
90 PRINT Z;
100 NEXT J
110 PRINT
120 NEXT I
```

Внутрь компьютера встроены "электронные часы" (таймер), которые задают ритм работы всего устройства. На

Развивающие идеи

1614 г.

Шотландский математик Джон Непер опубликовал "Описание таблиц логарифмов" – первое руководство по вычислениям с помощью логарифмов, идея которых у него возникла примерно лет на 20 раньше.

эти часы подаются импульсы с частотой от 4 млн имп/с и выше.

Зная возможности операторов FOR...NEXT, мы теперь сами можем написать программу, которая заставила бы компьютер работать в качестве электронных часов:

```
5 CLS
10 FOR M = 0 TO 59
20 FOR S = 0 TO 59
30 PRINT M; ":"; S
50 CLS
60 NEXT S
70 NEXT M
```

В этой программе два вложенных цикла – один для подсчета секунд, другой – для подсчета минут. Для каждой минуты цикл секунд повторяется 60 раз. Если вы попытаетесь выполнить эту программу на компьютере, то отсчет времени, возможно, будет меняться очень быстро. Вам придется вставить "цикл задержки", причем верхнюю границу изменения переменной этого цикла нужно подобрать такой, чтобы ваши компьютерные часы "тикали" синхронно с реальным временем. Это будет строка, подобная следующей:

```
40 FOR K=1 TO 1000:NEXT K
```

Из этого примера видно, что глубину вложения циклов можно делать любой (в разумных, с точки зрения программы, пределах).

Важно следить, чтобы циклы не скрестились, иначе произойдет ошибка:

Развивающие идеи

В 1617 г. он опубликовал работу об умножении с помощью палочек, названных "палочками Непера".

1670 г.

Немецкий математик Г.Лейбниц попытался создать *алгебру логики и интегральное счисление.*

ПРАВИЛЬНО:

```
10 FOR A=0 TO 7
20 ...
30 ...
40 FOR B=0 TO 10
50 ...
60 ...
70 FOR C=0 TO 100 STEP 5
80 ...
90 ...
100 NEXT C
110 NEXT B
120 NEXT A
```

НЕПРАВИЛЬНО:

```
10 FOR A=0 TO 7
20 ...
30 ...
40 FOR B=0 TO 10
50 ...
60 ...
70 FOR C=0 TO 100 STEP 5
80 ...
90 ...
100 NEXT A
110 NEXT B
120 NEXT C
```

Иногда версии Бейсика допускают форму оператора NEXT без имени переменной. В этом случае строки предыдущей программы записаны были бы в виде:

```
100 NEXT
110 NEXT
120 NEXT
```

Возможен вариант, когда вложенные циклы заканчиваются одной командой NEXT. В этом случае имена переменных после команды NEXT записываются через запятую, причем первыми должны стоять имена переменных более внутренних циклов. Для нашего примера:

```
100 NEXT C,B,A
```

И, наконец, последнее: если используемые в операторе FOR переменные не являются целыми, то при изменениях значения управляющей переменной вычисления будут

Развивающие идеи

1823 г.

Одновременно с английским ученым Ч.Беббиджем работала дочь Джорджа Гордона Байрона леди Ада Лавлейс. Она разработывала первые программы для его машины, заложила многие идеи и ввела ряд понятий и терминов, сохранившихся до настоящего времени.

проводиться приближенно и момент завершения цикла может зависеть от точности вычислений. Это очень неприятный момент, т.к. из-за этого могут время от времени получаться удивительные результаты (например, если в цикле с большим числом повторений используется шаг цикла с дробной частью).

FOR...NEXT можно использовать в комбинации с другими операторами для облегчения программирования. Полезные результаты получаются, когда внутри цикла используют оператор PRINT и стандартную функцию TAB. Тогда можно легко построить простые графики функций, таблицы, гистограммы. Вот некоторые примеры построения графиков:

```
5 ' ГРАФИК ФУНКЦИИ ВИДА Y=X
10 FOR X=1 TO 15
20 PRINT TAB(X);"*"
30 NEXT X
40 END
RUN
*
  *
    *
      *
        *
          *
            *
              *
                *
                  *
                    *
                      *
```

Развивающие идеи

1831 г.

Английский ученый Майкл Фарадей открыл индукционные токи. Он же изготовил динамомашину "Фарадея".

1847 г.

Английский математик Джордж Буль, изучая основы логики, ввел в обращение новый раздел математики. Его назвали *Булева алгерба*. Каждая величина в ней может

```
                *
                 *
                  *
                   *
5 ' ГРАФИК ФУНКЦИЙ ВИДА Y= X^2
10 FOR X=-5 TO 5
20 PRINT TAB(X^2);"*"
30 NEXT X

RUN
                          *
                    *
              *
         *
    *
  *
*
  *
    *
         *
              *
                    *
                          *

5 ' ГРАФИК ФУНКЦИЙ ВИДА Y=SQR(X)
10 PRINT TAB(20);"*"
20 FOR X=4 TO 40 STEP 4
30 K=3*SQR(X)
40 PRINT TAB (-K+20);"*";TAB (K+20);"*"
50 NEXT X
RUN
```

Развивающие идеи

принимать только одно из двух значений: истина и ложь, или 0 и 1. Эта алгебра очень пригодилась создателям современных компьютеров. Ведь компьютер понимает только два символа: ноль и единица. Его считают основоположником современной *математической логики*.

1884 г.

Известный американский изобретатель Томас Эдисон описывает явление *электронной эмиссии*, лежащее в основе ламповой электроники.

Проверьте свои знания

1. Поясните, какие циклы называются вложенными.

2. Даны вложенные циклы:

```
100 FOR I=1 TO 3
110 FOR J=1 TO 2
120 PRINT I,J
130 NEXT J,I
```

Поясните, какие значения будут принимать переменные I и J.

3. Существуют ли ограничения на глубину вложения циклов?

4. Можно ли, не дожидаясь окончания цикла, выйти из него? Как это сделать?

Тренировка

Напишите программу, которая выводила бы мигающее сообщение (например, "ОПАСНОСТЬ!").

Вам потребуется цикл, чтобы многократно очищать экран и выводить сообщение, а также два цикла задержки, чтобы сообщение не мигало слишком часто.

Развивающие идеи

Конец XIX в.

Португальский ученый А. ди Пайва и независимо от него русский ученый П.И.Бахметьев выдвигают принцип *последовательной передачи элементов изображения*, принятый затем в телевидении.

§ 34. Матрицы и массивы

Из математики вы знаете, что **матрицей** называют прямоугольную таблицу, образованную из элементов некоторого множества, состоящую из M строк и N столбцов.

Примером матрицы может служить таблица умножения, названная именем создателя, – таблица Пифагора:

	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9
2	2	4	6	8	10	12	14	16	18
3	3	6	9	12	15	18	21	24	27
4	4	8	12	16	20	24	28	32	36
5	5	10	15	20	25	30	35	40	45
6	6	12	18	24	30	36	42	48	54
7	7	14	21	28	35	42	49	56	63
8	8	16	24	32	40	48	56	64	72
9	9	18	27	36	45	54	63	72	81

Это квадратная матрица, где M=9 и N=9, т.е. M=N

Рис. 33

Но элементами множества в матрице не обязательно должны быть числа. Это могут быть любые элементы: знаки, слова, предложения – если это матрица на бумаге; детали, книги или вообще любые предметы – если это двухмерная полка в реальной жизни.

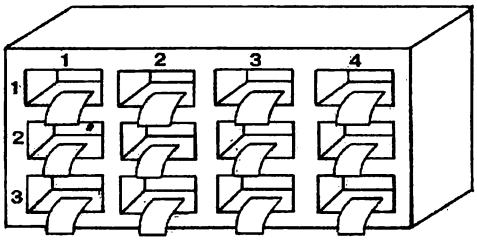
Развивающие идеи

1928 г.

Американский математик Джон Янош (фон) Нейман (уроженец Будапешта) сформулировал основы теории игр. А в середине 40-х годов совместно с Г.Голдстайном и А.Берисом изложил принципы построения ЭВМ, которые используются до сих пор.

1936 г.

Американский математик А.Тьюринг и независимо от него



Это прямоугольная матрица, где M=3, N=4

Рис. 34

Как пользоваться матрицами, вам известно с начальной школы. Скажем, в таблице Пифагора произведение числа можно найти на пересечении нужной строки и нужного столбца. В математике и, как продолжение, в программировании использование матриц имеет и более глубокий смысл. Приведем пример.

Решим систему уравнений с двумя неизвестными. Только договоримся о том, что решать ее будем в общем виде, обозначая коэффициенты при неизвестных так:

$$\begin{cases} a_{11}x+a_{12}y=a_{13} \\ a_{21}x+a_{22}y=a_{23} \end{cases}$$

Здесь первая цифра индекса обозначает номер уравнения, вторая – номер неизвестного.

Приступим к решению.

Умножим правую и левую части первого уравнения на a_{22}, а правую и левую части второго уравнения – на a_{12}:

Развивающие идеи

американский математик и логик Э.Пост (уроженец Польши) выдвинули и разработали концепцию абстрактной вычислительной машины. "Машина Тьюринга" – гипотетический универсальный преобразователь дискретной информации, теоретическая вычислительная система. Тьюринг и Пост показали принципиальную возможность решения автоматами любой проблемы при условии возможности ее алгоритмизации с учетом выполняемых ими операций.

$a_{11}a_{22}x+a_{12}a_{22}y=a_{13}a_{22}$;
$a_{21}a_{12}+a_{22}a_{12}y=a_{23}a_{12}$.

Вычтем из первого уравнения второе:

$$a_{11}a_{22}x - a_{21}a_{12}y = a_{13}a_{22} - a_{23}a_{12}.$$

Получилось уравнение с одним неизвестным, из него найдем

$$x = \frac{a_{13}a_{22} - a_{23}a_{12}}{a_{11}a_{22} - a_{21}a_{12}}.$$

Поступая аналогично, найдем

$$y = \frac{a_{23}a_{11} - a_{13}a_{21}}{a_{11}a_{22} - a_{21}a_{12}}.$$

Присмотримся теперь к ответам. Заметим, что знаменатели дробей равны, а числители "устроены" очень похоже. Вот если бы запомнить порядок записи всех этих попарных произведений и запомнить, когда стоит знак плюс, а когда – минус, то можно было бы решать системы сразу, не делая никаких преобразований.

И такой способ придуман.

Для составления знаменателей этих формул надо уметь правильно комбинировать коэффициенты. Чтобы научиться этому, выпишем коэффициенты в виде такой таблички:

$$\begin{matrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{matrix}$$

А теперь договоримся, что если справа и слева от этой и подобных табличек поставлены вертикальные палочки, то они, эти таблички, означают числа, вычисляемые по следующим правилам:

Развивающие идеи

1937 г.

Американский физик болгарского происхождения Дж.В.Атанасов формирует принципы *автоматической вычислительной машины* на ламповых схемах для решения систем линейных уравнений.

$$\begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = a_{11}a_{22} - a_{12}a_{21}.$$

Умножать надо "по стрелкам", причем если стрелки идут слева – вниз – направо, то произведение надо брать со знаком плюс, если же справа – вниз – налево, то со знаком минус. Запомнить такое правило очень легко.

Итак, со знаменателями все в порядке. А теперь заметьте, что для вычисления числителей надо действовать точно по таким же правилам, только, записывая числитель для x, надо всюду вместо коэффициентов при этом неизвестном писать соответствующие свободные члены уравнений. Точно так же поступим и для второго неизвестного.

Для усвоения этого правила решим конкретную систему уравнений:

$$\begin{cases} 2x + 4y = 18 \\ 3x - 2y = 11 \end{cases}$$

Вот здесь-то нам и понадобится **матрица**. Составим ее из коэффициентов при неизвестных и найдем значение знаменателя (принято говорить: вычислим **определитель** – этим словом называют найденное по описанному нами правилу число, обозначается он знаком "Δ" – дельта):

$$\Delta = \begin{vmatrix} 2 & 4 \\ 3 & -2 \end{vmatrix} = 2 \cdot (-2) - 4 \cdot 3 = -4 - 12 = -16.$$

А теперь остается проделать такие же вычисления для x и y:

$$\Delta x = \begin{vmatrix} 18 & 4 \\ 11 & -2 \end{vmatrix} = 18 \cdot (-2) - 4 \cdot 11 = -36 - 44 = -80;$$

Развивающие идеи

1938 г.

Американский математик и инженер Клод Шеннон и русский ученый В.И.Шестаков в 1941 г. показали возможность аппарата математической логики для синтеза и анализа релейно-контактных переключательных систем.

$$\Delta y = \begin{vmatrix} 2 & 18 \\ 3 & 11 \end{vmatrix} = 2 \cdot 11 - 18 \cdot 3 = 22 - 54 = -32 .$$

Значит

$$x = \frac{\Delta x}{\Delta} = \frac{-80}{-16} = 5; \qquad y = \frac{\Delta y}{\Delta} = \frac{-32}{-16} = 2.$$

Вы, наверное, догадались, что, действуя при помощи определителей, как бы используете стандартные правила формально, можно сказать, как машина. А нам в программировании это как раз и надо! Используя матричный язык, в дальнейшем придумали множество матричных правил. Мы не будем здесь их рассматривать, заметим только, что с изучения определителей и решения систем уравнений со многими неизвестными начинается очень важный раздел современной математики – так называемая **линейная алгебра**. Она-то и лежит в основе матричных преобразований в программировании.

Если у вас имеется группа связанных данных, например, адресов, номеров телефонов и дней рождения некоторых людей, то можно использовать массивы и помещать в одну строку такого массива всю информацию, касающуюся одного человека. Это облегчает нахождение всей информации об одном человеке или поиск, скажем, дня рождения данного человека.

В отличие от математики, в программировании вместо понятия *матрица* используют понятие *массив*.

МАССИВ – это упорядоченный набор величин, обозначаемых одним именем; доступ к элементу массива осуществляется по его номеру.

Развивающие идеи

1948 г.

Американский математик Норберт Винер выпустил в свет книгу "Кибернетика", которая положила начало развитию *теории автоматов* и становлению *кибернетики* – науки об управлении и передаче информации. Также Клод Шеннон выпускает книгу "Математическая теория передачи информации".

Массивы могут быть одномерными и многомерными (двух-, трехмерными и т.д.). Примером одномерных массивов может быть список фамилий учеников класса или вообще любой список числовых или символьных констант. Он может быть представлен одномерным столбцом или одномерной строкой значений:

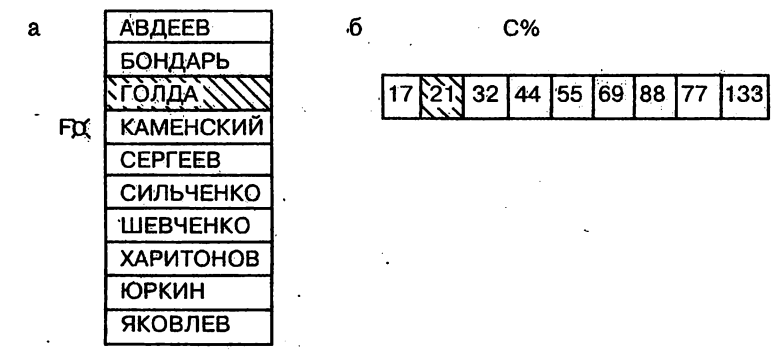


Рис. 35. Одномерные массивы

Примером двухмерного массива может быть классный журнал, аттестат зрелости или вообще любая таблица значений (т.е. матрица).

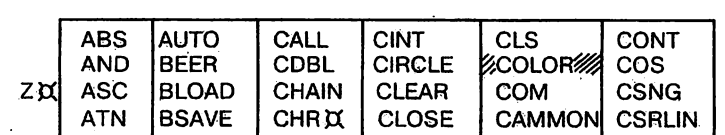


Рис. 36. Двухмерный массив
(зарезервированных слов языка Бейсик)

Развивающие идеи

1952 г.

Инженер из Великобритании Дж.Даммер на конференции в Вашингтоне по элементам электронных схем выдвинул идею возможности создания *интегральных схем*. Однако в продажу первая интегральная схема на пластине кремния поступила лишь в 1961 г.

Примером трехмерного массива может быть куб, в каждой ячейке которого находится элемент (значение) трехмерного массива.

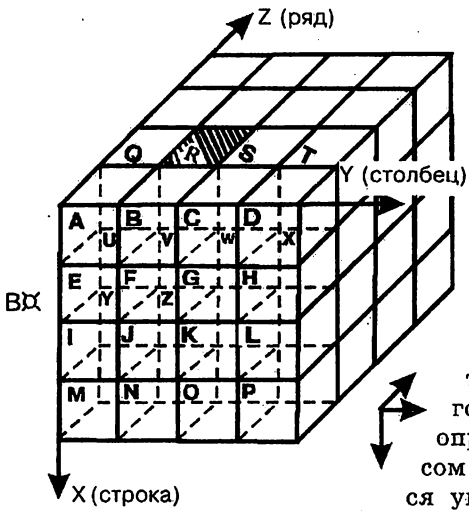


Рис. 37. Трехмерный массив (букв латинского алфавита)

Для массивов с числом измерений больше 3 трудно найти простой геометрический образ (по крайней мере, мы не знаем никого, кто мог бы нарисовать такой массив).

Чтобы находить определенный элемент из массива, существуют индексы, т.е. положение каждого элемента в массиве определяется его индексом. В этом и заключается упорядоченность. Индексы принято указывать в скобках после имени массива. Например, для рис. 35а значение F$(3) означает третий элемент массива F$, т.е. F$(3)="ГОЛДА", а для рис. 35б значение C%(2) означает второй элемент целочисленного массива C%, т.е. C%(2)=21.

Индексом может служить не только константа, но и переменная, что дает возможность задавать действия, относящиеся к любому элементу массива.

Развивающие идеи

1954 г.

Одновременно и независимо Н.Г.Басовым и А.М.Прохоровым в СССР и Ч.Таунсом в США был предложен принцип создания первого в мире *генератора квантов* электромагнитного излучения. Спустя несколько лет (в 1960 г.) в США был создан первый лазер.

Например, Z$(I,J) – элемент, стоящий на пересечении I-й строки и J-го столбца двухмерного символьного массива. Для рис. 36: если I=2, а J=5, тогда Z$(2,5) = "COLOR".

Для примера решения системы линейных уравнений коэффициенты уравнений, представляющие собой переменные, имеют два индекса, например, a_{12}. Мы их будем обозначать как A(1,2). В роли индексов могут выступать имена переменных, например, A(K,L).

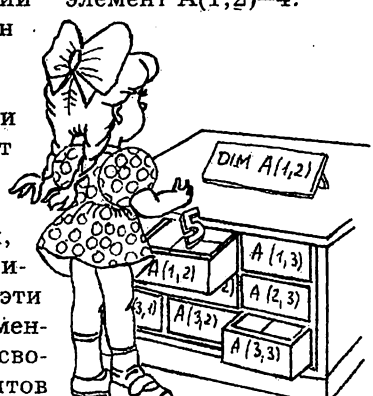
Множество переменных с двумя индексами образует двухмерный массив. Такой массив обычно представляют в виде матрицы, имеющей несколько строк и столбцов. Ниже приведен массив A:

$$\text{СТРОКИ}\left\{\begin{matrix} A(1,1) & A(1,2) & A(1,3) \\ A(2,1) & A(2,2) & A(2,3) \end{matrix}\right.$$

СТОЛБЦЫ

В массиве A две строки и три столбца, у элементов его первый индекс – номер строки, второй – номер столбца. Для нашей системы уравнений элемент A(1,2)=4.

На рис. 37 представлен трехмерный символьный массив B$ (X,Y,Z), где X – строка, Y – столбец; Z – ряд. Если X=1; Y=2; Z=2, то элемент B$(1,2,2)="R".



Предполагая работать при решении задачи с массивами, необходимо описать эти массивы в программе заранее. Зная эти описания, ЭВМ сумеет своевременно и правильно заготовить в своей памяти место для элементов (чисел), образующих массивы.

Для описания массивов используется оператор DIM (сокращение англ. слова dimension – "размерность").

Например:

```
10 DIM F$(10), C%(9)
```

За ключевым словом DIM следует имя массива, а за ним, в свою очередь, – его наибольший размер, заключен-

ный в круглые скобки. Если элементы массива целые или символьные, то рядом с именем массива необходимо записать соответствующее обозначение (%,¤). Одним оператором можно описывать несколько массивов, отделяя их друг от друга запятыми.

В нашем примере символьный массив F¤ содержит 11 символьных элементов, и целочисленный массив C% имеет 10 численных элементов. Вы спросите, почему длина отводимой памяти не равна длине, заданной в операторе DIM? Ответ простой. Дело в том, что первый элемент массива имеет номер 0. Наличие именно этого дополнительного элемента и приводит к тому, что массив занимает на один элемент больше памяти, чем это задано в его описании. Однако нумерация элементов с нуля для большинства задач неудобна, поэтому мы будем игнорировать нулевые индексы без дополнительных замечаний, т.е. будем считать, например, что в соответствии с описанием массивов

```
20 DIM Z¤(4,6), B¤(4,4,4)
```

Этим оператором дается указание ЭВМ зарезервировать место в памяти двухмерного символьного массива Z¤, содержащего 24 элемента, располагающихся в 4 строках и 6 столбцах, и место для символьного массива B¤, содержащего 64 элемента, располагающихся в 4 строках, 4 столбцах и 4 рядах.

И последнее, инструкция DIM помещается в программе до операторов использования массивов, предпочтительно в голове программы.

Проверьте свои знания

1. Что такое матрица?
2. Можно ли сказать, что, используя матрицы при решении систем уравнений, мы формально производим решение? Почему?
3. Дайте определения массива.
4. Какие вы знаете массивы? Покажите их геометрическое представление.
5. Как обозначают массивы?
6. Какой оператор служит для описания массивов? Где и как он используется в программах?
7. Почему ЭВМ резервирует больше места в своей памяти, чем это требует оператор DIM?

Тренировка

I. Составьте алгоритм решения линейных систем с помощью определителей

$$a_{11}x+a_{12}y=a_{13}$$
$$a_{21}x+a_{22}y=a_{23}$$

Представьте его в графическом виде.

II. Составьте программу для решения линейных систем в общем виде, предусмотрев ввод коэффициентов с помощью оператора INPUT. Проверьте ее работу на следующих системах:

$$\begin{cases}14x - 9y = 24\\ 7x - 2y = 17\end{cases} \qquad \begin{cases}4x + 7y = 23\\ 6x - 2y = -28\end{cases}$$

III. Пусть имеется массив x_1, x_2, ..., x_n. Требуется составить алгоритм вычисления среднего арифметического этих чисел, т.е.

$$S=\frac{x_1+x_2+\ldots x_n}{n}.$$

IV. Даны массивы:

N%

а) ПОНЕДЕЛЬНИК;
ВТОРНИК;
СРЕДА;
ЧЕТВЕРГ;
ПЯТНИЦА;
СУББОТА;
ВОСКРЕСЕНЬЕ

M%

б) JAN FEB MAR APR MAY JUN
AUG SEP OCT NOV DES

D%

в) 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31

TR

г)

COSX	SINX	1	−STGX
SINX	COSX	CTGX	1
SINX	−COSX	1	1
COSX	SINX	COSX	1
TGX	SECX	−1	COSECX

С помощью оператора DIM опишите предложенные массивы.

§ 35. Операторы READ...DATA (читать... данные)

Наряду с операторами LET и INPUT связка READ ...DATA также позволяет помещать различные значения в переменные. С ее помощью программист может ввести в компьютер определенное количество информации и четко контролировать, когда и где эта информация будет поступать в переменные или массивы.

Например:

```
10 LET A=42.73
20 LET B%=5
30 LET C¤ ="КНИГА"
40 PRINT A,B%,C¤
50 END

10 INPUT"ВВЕДИТЕ ЗНАЧЕНИЯ: A,B%,C¤";A,B%,C¤
20 PRINT A,B%,C¤
30 END

10 READ A
20 READ B%
30 READ C¤
40 DATA 42.73,5,КНИГА
50 PRINT A,B%,C¤
60 END
```

Оказывается, если в оператор INPUT ввести те же константы 42.73, 5 и "КНИГА", то все три приведенные программы будут выполнять одно и то же, но, как вы увидите далее, третий вариант ввода констант – самый удобный.

Что же происходит в последней программе?

Выполняя строки 10, 20, 30, компьютер читает (READ) сначала первое число (42.73) в переменную A из блока данных (DATA) строки 40, затем второе число (5) – во вторую переменную B%, потом третье (теперь символьную константу "КНИГА") – в третью переменную C; и все из того же блока DATA строки 40.

Для соблюдения порядка имеется специальный счетчик (указатель), который следит за порядком последовательности. Если вдруг окажется, что переменных в READ больше, чем значений в DATA, то ЭВМ выдаст сообщение о нехватке данных. При обратной ситуации – значений в DATA будет больше, чем переменных в READ, – "лишние" значения просто не считываются, но они могут быть использованы другим оператором READ, когда очередь дойдет до него. Счетчик будет хранить информацию, сколько значений уже использовано из списка данных оператора DATA, т.е. сколько значений уже считано (затребовано) оператором READ (или операторами, как у нас). Поэтому новым переменным будут присваиваться еще не использованные данные.

Прежде чем дальше рассматривать примеры с использованием операторов READ... DATA, посмотрим на правила, необходимые при работе с данными, найденными в строке DATA:

---

1. Данные могут располагаться в любом месте программы, но общепринятым является помещение их в конце программы.

2. Данные на единичной строке могут быть строкой, числом или их комбинацией.

3. Тип переменной, использованной в строке READ, должен соответствовать типам переменных в строке DATA, которые в нее считываются (иными словами, вы не можете читать строку в числовую переменную или наоборот).

4. Данные в строке DATA должны отделяться друг от друга запятой.

5. Не существует ограничений на общее количество данных, содержащихся в строке DATA, кроме тех, которые в определенной версии Бейсика накладывают на длину выражения.

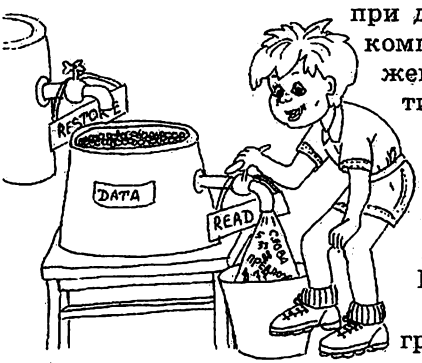
6. Если данные являются строковыми переменными, то их можно заключать в кавычки, а можно и не заключать.

7. Если данные являются числовыми переменными, то кавычки никогда не используются.

Такой строгий порядок считывания удобен особенно тогда, когда константами заполняется массив:

```
10 DIM A(12)
20 DATA 2,4,18,3,-2,11
30 DATA 8,-33,19,12,55,29
40 FOR I=1 TO 6
50 READ A(I)
60 NEXT
```

Как обычно, при выполнении программы компьютер последовательно движется со строки на строку. Однако при достижении строки DATA компьютер не выполняет заложенную в нем команду. Фактически он попросту игнорирует ее.



Компьютер обращает внимание на строку DATA только тогда, когда ему говорит об этом команда READ (это строка 50).

После запуска этой программы на выполнение строка 50 позволяет получить следующие значения одномерного массива A(1)=2, A(2)=4, A(3)=18, A(4)=3, A(5)=−2, A(6)=11. Затем счетчик оператора данных будет указывать, что использовано шесть чисел, а седьмое (число 8) стоит на очереди.

Надо сказать, что этот таинственный счетчик-указатель нам "не виден", программист не знает, что в нем находится. Однако воздействовать на него он может с помощью оператора RESTORE (вернуть).

В простейшем случае оператор RESTORE возвращает указатель к началу списка блока данных, т.е. на первую константу первой строки с командой DATA, при этом очередной оператор READ, следующий после оператора RESTORE, будет использовать значения из списка повторно.

Оператор RESTORE позволяет также устанавливать указатель на элемент списка, соответствующий любому промежуточному оператору DATA. Для этого в операторе RESTORE надо указать номер строки требуемого операто-

ра DATA. Если в строке с указанным номером не обнаружится такой оператор, то будет выбран ближайший оператор DATA, расположенный в программе после указанной строки:

```
10 READ A,B,C
20 PRINT A+B+C
30 DATA 2,8,9,10
40 RESTORE
50 READ X,Y,Z
60 PRINT X;Y;Z
70 END
RUN
19
2 8 9
```

```
10 DATA 1,2,3,4,
20 DATA 7,8,9,10
30 READ K,L
40 PRINT"ЗНАЧЕНИЕ K=";K;"L=";L
50 READ M,N
60 PRINT"ЗНАЧЕНИЕ M=";M;"N=";N
70 RESTORE
80 READ Q,P
90 PRINT"ЗНАЧЕНИЕ Q=";Q;"P=";P
100 RESTORE 20
110 READ K,L,M,N
120 PRINT"НОВЫЕ ЗНАЧЕНИЯ K=";K;"L=";L=";L;"M=";
M;"N=";N
130 END

RUN
ЗНАЧЕНИЕ K=1 L=2
ЗНАЧЕНИЕ M=3 N=4
ЗНАЧЕНИЕ Q=1 P=2
НОВЫЕ ЗНАЧЕНИЯ K=7 L=8 M=9 N=10
```

Заполнять константами можно и двухмерный массив, для чего нужно использовать вложенный цикл – один цикл вложен в другой:

```
100 DIM D%(2,4)
110 FOR I=1 TO 2
```

```
120 FOR K=1 TO 4
130 READ D%(I,K)
140 NEXT K
150 NEXT I
160 DATA 8,-33,19,14,-9,24,12,55
```

При этом элементы массива будут принимать следующие значения: D%(1,1)=8; D(1,2)=−33; D%(1,3)=19, D%(1,4)=14, D%(2,1)=−9, D%(2,2)=24, D%(2,3)=12, D%(2,4)=55, т.е. сначала заполняется первая строка целочисленного массива D% (внешний цикл – задается изменение строк), а потом и вторая строка (внутренний цикл – производится заполнение элементов в разных столбцах). Если же поменять местами (номерами) строки 120 и 110, а также 140 и 150, то заполнения будут вестись по столбцам. Так, например, в элементе D%(1,2) будет помещено число 19.

Если вы хотя бы в одном месте оператора DATA вместо запятой поставите точку, то программа воспринимает ее не как разделитель, а как десятичную точку, два соседних целых числа превратит в одно дробное, и компьютер выдаст вам сообщение о нехватке данных. В этом случае она введет не 8, а 7 чисел, и на последнее число не хватит данных.

Проверьте свои знания

1. В чем состоит удобство ввода значений переменных с помощью связки READ...DATA по сравнению с операторами LET и INPUT?
2. Расскажите правила, необходимые при работе с оператором DATA.
3. Объясните "механизм" заполнения константами одномерного массива.
4. Каково назначение оператора RESTORE?
5. Как происходит заполнение константами двухмерного массива?

Тренировка

1. В следующих программах допущены ошибки.
Найдите их:

а)
```
10 DIM B¤(3,1)
20 FOR J=1 TO 4
30 FOR K =1 TO 4
40 READ B¤(J,K)
50 NEXT J
60 NEXT K
```

б)
```
100 FOR K=1 TO 4
110 DATA 11,12.3,78
120 READ Q(K)
130 NEXT K
```

в)
```
10 DIM C%(18,3,9)
20 FOR I=3 TO 0 STEP -1
30 FOR L=9 TO 0 STEP -1
40 READ C%(I,L)
50 PRINT
60 DATA ШКОЛА,17,ДВЕРЬ, STOP, 4,73, 22,8, 3.2, -3,-11,
37,111,145,263,16,17,18
70 NEXT L,I
```

II. Запишите значения, которые примут переменные при выполнении команд:

а)
```
10 READ X,Y,Z
20 RESTORE
30 READ K,L
40 DATA 1.5,0.003E3,-9,2,4
```

б)
```
10 DATA 11,13,15
20 DATA 17,19,21
30 RESTORE 20
40 READ A,B
50 RESTORE
60 READ C,D
70 READ E
```

III. Составьте программу вычисления среднего геометрического двух чисел, содержащихся в DATA. Если хотя бы одно из чисел в DATA отрицательно, то задачу не решать.

IV. Вычислите плату за электроэнергию, потребляемую восемью электролампочками (мощность 100 Вт), телевизором (120 Вт) и холодильником (140 Вт). Входными данными являются: количество работы каждого прибора и стоимость за 1 кВт·ч.

Названия и мощности приборов должны храниться в блоке данных. Результаты вывести в виде таблицы. В конце привести общую сумму затрат.

§ 36. Примеры использования массивов

Иногда при выполнении программы заранее неизвестно, сколько раз должен повториться цикл, и вам нужно, чтобы компьютер продолжал исполнение цикла до тех пор,

пока не станет истинным определенное условие. Например, возможно, вы захотите повторить ввод строки до тех пор, пока пользователь не наберет слово "STOP", или хотите заставить компьютер просматривать массив до тех пор, пока он не найдет определенный элемент. Возможен случай, когда вы вводите элементы массива до тех пор, пока не встретится нужный элемент.

```
10 REM ЦИКЛ - ПОВТОРИТЬ ДО ТЕХ ПОР
20 DIM D¤(150)
30 LET I=0
40 LET I=I+1
50 PRINT"НОМЕР ЭЛЕМЕНТА ДАННЫХ";I
60 INPUT D¤(I)
70 IF D¤(I)<>"СТОП" THEN 40
```

Эта программа повторяет строку ввода до тех пор, пока пользователь не введет слово "СТОП". Понятно, элементы массива D¤(I) должны быть заранее определены.

```
10 REM НЕ ДЕЛАЙТЕ ТАКОЙ ВЫХОД ИЗ ЦИКЛА
20 DIM D¤(100)
30 FOR I=1 TO 100
40 READ D¤(I)
50 IF D¤(I)="STOP" THEN GOTO 70
60 NEXT I
65 END
70 REM ОСТАЛЬНАЯ ПРОГРАММА
80 ...

10 REM НЕ ВЫХОДИТЕ С ПОМОЩЬЮ GOTO ИЗ ЦИКЛА
20 DIM D¤(100)
```

Что ждет человека в ближайшие два десятка лет (предсказания японских ученых)

2003 г.

Водород будет производиться из воды.

2004 г.

Появятся роботы для тушения пожаров. Выйдут и начнут трудиться первые роботы-строители, а также роботы-шахтеры. Врачи научатся пересаживать больным легкие. Нефть будет добываться из морских глубин, превышающих километр.

```
30 FOR I=1 TO 100
40 INPUT D¤ (I)
50 IF D¤ (I)="КОНЕЦ" THEN GOTO 70
60 NEXT I
65 END
70 REM ОСТАЛЬНАЯ ПРОГРАММА
80 ...
```

В параграфе 33 мы показали такой же способ выхода из цикла. Рекомендуем: никогда не выходите из цикла FOR...NEXT с помощью оператора GOTO. Дело в том, что компьютер выполнит ваши команды, но где-то в его памяти останется частично заполненный счетчик цикла, и позже это может привести к ошибкам в программе.

Но как же выйти из цикла "безболезненно"?

Предлагаем два других приема:

```
10 REM ЦИКЛ ПОИСКА ЭЛЕМЕНТА В МАССИВЕ
20 LET R¤="РЕЦЕПТ"
30 FOR J=1 TO 5
40 READ D¤(J)
50 IF D¤(J)=R¤ THEN LET K=J
60 NEXT J
65 END
70 PRINT D¤(K)
80 DATA СТАТЬЯ,КНИГА,РЕЦЕПТ,ТЕЛЕФОН,СПРАВОЧНИК
```

```
10 REM АЛЬТЕРНАТИВНЫЙ ВАРИАНТ ЦИКЛА ПОИСКА
20 LET R¤="КНИГА"
30 FOR J=1 TO 5
40 READ D¤(J)
```

Что ждет человека в ближайшие два десятка лет (предсказания японских ученых)

2006 г.

Открытие вакцины против СПИДа.

2007 г.

Будут излечимы почти все формы рака, побежден инфаркт, изобретена косметика, благодаря которой кожа останется всегда молодой.

```
50 IF D¤(J)=R¤ THEN LET K=J:LET J=5
60 NEXT J
70 PRINT D¤(K)
80 DATA СТАТЬЯ,КНИГА,РЕЦЕПТ,ТЕЛЕФОН,СПРАВОЧНИК
```

Эти две программы иллюстрируют два разных способа прекращения циклов FOR...NEXT, когда условие становится истинным. На первом примере компьютер просматривает массив D¤ до тех пор, пока он не найдет элемент, который совпадает с R¤. Компьютер запоминает номер позиции этого элемента в массиве, присваивая другой переменной К текущее значение счетчика цикла. Затем он пробегает остальную часть цикла. Однако если важна скорость исполнения, можно прекратить присваивать переменной цикла ее конечное значение, как показано во втором примере.

Когда вы используете двухмерные массивы, вам, возможно, понадобится второй массив, который будет служить в качестве указателя для первого массива. Если у вас имеется группа связанных данных, например, место учебы, адрес, номера телефонов и даты рождения некоторых людей, то, используя двухмерный массив, вы помещаете в одну строку такого массива всю информацию, касающуюся одного человека, а в отдельный одномерный массив – фамилии этих людей. В таком случае можно было бы, как показано ниже, сначала осуществлять поиск конкретного человека в массе фамилий, а затем находить его (или ее) данные в другом массиве.

Что ждет человека в ближайшие два десятка лет (предсказания японских ученых)

2008 г.

Окончательно исчезнет озоновая дыра.

2012 г.

Массовое производство роботов-переводчиков, которые смогут переводить речь и рукопись со всеми тонкостями. Наступит конец аллергии. Ее окончательно победит медицина.

Фамилии людей должны быть расположены в том же порядке, что и строки с их данными в двухмерном массиве. Например, данные о человеке, имя которого хранится в N¤(3), должны быть в третьей строке массива B¤.

```
FOR J=1 TO 4
IF N¤(J)="ГОЛДА" THEN LET X=J
NEXT J
```

Этот цикл заставляет компьютер искать в массиве N фамилию "ГОЛДА". Когда компьютер находит нужную фамилию, он записывает в переменную X значение счетчика, т.е. индекса элемента, где хранится данная фамилия. Как было описано выше, этот прием позволяет избегать преждевременного выхода из цикла.

```
FOR K=1 TO 4
PRINT B¤ (X,K)
NEXT
```

После этого можно использовать число в переменной X, чтобы найти строку двухмерного массива B¤, где хранятся данные об этом человеке, и затем вывести все данные на экран. В заключение скажем, что такие связанные массивы называются **массивами с перекрестными ссылками**.

А теперь все изложенное воплотим в конкретной программе:

```
10 DIM B¤ (5,4)
15 DIM R¤ (5)
20 FOR X=1 TO 5
```

Что ждет человека в ближайшие два десятка лет (предсказания японских ученых)

2013 г.

Познание процесса отмирания клеток мозга у стареющих людей и его торможение.

2014 г.

Люди научатся точно предсказывать извержения вулканов и уменьшат их силу.

```
30 FOR K=1 TO 4
40 READ B¤(X,K)
50 NEXT K
60 NEXT X
70 DATA ТЕХН. КОЛЛЕДЖ,УНИВЕРСИТЕТСКАЯ, 33/2,
76-22-34,17-05-80
72 DATA МЕД.УЧИЛИЩЕ,АРТЕМА, 44/17,93-12-78,
5-02-80
74 DATA МЕД.ИНСТИТУТ,ПЕТРОВСКОГО, 123,
55-09-74,8-11-75
70 DATA ТЕХН.УНИВЕРСИТЕТ,МИРА, 31/8,96-17-12,
11-08-77
78 DATA ГОС.УНИВЕРСИТЕТ,ЩОРСА, 28/3,54-11-48,
24-07-78
80 INPUT"ВВЕДИТЕ ИНТЕРЕСУЮЩУЮ ФАМИЛИЮ";R¤
90 FOR J=1 TO 5
100 READ D¤ (J)
110 IF D¤(J)=R¤ THEN LET X=J
120 NEXT J
130 DATA ШЕВЧЕНКО,КАМЕНСКАЯ,ГОЛДА,ЗИМИН,
СЕРГЕЕВ
140 PRINT R¤ ;":"
150 FOR K=1 TO 4
160 PRINT B¤(X,K)
170 NEXT K
```

Представленная программа практически является основой программы **базы данных** (см. § 19). Разумеется, сами данные должны иметь большой объем информации, и было бы неплохо, если бы имеющаяся информация выводилась не только по фамилиям, но и по месту учебы, адресу, но-

Что ждет человека в ближайшие два десятка лет (предсказания японских ученых)

2015 г.

Торжественное открытие научно-исследовательской станции на Луне с постоянной и сменяемой командой. Развитие абсолютно безопасных и необычно мощных атомных электростанций.

меру телефона; могла объединяться, сравниваться, а пользователь этой базы данных имел возможность очень быстро получать нужные ему данные. Как говорят программисты, программа должна иметь "сервисную наполняемость", т.е. удобное для пользователя "меню". Меню должно представлять собой список различных действий, которые программа могла бы исполнить по вашему выбору.

Кроме того, такая программа должна быть "дружественной", т.е. она должна представлять пользователю развитые средства подсказки и не должна давать сбои или разрушаться, если пользователь допускает ошибки.

И вот мы вплотную подошли к еще более глубокому понятию – к базе знаний.

База знаний – это базы данных, в которых можно не только накапливать и искать имеющуюся в памяти ЭВМ информацию, но и получать от машин логически осмысленные заключения и рекомендации.

Такую организованную совокупность знаний с использованием ЭВМ иногда называют **экспертной системой**.

Основной функцией такой системы является хранение в машине не только фактического материала, но и правил вывода следствий из имеющихся данных, что позволяло бы устанавливать факты, которых в базе не было. Но для этого программист должен предварительно сообщить машине имеющуюся информацию и необходимые правила вывода умозаключений.

Такие программы, которые проводили бы осмысленные рассуждения, должны опираться на законы логики, к изучению основ которых мы приступим в следующей главе.

Что ждет человека в ближайшие два десятка лет (предсказания японских ученых)

2016 г.

Рождение космических кораблей на атомной тяге. Космическое (читайте: туристическое) бюро путешествий будет предлагать желающим взглянуть на Землю с околоземной орбиты.

Проверьте свои знания

1. Почему не рекомендуют выходить из цикла FOR...NEXT с помощью оператора GOTO?
2. Приведите приемы поиска элемента в массиве.
3. Объясните, как используют массивы с перекрестными ссылками.
4. Расскажите, каким требованиям должны удовлетворять программы базы данных.
5. Что такое база знаний?

Тренировка

I. Напишите базу данных "погода в вашей местности". При выводе информации постарайтесь учесть следующие направления:

1) дата; 2) температура; 3) давление; 4) влажность; 5) наличие облачности, ветра; 6) осадки и их количество.

II. Составьте программу данных газетных и журнальных статей. Для начала пусть она имеет пять разделов:

1) история; 2) политика; 3) рецепты блюд; 4) компьютер; 5) разное.

Предусмотрите вывод информации по следующим параметрам:

а) источник (название журнала или газеты); б) дата выхода; в) краткое содержание.

III. Составьте программу базы данных одного из вышеуказанных разделов (см. задачу II из этого упражнения), например, рецепты блюд, предусмотрев следующие главы:

Что ждет человека в ближайшие два десятка лет (предсказания японских ученых)

2017 г.

Появится первый искусственный мозг.

2018 г.

Полет на Марс для сооружения постоянного марсианского космодрома. Появятся солнечные электростанции, передающие энергию с орбиты на Землю.

1) первые блюда; 2) вторые блюда; 3) салаты; 4) напитки; 5) мучные блюда.

Информацию выводить в такой последовательности:

а) содержание изделия; б) энергетическая ценность (витамины); в) сам рецепт.

IV. Напишите базу данных по любому интересуемому вас направлению: каталог вашей библиотеки, музыкальная фонотека, ваша коллекция, природоведческая база данных, база данных продуктов питания, ваше самочувствие, домашние расходы, домашнее консервирование и т.д.

§ 37. Подпрограммы. Операторы GOSUB... RETURN (Иди к подпрограмме... возврат)

Может случиться, что в одной и той же программе надо выполнять несколько раз одну и ту же последовательность инструкций (операторов). Чтобы избежать повторений, указанную группу операторов можно записать в программе один раз и обращаться к ней, когда в этом возникает необходимость.

Обособленную группу операторов, которую можно выполнять многократно, обращаясь к ней из различных мест программы, называют подпрограммой.

Что ждет человека в ближайшие два десятка лет (предсказания японских ученых)

2019 г.

Искусственные клетки и искусственная жизнь в пробирке. Изобретение искусственных глаз для слепых и слабовидящих.

2020 г.

Выведение "формулы бессмертия"

Подпрограмма располагается в произвольном месте общей программы, а ее строки нумеруются обычным порядком. Существенное значение при этом имеет номер первой строки подпрограммы, так как он используется при обращении к подпрограмме.

Обращение к подпрограмме осуществляется с помощью специального оператора GOSUB (сокращение англ. слова GOTO и SUBROUTINE – перейти к подпрограмме). После служебного слова GOSUB указывается натуральное число – номер строки, с которой начинается подпрограмма (то есть метка), например:

```
30 GOSUB 1000
40 REM ПРОДОЛЖЕНИЕ ПРОГРАММЫ
```

Вы помните оператор GOTO? С помощью GOTO мы заставляли компьютер переходить на другую строку программы. С GOSUB происходит то же самое, но затем с помощью оператора RETURN (возврат) мы автоматически возвращаемся в основную программу. А точнее, на строку, которая следует за строкой GOSUB. Иными словами, оператор, к примеру, GOSUB 1000, не только передает управление на указанную в нем строку (строка 1000), но при этом еще и запоминает точку вызова, в которую необходимо вернуться после завершения работы вызванной подпрограммы.

После обращения к подпрограмме выполняются все предусмотренные в ней действия, а когда очередь доходит до оператора RETURN, он как раз и возвращает управление в точку вызова, а точнее – оператору, следующему за GOSUB (у нас таковым будет оператор REM в строке 40).

Представим себе, что у нас есть программа и совокупность команд (операторов), которые несколько раз повторяются. Для того чтобы не писать их в программе несколько раз, мы напишем их всего один раз (обычно это делают в конце программы) и посредством оператора GOSUB будем выполнять их, когда необходимо. Компьютер будет автоматически переходить на эту часть программы.

Рассмотрим пример.

Вам необходимо запрограммировать детскую песенку:

```
10 PRINT "ЖИЛИ У БАБУСИ ДВА ВЕСЕЛЫХ ГУСЯ:"
```

```
20 PRINT "ОДИН СЕРЫЙ, ДРУГОЙ БЕЛЫЙ,"
30 PRINT "ДВА ВЕСЕЛЫХ ГУСЯ."
40 PRINT "ВЫТЯНУЛИ ШЕИ, У КОГО ДЛИННЕЕ-"
50 PRINT "ОДИН СЕРЫЙ, ДРУГОЙ БЕЛЫЙ,"
60 PRINT "У КОГО ДЛИННЕЕ."
70 PRINT "МЫЛИ ГУСИ ЛАПКИ В ЛУЖЕ У КАНАВКИ-"
80 PRINT "ОДИН СЕРЫЙ, ДРУГОЙ БЕЛЫЙ,"
90 PRINT "СПРЯТАЛИСЬ В КАНАВКЕ."
100 PRINT "ВОТ КРИЧИТ БАБУСЯ: ОЙ, ПРОПАЛИ ГУСИ-"
110 PRINT "ОДИН СЕРЫЙ, ДРУГОЙ БЕЛЫЙ"
120 PRINT "ГУСИ МОИ, ГУСИ!"
130 PRINT "ВЫХОДИЛИ ГУСИ, КЛАНЯЛИСЬ БАБУСЕ-"
140 PRINT "ОДИН СЕРЫЙ, ДРУГОЙ БЕЛЫЙ,"
150 PRINT "КЛАНЯЛИСЬ БАБУСЕ."
```

Эта программа просто умоляет о подпрограмме, ведь в таком виде она требует большого количества работы. Теперь давайте посмотрим, как будет выглядеть то же самое, но написанное при помощи подпрограммы:

```
10 PRINT"ЖИЛИ У БАБУСИ ДВА ВЕСЕЛЫХ ГУСЯ:"
20 GOSUB 200
30 PRINT"ДВА ВЕСЕЛЫХ ГУСЯ. ВЫТЯНУЛИ ШЕИ У
КОГО ДЛИННЕЕ-"
50 GOSUB 200
60 PRINT"У КОГО ДЛИННЕЕ. МЫЛИ ГУСИ ЛАПКИ В
ЛУЖЕ У КАНАВКИ-"
60 GOSUB 200
90 PRINT"СПРЯТАЛИСЬ В КАНАВКЕ. ВОТ КРИЧИТ БА-
БУСЯ: ОЙ, ПРОПАЛИ ГУСИ-"
110 GOSUB 200
120 PRINT"ГУСИ МОИ, ГУСИ! ВЫХОДИЛИ ГУСИ,
КЛАНЯЛИСЬ БАБУСЕ-!"
140 GOSUB 200
150 PRINT"КЛАНЯЛИСЬ БАБУСЕ."
160 END
200 PRINT"ОДИН СЕРЫЙ, ДРУГОЙ БЕЛЫЙ"
210 RETURN
```

А теперь, прежде чем перейти к некоторым усложненным вариантам использования подпрограмм, нам необходимо изучить некоторые фундаментальные правила, каса-

ющиеся их. Работая с подпрограммами, постарайтесь придерживаться следующего:

1. Программа может содержать любое количество подпрограмм.

2. Подпрограмма может содержать любое количество строк.

3. Подпрограмма может располагаться в любом месте программы, но обычно ее помещают в конце – перед строкой данных (DATA), если таковая имеется.

4. Подпрограмма может быть выполнена только при помощи команды GOSUB, что означает "Иди к подпрограмме", так же, как и строка DATA может быть задействована только командой READ. Вход в подпрограмму без оператора GOSUB ЗАПРЕЩЕН.

5. Конец подпрограммы всегда отмечается командой RETURN (возврат). Этот оператор означает: "Возвращайся к оператору, следующему сразу за оператором GOSUB, и оттуда двигайся дальше".

6. Основная программа чаще всего отделяется от подпрограммы оператором STOP или END, что означает конец основной программы.

7. Подпрограммы могут быть вызваны из других подпрограмм.

Поскольку подпрограммы легко могут быть связаны собой, они часто рассматриваются в качестве строительных блоков (модулей) подпрограммы. При создании большой программы программист как бы посвящает отдельные секции или блоки инструкций определенным заданиям внутри программы. Работая с несколькими такими модулями, или подпрограммами, легче выполнить большую программистскую работу. Вспомните, что такой подход в программировании мы назвали структурным (см. § 32). Подпрограммы очень полезны и с другой точки зрения.

Они облегчают программисту поиск ошибок. Обычно "настройку" и исправление ошибок и помех программисты называют **отладкой**. Поэтому использование подпрограмм как раз и облегчает отладку всей программы, т.к. работа каждой подпрограммы может быть проверена по отдельности. И помехи, происходящие в каждой, могут быть легко изолированы и исправлены. Другими словами, проще определить местонахождение ошибки, которая спрятана в маленькой подпрограмме, чем ошибки, спрятавшиеся в бесконечных закоулках и тупиках большой программы.

Кроме того, многие подпрограммы имеют дополнительную ценность, поскольку ими может воспользоваться не только тот, кто написал подпрограмму, но и другие лица.

Большинство программистов коллекционируют, собирают часто используемые подпрограммы, которые существенно облегчают программирование реальных задач. Это универсальные подпрограммы вывода сообщений, различных проверок исполнения; подпрограммы, заставляющие ждать компьютер до нажатия любой клавиши; универсальные подпрограммы "меню", различные графические подпрограммы (рисования линий, графиков функций, гистограмм, различных фигур) и т.д.

Чтобы подпрограмма при обращении к ней выполнялась каждый раз с новыми данными, ее нужно составлять в общем виде, а исходные данные для ее работы нужно передавать в переменные подпрограммы перед обращением к ней.

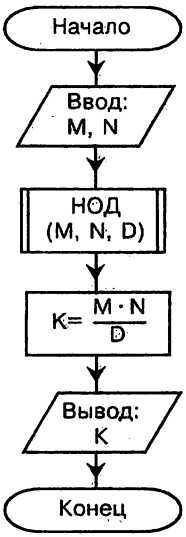
Например.

Вычислить наименьшее общее кратное (НОК) двух натуральных чисел M и N с использованием подпрограммы.

Решение

Для нахождения НОК двух натуральных чисел M и N используем формулу $K = \frac{M \cdot N}{D}$, где D – наибольший общий делитель (НОД) тех же двух натуральных чисел M и N. Поэтому в качестве подпрограммы используем фрагмент программы вычисления НОД двух чисел (см. § 16).

Составим блок-схему решения этой задачи, напомнив читателю, что на таких блок-схемах стандартная подпрограмма обозначается в виде прямоугольника с боковыми линиями:




```
10 REM ПРОГРАММА НАХОЖДЕНИЯ
НАИМЕНЬШЕГО ОБЩЕГО КРАТНОГО
ДВУХ ЧИСЕЛ
20 INPUT"ВВЕДИТЕ НАТУРАЛЬНЫЕ
ЧИСЛА M,N";M,N

30 GOSUB 100


40 LET K=M*N/D

50 PRINT"НАИМЕНЬШЕЕ ОБЩЕЕ
КРАТНОЕ ЧИСЕЛ";M;"И";N;"РАВНО";K

60 END

100 LET X=M
110 LET Y=N
120 IF X=Y THEN LET D=X ELSE
GOTO 140
130 GOTO 170
140 IF X>Y THEN LET X=X-Y ELSE
GOTO 150
145 GOTO 120
150 LET Y=Y-X
160 GOTO 120
170 RETURN
```

Обращения к программам могут вкладываться одно в другое; при этом вложенность программ несколько отличается от вложенности циклов или вложенности скобок в выражении. В случае подпрограмм вкладываются не сами подпрограммы, а вызывающие их операторы GOSUB. Таким образом, первая подпрограмма может вызывать вторую, вторая – третью и т.д.

Например:

```
1000 GOSUB 2000
1010 END

2000 'ПОДПРОГРАММА ПЕЧАТИ
ЦИТАТЫ
2010 GOSUB 3000
2020 PRINT"ИСПОЛЬЗУЯ ПРИНЦИП
ВЛОЖЕНИЙ, МОЖНО НЕ ТОЛЬКО
РАЗБИВАТЬ СЛОЖНУЮ ";
2030 PRINT"ПРОГРАММУ НА
ОТДЕЛЬНЫЕ МОДУЛИ И ДЛЯ
КАЖДОГО ИЗ НИХ ПИСАТЬ СВОЮ ";
2040 PRINT"ПОДПРОГРАММУ, НО И
РАЗДЕЛЯТЬ НА МОДУЛИ ЛЮБЫЕ
ПОДПРОГРАММЫ. "
2050 PRINT"ЧЕМ МЕНЬШЕ БУДЕТ
КАЖДЫЙ ВЫДЕЛЕННЫЙ ПРОГРАМ
МНЫЙ МОДУЛЬ,"
2060 PRINT"ТЕМ ЛЕГЧЕ БУДЕТ ЕГО
ПРОГРАММИРОВАТЬ И ОТЛАЖИВАТЬ!"
2070 GOSUB 3000
2080 RETURN

3000 REM ПОДПРОГРАММА ПЕЧАТИ
ЗВЕЗДОЧЕК
3010 FOR Z%=1 TO 64
3020 PRINT "*";
3030 NEXT Z%
3040 RETURN
```

Основная программа в этом примере состоит лишь из оператора GOSUB (строка 1000) и оператора END (строка 1010). Первая подпрограмма (строки 2000-2080) осуществляет печать текста цитаты, окаймленного звездочками. Для печати звездочек первая подпрограмма вызывает вторую (строки 3000-3040), причем в двух местах.

ЧТО МОЖЕТ КОМПЬЮТЕР

Как решить до конца неопределенную задачу

Сколько будет: кажется, два умножить на примерно два? Ответить на этот вопрос однозначно ни один компьютер не сможет. А ведь в жизни вопросы чаще всего ставятся именно в такой форме. Если добавить, что при формулировке проблем часто используются такие неопределенные термины, как "похожий", "быстро", "примерно", то становится понятно, насколько сложна работа программистов-прикладников.

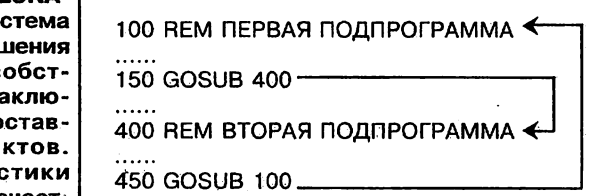
Чтобы облегчить поиск решений для задач, начальные условия которых определены не полностью или расплывчато, а также

упростить формализацию неопределенных систем, японская фирма Omron спроектировала программно-аппаратную компьютерную систему для работы с нечеткой логикой LUNA-Fuzzy RON. Система принимает решения на основе собственных умозаключений и сопоставления фактов. Никакой мистики здесь нет – качество выводов определяется умелым выбором для решения той или иной задачи одной из пяти заложенных в компьютер математических систем, которые используют минимальную логику, весовые вычисления и еще три детерминированных термина.

Используя простые правила "если – то", другие логические операции и операции отношения, вы можете сформулировать до 300 правил, описывающих проблему в привычных вам терминах. А уж найти единственное в этих условиях решение поможет LUNA-Fuzzy.

Вид вложенности, когда подпрограмма обращается к самой себе, называется рекурсией.

Понятие рекурсии охватывает также и ситуацию, при которой первая подпрограмма вызывает вторую, а та, в свою очередь, вызывает первую.



Надо заметить, что при вхождении подпрограммы одной в другую, оператор GOSUB нельзя применять для входа внутрь цикла. Если подпрограмма имеет в своем составе цикл, то он должен полностью заканчиваться в этой подпрограмме (т.е. оператор NEXT цикла должен стоять раньше оператора RETURN соответствующей подпрограммы).

Примеры рекурсии "из жизни": конфета "А ну-ка отними!" (на фантике нарисована девочка, держащая конфету, на которой нарисована девочка, держащая конфету... и т.д.), "У попа была собака...", зеркала, расположенные друг против друга.

Приведем пример рекурсивной программы:

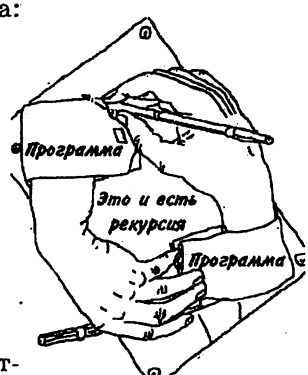
```
10 REM ПРИМЕР РЕКУРСИИ
20 K=1
30 GOSUB 100
40 END
```

```
100 REM РЕКУРСИВНАЯ ПОДПРОГРАММА
110 IF K=3 THEN 140
120 K=K+1
130 GOSUB 100
140 K=K-1
150 PRINT K
160 RETURN
200 REM ...
```

При первом обращении к рекурсивной подпрограмме из строки 30 меткой возврата будет адрес строки 40, при втором обращении – из строки 130 – адрес следующей строки (140) и т.д.

Проследим за метками возврата:

K	МЕТКА ВОЗВРАТА
1	40
2	140
3	140
2	140
1	140
0	40



Адреса возврата по оператору RETURN: берутся номера строк (метки) в обратном порядке (в таблице они выделены рамками). В последнюю очередь будет выполнен возврат к строке 40 и останов по оператору END.

Проверьте свои знания

1. Что называют подпрограммой?
2. Для чего используются и как взаимосвязаны операторы обращения к подпрограмме GOSUB и возврата RETURN?
3. Чем отличается оператор безусловного перехода GOTO от оператора вызова подпрограммы GOSUB?
4. Приведите правила работы с подпрограммами.
5. Что такое отладка программы?
6. В чем состоят преимущества в составлении программ с использованием подпрограмм?

7. Назовите особенности вложенных подпрограмм.

8. Что такое рекурсия?

Тренировка

I. Используя подпрограмму, уменьшите объем следующей программы:

```
10 INPUT"ВВЕДИТЕ АРГУМЕНТ ФУНКЦИИ";X
20 IF X<=0 THEN PRINT "ПОВТОРИТЕ ВВОД" ELSE 30
25 GOTO 10
30 LET Y1=SQR(X)
40 INPUT "ВВЕДИТЕ АРГУМЕНТ ФУНКЦИИ"; X
50 IF X <=0 THEN PRINT "ПОВТОРИТЕ ВВОД" ELSE 60
55 GOTO 40
60 LET Y2=SIN(X)
70 INPUT"ВВЕДИТЕ АРГУМЕНТ ФУНКЦИИ";X
80 IF X<=0 THEN PRINT"ПОВТОРИТЕ ВВОД" ELSE 90
90 LET Y3=COS(X)
100 PRINT Y1,Y2,Y3
110 END
```

II. В следующей программе исправьте ошибки:

а)
```
10 FOR K=1 TO 5
20 GOSUB 50
30 NEXT K
40 PRINT "ЭТА ПРОГРАММА ВЫЗЫВАЕТСЯ";K-1;"РАЗ"

50 REM НАЧАЛО ПОДПРОГРАММЫ
60 PRINT "ПРИВЕТ ИЗ ПОДПРОГРАММЫ!"
70 RETURN
```

б)
```
10 REM ОСНОВНАЯ ПРОГРАММА
20 GOSUB 70
30 END

50 REM НАЧАЛО ПОДПРОГРАММЫ
60 FOR I=1 TO 10
70 PRINT I
80 NEXT I
90 RETURN
```

III. Поясните работу программы с рекурсией:

```
10 REM ОСНОВНАЯ ПРОГРАММА
20 S=0
25 PRINT "ПРОХОД ОСНОВНОЙ ПРОГРАММЫ";S;"РАЗ"
30 IF S>=3 THEN 70
40 GOSUB 100
```

```
60 RETURN
70 END
100 REM НАЧАЛО ПОДПРОГРАММЫ
110 PRINT"ПРИВЕТ ИЗ ПЕРВОЙ ПРОГРАММЫ";S;"РАЗ"
120 S=S+1
130 GOSUB 25
140 RETURN
```

IV. Составьте блок-схему и программу вычисления площади четырехугольника ABCD по его сторонам и диагонали: AB=X; BC=4; CD=Z; AD=T; AC=D.
Для решения используйте подпрограмму нахождения площади треугольника по формуле Герона (см. § 15).

V. Составьте блок-схему и программу решения уравнения вида $y=a^x$, где x – любое целое число. Используйте в качестве подпрограммы программу вычисления степени с натуральным показателем (см. § 16, тренировка V). Учтите, что:

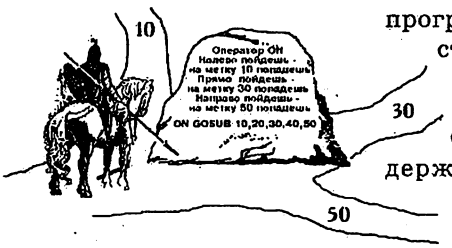
$$a^x = \begin{cases} 1, \text{ если } x=0 \\ a^x, \text{ если } x>0 \\ \left(\frac{1}{a}\right)^{-x}, \text{ если } x<0 \end{cases}$$

VI. Составьте блок-схему и программу решения биквадратного уравнения вида $az^4+bz^2+c=0$ в области действительных корней ($a\neq0$). В качестве подпрограммы используйте фрагмент программы решения квадратного уравнения (см.§ 16 и § 30).

VII. Игра Баше. Игра состоит в том, что играющие берут по очереди от 1 до P предметов из заданной совокупности N предметов. Проигрывает тот, кто берет последний предмет. Значения N и P выбирают до начала игры. Разработайте выигрышную стратегию игры Баше и по ней составьте блок-схему и программу.

§ 38. Операторы ON...GOTO, ON...GOSUB. Функция пользователя

Мы с вами изучили операторы, передающие управление в другое место программы: GOTO и GOSUB. Но часто возникает необходимость передавать управление не одной и той же программной строке или подпрограмме, а разным – в зависимости от сложившейся в ходе выполнения



программы ситуации. Средствами таких передач управления являются операторы ON-GOTO и ON-GOSUB, которые содержат выражение (указатель) и список номеров строк.

Оператор выбора ON (от англ. – "по") имеет следующий вид:

$$\text{ON ВЫРАЖЕНИЕ} \left\{ \begin{matrix} \text{GOTO} \\ \\ \text{GOSUB} \end{matrix} \right\} \text{СПИСОК НОМЕРОВ}$$

Значение выражения служит индексом в списке номеров: значение, равное 1, означает, что управление передается строке с номером, стоящим первым в списке номеров; значение, равное 2, означает, что управление передается строке с номером, стоящим вторым в этом списке, и т.д.

Например:

```
10 INPUT"ВВЕДИТЕ НОМЕР ХИМИЧЕСКОГО ЭЛЕМЕНТА
(ОТ 1 ДО 3):";N
20 ON  N GOTO 100,200,300
30 PRINT"НОМЕР НЕ ОПРЕДЕЛЕН"
40 GOTO 10
100 PRINT"ВОДОРОД. АТОМНЫЙ ВЕС 1.00797"
110 GOTO 10
200 PRINT"ГЕЛИЙ. АТОМНЫЙ ВЕС 4.0026"
210 GOTO 10
300 PRINT"ЛИТИЙ. АТОМНЫЙ ВЕС 6.939"
310 GOTO 10
```

При выполнении этих операторов сначала вычисляется значение арифметического выражения. Дробная часть его отбрасывается, а затем происходит переход к строке из списка номеров.

Если выполняется команда ON...GOSUB, то она равнозначна команде перехода к подпрограмме GOSUB, номер строки, где номер строки определяется выражением, и по

команде RETURN управление вычислениями перейдет к оператору, стоящему после команды ON... GOSUB.

Если текущее значение переменной выражения больше количества элементов в списке номеров строк или значение указателя равно нулю, то управление передается оператору, следующему за данной командой выбора. Отрицательное значение переменной выражения приводит к программной ошибке, и вычисления прекращаются.

В списке номера строк могут повторяться.

Пример. Изменим программу решения квадратного уравнения (см. § 16 и § 30) с учетом нахождения мнимых корней:

```
10 REM РЕШЕНИЕ УРАВНЕНИЯ ВИДА AX^2+BX+C=0
20 INPUT "ВВЕДИТЕ КОЭФФИЦИЕНТЫ A,B,C";A,B,C
30 IF A=0 THEN Q=1 ELSE 50
40 GOTO 80
50 LET D=B^2-4*A*C
60 IF D=0 THEN Q=2
70 IF D>0 THEN Q=3 ELSE Q=4
80 ON Q GOSUB 100,200,300,400
90 END

100 PRINT"УРАВНЕНИЕ ИМЕЕТ ОДНО РЕШЕНИЕ: X=";
-C/B
110 RETURN

200 PRINT"КОРНИ УРАВНЕНИЯ ДЕЙСТВИТЕЛЬНЫЕ И
РАВНЫЕ: X1=X2=";-B/(2*A)
210 RETURN

300 PRINT"ОБА КОРНЯ УРАВНЕНИЯ ДЕЙСТВИТЕЛЬНЫЕ:
X1=";(-B+SQR(D))/(2*A);"X2=";(-B-SQR(D))/(2*A)
310 RETURN

400 PRINT"УРАВНЕНИЕ ИМЕЕТ МНИМЫЕ КОРНИ:"
410 PRINT"X1=";-B/(2*A);"+";SQR(-D)/(2*A));"i"
420 PRINT"X2=";-B/(2*A);"-";SQR(-D)/(2*A));"i"
430 RETURN
```

Рассмотренные операторы часто называют "переключателями". Действительно, их работа похожа на работу многопозиционного переключателя, коммутирующего

прохождение выполнения программы по одному из возможных направлений.

Такую блок-схему оператора ON можно отнести к схематической структуре **"ПЕРЕКЛЮЧЕНИЕ"** (рис. 38).

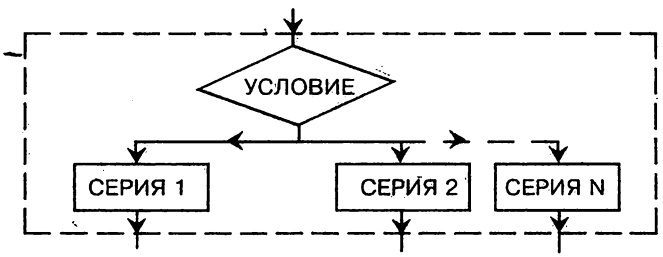


Рис.38. Структура "ПЕРЕКЛЮЧЕНИЕ"

А теперь поговорим о **функции пользователя.**

Помимо стандартных числовых (см. § 25) или символьных функций (см. § 39) пользователь может определить и свои собственные. Иногда при решении задач возникает необходимость вычисления одного и того же выражения при различных значениях величин, входящих в это выражение. Вот тогда и применяют **функции, определяемые пользователем.**

Например, нам необходимо часто вычислять функцию вида $F(x)=ax^2+ bx+c$.

Для этого мы должны определить соответствующую функцию командой:

```
DEF FNF(X)=A*X^2+B*X+C
```

Оператор определения функции начинается с командного слова DEF, после которого должно быть указано имя функции, начинающееся обязательно с букв FN (F – функция, N – нестандартная). Оставшаяся часть этого имени может представлять собой любое допустимое имя переменной, выбираемое пользователем. Вслед за именем функции

идет список параметров, заключенный в скобки. Параметры – это имена фиктивных переменных, резервирующих место в памяти для фактических значений, которые должны быть определены к моменту выполнения функции. Список этих переменных разделяется запятыми (они вовсе могут отсутствовать). В оставшейся части оператора записывается выражение, определяющее, какие действия производит данная функция.

Выражение может быть как арифметическим, так и символьным, оно может содержать переменные, не входящие в список параметров. При обращении к функции будут использованы текущие значения переменных. Следовательно, нужно следить за тем, чтобы к моменту обращения к функции эти значения были заданы.

В качестве операндов выражения могут использоваться константы, элементы массивов, стандартные функции.

Приведем примеры определения функции:

```
DEF FN KT(X)=COS(X)/SIN(X)
DEF FN SG(X)=SIN(X*Pi/180)
DEF FN P(A,B)=A/B*100
DEF FN T$="УЧЕНИК"+F$
```

Подчеркнем еще раз, что имена переменных в списке параметров являются формальными, т.е. используются для задания выражений, стоящих в правой части, и не имеют отношения к переменным в основной программе.

Обращение к функции пользователя происходит по имени с добавлением впереди приставки FN.

Примеры обращения к функции пользователя:

```
FN KT(X)          FN SG(A)          FN P(A,B)
FN T$("ШЕВЧЕНКО")
```

Обращение к функции пользователя можно записать в арифметическом выражении в списке вывода оператора PRINT и т.д., вообще везде, где требуются значения этой функции (т.е. аналогично использованию стандартной функции).

При обращении к функции пользователя необходимо в список параметров подставить имена реальных (фактических) переменных из программы.

Известно ли вам, что...

НУЖНО ДОКАЗЫВАТЬ ИЛИ ТЕСТИРОВАТЬ

Задача программиста не просто написать программу, но доказать, что написанная программа отвечает заданным функциям. Разрабатывая программы и их доказательства параллельно, студенты должны активно пользоваться математической логикой и исчислением предикатов (см.§46). Вводный курс программирования должен в значительной степени быть формализованным курсом. Изучаемый в этом курсе язык программирования вообще не должен быть реализован на компьютере, чтобы заранее предотвратить попытки студентов тестировать (а не доказывать) свои программы.

Такой подход к обучению будущих программистов-профессионалов предложил известный специалист в области программирования профессор Эдсгер Дийкстра в ходе дискуссии на страницах печати.

Например:

```
10 REM СИНУС С АРГУМЕНТОМ
В ГРАДУСАХ
20 DEF FN SG(X)=SIN(X*Pi/180)
40 INPUT "ВВЕДИТЕ ЗНАЧЕНИЕ
УГЛА В ГРАДУСАХ"; L
50 PRINT "СИНУС";L;"ГРАДУСОВ
РАВЕН";FN SG(L)
60 END

RUN
ВВЕДИТЕ ЗНАЧЕНИЕ УГЛА В
ГРАДУСАХ? 30(ВК)
СИНУС 30 ГРАДУСОВ РАВЕН 0.5
```

Еще пример. Задача по физике:

С каким ускорением скользит по наклонной эстакаде ящик, если высота эстакады H=8 м, а ее длина L=10 м? Коэффициент трения скольжения ящика по эстакаде M=0,5.

```
10 REM ТЕЛО НА НАКЛОННОЙ
ПЛОСКОСТИ
20 DEF FN S(H,L)=SIN(H/L)
30 LET G=9.8
40 INPUT "ВВЕДИТЕ ВЫСОТУ (H),
ДЛИНУ (L) ЭСТАКАДЫ,
КОЭФФИЦИЕНТ ТРЕНИЯ (M)";H,L,M
50 A=G*(FN S(H,L)-M*SQR(1-FN
S(H,L)^2))
60 PRINT"УСКОРЕНИЕ РАВНО";A;
"M/КГ"
70 END
```

В арифметическом выражении правой части оператора, стоящего в строке 50, имеется двойное обращение к функции пользователя FN S.

Проверьте свои знания

1. Почему оператор ON... называют командой выбора?

2. Поясните работу оператора ON совместно с оператором GOTO и GOSUB.

3. Как будет выполняться программа, если текущее значение переменной оператора ON больше количества элементов в списке номеров строк? Когда текущее значение равно нулю? Когда оно отрицательное?

4. Как задается и как вызывается функция пользователя?

5. Поясните смысл формальных и фактических параметров. Какое между ними существует соответствие?

6. Какое значение переменной, стоящей в правой части функции пользователя, берется при обращении к функции, если имя этой переменной не стоит в списке формальных параметров?

Тренировка

I. Чему будет равно значение X после выполнения следующей серии команд:

```
10 INPUT"ВВЕДИТЕ ЗНАЧЕНИЕ A";A
20 X=2*A-4
30 ON X GOTO 40,50,60,70
40 X=2
45 GOTO 80
50 X=2
60 X=X+1
70 X=X*X+4
80 END
```

при а) A=1; б) A=2; в) A=3; г) A=4?

II. Используя оператор ON... GOSUB, составьте меню для программ к тренировкам § 36.

III. Используя функцию пользователя, напишите программу нахождения гипотенузы (C) треугольника по его катетам (A и B).

IV. Применяя функции пользователя и оператор ON...GOSUB, напишите программу перевода температуры из одной шкалы измерения в другую.

Используйте формулы перевода:

F=9/5*C+32
K=273+C
C=(F-32)*5/9
C=K-273
K=273+(F-32)*5/9
F=9/5*(K-273)+32

где C – шкала Цельсия;
K – шкала Кельвина;
F – шкала Фаренгейта.

V. Составьте программу проверки: принадлежит ли точка внутренней области круга с центром в точке X0, Y0 и радиусом R? Для этого опишите функцию пользователя для вычисления расстояния от точки до центра, в качестве формальных параметров возьмите координаты этой точки (X,Y).

§ 39. Символьные функции

Часто пользователю приходится не только делать вычисления, но и обрабатывать текстовую информацию; находить нужное слово (фамилию, какое-то наименование и т.д.), либо какую-то часть его; размещать или передвигать по экрану дисплея текст и т.п.

Для выполнения всех этих действий предназначены функции обработки текстов. Напомним, что под текстовой (символьной или литерной) строкой понимают символьное выражение, символьную переменную или символьную константу (см. § 24). Все строковые функции, как и строковые переменные (кроме LEN), в конце своего имени имеют отличительный знак "Ꞩ" (или знак доллара "$").

Вообще, для работы над символьными данными существуют две операции:

1) сцепление (соединение) слов;

2) выделение в слове его части (подслов).

Первую операцию мы с вами уже разбирали (см. § 24), ее называют **конкатенация**. Это когда к первому слову справа приписывают второе, соединение слов обозначают знаком "+" (как сложение чисел). Например:

```
10 PRINT"САМ"+"О"+"СВАЛ"

RUN
САМОСВАЛ
```

Для работы со словами часто необходимо знать их длину. Для обозначения длины в языке Бейсик используют оператор LEN (LENGTH – длина). Например:

```
5 Z¤="ЗДРАВСТВУЙТЕ, Я ВАША ТЕТЯ!"
10 PRINT LEN(Z¤)
RUN
 26
```

Проверьте!

Напомним, что символьные константы заключаются в кавычки. Кавычки в этом операторе не учитывают при вычислении длины. Для чего же они нужны? А для того, чтобы ЭВМ отличала, скажем, слово 732 от числа 732, букву N от переменной N, слово M(5,4) от элемента массива M(5,4). Причем пробел внутри строки является полноправным символом.

Еще пример:

```
10' ФУНКЦИЯ LEN
20 T¤="УМ ХОРОШО"
30 L=LEN(T¤)
40 L1=LEN(T¤+" 2 ЛУЧШЕ")
50 L2=LEN("")
60 PRINT L;L1;L2;LEN("СМЕЛЫМ СУДЬБА
ПОМОГАЕТ")

RUN
 9  17  0  22
```

Используя этот оператор, можно написать программу, которая будет искать самое длинное из введенных слов:

```
10 LET A¤=" "
20 PRINT "ПОЖАЛУЙСТА,СЛОВА"
30 FOR K=1 TO 5
40 INPUT B¤
50 IF LEN(B¤)>LEN(A¤) THEN LET A¤=B¤
60 NEXT K
70 PRINT"САМОЕ ДЛИННОЕ СЛОВО:"
80 PRINT A¤
```

Компьютер при каждом повторении цикла сравнивает длину слова A$. Если слово B$ длиннее A$, компьютер заменяет A$ на B$ и затем сравнивает следующее слово. В конце цикла переменяя A$ содержит самое длинное из введенных слов.

❐

Иногда приходится работать с отдельными символами или группой символов внутри текстовой строки. Для этого служит функция подстроки – MID (от англ. midle – середина):

MID$ (строка, с какого, сколько)
MID$ (T$, M, N),

где M (с какого) – арифметическое выражение, задающее порядковый номер символа в строке, с которого начнется обработка;

N (сколько) – арифметическое выражение, задающее количество символов, подлежащих обработке.

Значения обоих выражений должны быть не меньше единицы. Их дробная часть отбрасывается.

Если третий аргумент (N) опущен (вместе с запятой), то действия проводятся до конца строки.

1. Функция MID$ может быть использована в правой части команды присваивания для **выделения подстроки**:

```
X$=MID$(T$,M,N)
```

где T$ – символьная строка.
Например:

```
10 ' ФУНКЦИЯ MID$ СПРАВА
20 T$="КИБЕРНЕТИКА"
30 X$=MID$(T$,6,3)
40 PRINT X$

RUN
НЕТ
```

2. Функция MID¤ может быть использована и в левой части команды присваивания **для замены части строки** началом другой строки:

MID¤(T¤,M,N)=Z¤ ,

где T¤ – символьная переменная;

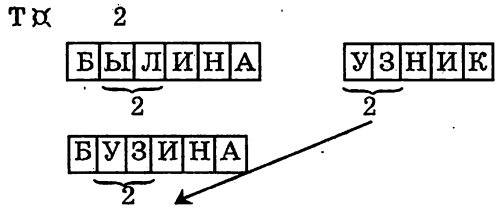
Z¤ – символьная строка.

N первых символов строки Z¤ будет перемещено на место N символов переменной T¤, начиная с позиции M.

Например:

```
10 ' ФУНКЦИЯ MID¤ СЛЕВА
20 T¤="БЫЛИНА"
30 MID¤(T¤,2,2)="УЗНИК"
40 PRINT T¤

RUN
БУЗИНА
```



Действие, выполняемое функцией MID¤ в левой части команды присваивания, можно выразить через функцию MID¤ в правой части:

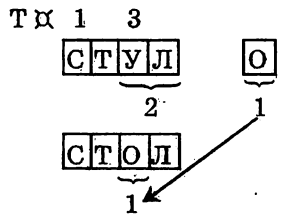
T¤=MID¤(T¤,1,M-1)+MID¤(Z¤,1,N)+MID¤(T¤,M+N)

Если длина строки Z¤ меньше N (LEN(Z¤)<N), то в T¤ заносится вся строка Z¤, остальные символы переменной T¤ остаются без изменений.

Например:

```
10 ' СЛУЧАЙ КОГДА LEN(Z¤)<N
20 T¤="СТУЛ"
30 MID¤(T¤,3,2)="О"
40 PRINT T¤
```

```
RUN
СТОЛ
```



Используя функцию MID¤, можно проделать множество занимательных вещей. Например:

1) Эту функцию можно использовать для последовательного выделения из строки одного символа при каждом проходе цикла:

```
10 INPUT "НАБЕРИТЕ СЛОВО:";S¤
20 LET L=LEN(S¤)
30 FOR I=1 TO L
40 PRINT TAB(I);MID¤(S¤,I,1)
50 NEXT I
```

Оператор в строке 40 заставляет компьютер выводить каждый раз одну букву из слова в позицию I, начиная с буквы с номером 1 (слева направо):

```
RUN
НАБЕРИТЕ СЛОВО:?БАРАБАН
          Б
           А
            Р
              А
                Б
                  А
                    Н
```

Если вы хотите вывести справа налево, то перепишите строку 30 так:

```
FOR I=L TO 1 STEP-1.
```

2) Можно написать программу, которая определяла бы, сколько раз в данном слове встречается указанная буква:

```
10 INPUT"КАКУЮ БУКВУ БУДЕМ ИСКАТЬ";B¤
20 INPUT"В КАКОМ СЛОВЕ";S¤
30 FOR I=1 TO LEN(S¤)
40 IF MID¤(S¤,I,1)=B¤ THEN N=N+1
50 NEXT I
60 PRINT "В СЛОВЕ ";S¤
70 PRINT N;" БУКВ ";B¤
```

Или, например, необходимо в данном слове всюду заменить одну букву на другую:

```
10 INPUT"В КАКОМ СЛОВЕ ЗАМЕНЯТЬ";S¤
20 INPUT"КАКУЮ БУКВУ";E¤
30 INPUT"НА КАКУЮ";I¤
40 FOR I=1 TO LEN(S¤)
50 IF MID¤(S¤,I,1)=E¤ THEN S¤=MID¤(S¤,1,I-1)
+I¤+MID¤ (S¤,I+1,LEN(S¤)-I)
60 NEXT I
70 PRINT "НОВОЕ СЛОВО:";S¤
```

Поясним 50-ю строку. Пусть буква E¤="Е" в слове S¤ стоит на I-м месте и ее надо заменить на букву I¤="И". Слово S¤ разобьем на три части: первая – часть слова S¤ от начала буквы "Е" (его длина I-1), вторая часть – сама буква "Е", третья часть – все остальное (его длина равна LEN(S¤)–I). Замена коснется только второй части, а первая и третья части останутся нетронутыми, поэтому результат такой замены можно записать так:

```
MID¤(S¤,1,I-1)+"E"+MID¤ (S¤,I+1, LEN(S¤)-I).
```

3) Можно составить программу бегущей строки:

```
10 CLS
20 Y=11
30 A¤="И Н Ф О Р М А Т И К А - ОЧЕНЬ
ИНТЕРЕСНАЯ НАУКА В ШКОЛЬНОМ КУРСЕ "
40 FOR J=0 TO 63
50 PRINT AT(63-J,Y) MID¤ (A¤,1,1+J)
60 FOR J1%=0% TO 1000%
70 NEXT J1%,J
80 FOR J=0 TO 63
90 PRINT AT(0,Y) MID¤(A¤,1+J)
```

```
100 FOR J1%=0% TO 1000%
110 NEXT J1%,J
120 GOTO 20
```

❐

При работе с текстовой информацией часто необходимо знать, **какая клавиша была** только что **нажата**. Для этого служит функция INKEY¤, однако в отличие от оператора INPUT компьютер в данном случае не останавливается и не ожидает вас, а продолжает выполнять программу (не высвечивая на экране курсор или символы, соответствующие нажимаемым клавишам).

Если нужно, чтобы программа останавливалась и не возобновляла работу до момента нажатия пользователем какой-либо клавиши, то следует повторять выполнение до тех пор, пока это не произойдет, что можно реализовать различными способами:

```
10 LET X¤=INKEY¤
20 IF X¤=""THEN GOTO 10
30 REM ОСТАЛЬНАЯ ПРОГРАММА
```

В этом примере компьютер повторяет строку 10 до тех пор, пока не нажата какая-либо клавиша.

Возможны другие примеры для того, чтобы заставить компьютер ждать:

```
10 LET K¤=INKEY¤
20 IF LEN(K¤)=0 THEN 10 ELSE 100
100 REM ОСТАЛЬНАЯ ПРОГРАММА

10 PRINT"НАЖМИТЕ ЛЮБУЮ
КЛАВИШУ ДЛЯ ПРОДОЛЖЕНИЯ..."
20 IF INKEY¤="" THEN 20
30 REM ПРОДОЛЖЕНИЕ...
```

ЧТО МОЖЕТ КОМПЬЮТЕР

Музыка души

Если у вас нет слуха, вы не владеете ни одним из музыкальных инструментов, не можете выразить словами звучащую у вас в душе музыку – вам поможет новый электронный прибор, разработанный двумя молодыми американцами: инженером и музыкантом.

Прибор, которому авторы дали недвусмысленное название "Биомьюз", преобразует биотоки мозга, мускулов рук, ног и век, поступающие с датчиков, в музыку,

❐

Внутри компьютера символы представляются числовыми кодами. Используя эти числа, можно выполнять разные действия над символами. Функция CHR **преобразует число в символ**, а функция ASC, наоборот, **символ в его числовой код**. Большинство компьютеров используют для представления символов так называемый американский стандартный код (ASCII–КОД) (см. § 11). В руководстве пользователя вы найдете таблицу символов для своего компьютера.

Попробуйте исполнить несколько операторов PRINT ASC ("СИМВОЛ") и PRINT CHR (ЧИСЛО). Некоторые числовые коды соответствуют таким клавишам, как пробел или (ВК), и поэтому при их использовании на экране ничего не появится. Из своего руководства пользователя можно узнать, какие это числовые коды, или запустив такую программу:

```
10 C$=INKEY$
20 IF C$= "" THEN 10
30 PRINT ASC(C$)
40 GOTO 10
```

Программа печатает коды клавиш.

Функции ASC и CHR используют в профессиональных программах, когда хотят проверить правильность ответов или данных, введенных пользователем.

❐

Поиск в строке заданного фрагмента осуществляется функцией

INSTR (N,S$,P$)

(от англ. in string – в строке),
где N – с какой позиции вести поиск (числовая константа, переменная или арифметическое выражение);

воспроизводимую синтезатором. Программа, управляющая работой оригинального музыкального инструмента, написана таким образом, чтобы как можно точнее преобразовать жестикулярные и мимические ассоциации человека в музыкальную форму. Например, направление взгляда перемещает источник звука в стереофоническом пространстве, работа мускулов ног задает музыкальный ритм, движением костей формируется гармонический ряд, а биотоки мозга определяют мелодию. Программа может быть настроена под конкретного исполнителя, что позволит ему наиболее полно реализовать свои творческие возможности.

Закрепив датчики, "биомузыкант" выходит на сцену и начинает изящно пританцовывать, жестикулировать, меч-

тать, и весь зал может слышать, о чем поет его душа.

И хотя человек, имеющий классическую музыкальную подготовку, возможно, сумеет извлечь из нового инструмента значительно более изящную мелодию, "Биомьюз", по мнению авторов, принесет огромную пользу и людям с поврежденными функциями моторного контроля, а также тем, кто надолго прикован к постели. "Биомьюз" – это еще и средство "музыкальной терапии", и способ почувствовать себя полноценным человеком, способным творить и создавать прекрасное.

S$ – основная строка (в какой ищут);

P$ – подстрока (которую ищут).

Результатом действия этой функции является число, указывающее номер позиции в строке S$, с какой в нее входит подстрока P$. Поиск в строке S$ происходит с позиции N.

Если указанный фрагмент (подстроки P$) в строке S$ не содержится, то значение функции будет равным нулю.

Если опущен параметр N (вместе с запятой), то поиск будет вестись с начала строки.

Приведем примеры:

1.
```
10 C=INSTR(1,"АВТОСТОП","ТО")
20 PRINT C

RUN
 3
```

Работа программы будет аналогична, если строку 10 переписать:

```
10 C=INSTR("АВТОСТОП","ТО")
```

Здесь по умолчанию компьютер берет значение N, равное 1.

2.
```
10 K=INSTR("АВТОСТОП","СНОП")
20 PRINT K

RUN
 0
```

то есть подстрока "СНОП" не содержится в строке "АВТОСТОП".

3.
```
10 Z=INSTR(1,"АВТОСТОП","ОСТ")
20 PRINT Z

RUN
 4
```

4.
```
10 Z=INSTR(4,"АВТОСТОП","ТО")
20 PRINT Z

RUN
 6
```

5.
```
10 A$="АВТОСТОП"
20 B$="СТОП"
30 C=INSTR(1,A$,B$)
40 PRINT C

RUN
 5
```

Эта функция позволяет осуществлять некоторые операции над множествами.

Например:

6.
```
10 'НОМЕР БУКВЫ В АЛФАВИТЕ
20 A$="АБВГДЕЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЬЫ
ЭЮЯ"
30 INPUT"ВВЕДИТЕ БУКВУ";B$
40 P=INSTR(A$,B$)
50 IF P=0 THEN PRINT"ВЫ СДЕЛАЛИ
НЕПРАВИЛЬНЫЙ ВВОД" ELSE 70
60 GOTO 30
70 PRINT "БУКВА ";B$;" ЕСТЬ";P;"-Я БУКВА В
АЛФАВИТЕ"

RUN
ВВЕДИТЕ БУКВУ?К (ВК)
БУКВА К ЕСТЬ 11-Я БУКВА В АЛФАВИТЕ
```

7.
```
10 'ПРОВЕРКА НАЛИЧИЯ СЛОВА В СЛОВАРЕ
20 FOR K=1 TO 5
30 READ M$(K)
40 NEXT K
50 INPUT"ВВЕДИТЕ СЛОВО";S$
60 FOR K=1 TO 5
70 IF INSTR (1,M$(K),S$)<>0 THEN 500
80 NEXT K
90 PRINT"ТАКОГО СЛОВА В СЛОВАРЕ НЕТ"
100 GOTO 50
```

```
110 DATA АЛГОРИТМ,ПРОГРАММА,МОДЕЛЬ,
ФАЙЛ,ПРОЦЕССОР

500 ' ОБЪЯСНЕНИЕ ЗНАЧЕНИЯ СЛОВА
510 ON K GOSUB 1000,2000,3000,4000,5000
520 GOTO 50
1000 PRINT "АЛГОРИТМ - ДЕТАЛЬНЫЙ ПЛАН
ДЕЙСТВИЙ,НАПРАВЛЕННЫХ НА ПОЛУЧЕНИЕ
ОПРЕДЕЛЕННЫХ РЕЗУЛЬТАТОВ"
1010 RETURN

2000 PRINT "ПРОГРАММА - АЛГОРИТМ,
ЗАПИСАННЫЙ НА ЯЗЫКЕ, ПОНЯТНОМ ДАННОЙ ЭВМ"
2010 RETURN

3000 PRINT"МОДЕЛЬ - СИСТЕМА СООТНОШЕНИЙ,
ОТРАЖАЮЩИХ СВОЙСТВА ОБЪЕКТА"
3010 RETURN

4000 PRINT "ФАЙЛ - СОВОКУПНОСТЬ ДАННЫХ,
ЗАПИСАННЫХ В ЭВМ НА ВНЕШНЕМ НОСИТЕЛЕ"
4010 RETURN

5000 PRINT"ПРОЦЕССОР - ОСНОВНОЕ УСТРОЙСТВО
ЭВМ,ОСУЩЕСТВЛЯЮЩЕЕ ВЫПОЛНЕНИЕ ПРОГРАММ
И ИСПОЛНЕНИЕ ОПЕРАЦИЙ НАД ДАННЫМИ."
5010 RETURN
```

С помощью этой функции можно подсчитать количество гласных (или согласных) в слове:

8.
```
10 ' ПОДСЧЕТ ГЛАСНЫХ В СЛОВЕ:
20 N=0
30 G¤="АЕИОУЫЭЮЯ"
40 INPUT"ВВЕДИТЕ СЛОВО";P¤
50 IF P¤=" " THEN 40
60 L=LEN(P¤)
70 FOR I=1 TO L
80 S¤=MID¤(G¤,I,1)
85 IF INSTR(P¤,S¤)<>0 THEN N=N+1
90 NEXT I
100 IF N=0 THEN PRINT"ГЛАСНЫХ НЕТ" ELSE 120
110 STOP
120 PRINT"ИМЕЕТСЯ ";N;" ГЛАСН";
```

```
130 IF N>1 THEN PRINT"ЫХ": REM ДЛЯ МНОЖ.
ЧИСЛА
140 IF N=1 THEN PRINT"АЯ": REM ДЛЯ ЕДИН. ЧИСЛА
150 END
```

Эту программу можно переделать так, чтобы она искала гласные. Для этого нужно переписать и добавить строки:

```
35 B¤=" "
85 IF INSTR(P¤,S¤)=0 THEN 90
87 IF INSTR(B¤,S¤)=0 THEN B¤=B¤+S¤
100 IF LEN(B¤)=0 THEN PRINT"ГЛАСНЫХ НЕТ"
ELSE 120
110 GOTO 125
120 IF LEN(B¤)=1 THEN PRINT"ВСТРЕЧЕННАЯ
ГЛАСНАЯ:"; B¤ ELSE 130
125 STOP
130 PRINT"ВСТРЕЧЕННЫЕ ГЛАСНЫЕ:";
140 FOR I=1 TO LEN(B¤)
145 PRINT" ";MID(B¤,I,1);
147 NEXT I
```

❐

Для порождения цепочки из какого-то количества букв или других символов служит функция STRING – строка (от англ. string – строка).

```
STRING¤(N,M,S¤),
```

где N – длина создаваемой строки;

M – число, соответствующее коду ASCII;

S¤ – символьное выражение, первым символом которого заполняется создаваемая строка.

Например:

```
10 INPUT"ВВЕДИТЕ КОЛИЧЕСТВО ПОВТОРЕНИЙ";N
20 INPUT"КАКИМ СИМВОЛОМ";S¤
30 P¤=STRING¤(N,S¤)
40 PRINT P¤
```

Выражения STRING¤(41,32) и STRING¤(41," ") приводят к одинаковому результату. Эта функция может быть полезна при построении диаграмм для функций.

Известно ли вам, что...

... одно время по американским ПК ходила программа с завлекательным названием "Rockvideo". Запустив ее, владелец компьютера мог наслаждаться мульфильмом, посвященным звезде эстрады и кино Мадонне; однако кончался мультик сообщением: "Только идиот использует компьютер для рассматривания кинозвезд!".

Мораль этой "электронной басни" подействовала недостаточно. Фирма IBM выпускает в продажу набор "IBM PS/2 TV" (плата видеоадаптера, динамик, телентюнер, программное обеспечение), позволяющий смотреть телепрограммы (до 70 одновременно – вот извращенцы!) на экране компьютера PS/2.

Существует функция VAL (от англ. vallue – значение), служащая для **преобразований символов-цифр в число.**

Допустим, в символьной строке хранится набор символов, состоящий из последовательности цифр, скажем,

X$ ="1995" или "02",

и у нас возникла необходимость использовать набор этих цифр как число. Однако написать, например, X$/15 мы не можем, так как в арифметическом выражении использование символьных переменных (констант) не допускается. Обойти это препятствие помогает функция, имеющая следующий вид:

VAL ("СТРОКА").

Вернемся к нашему примеру и запишем правильно арифметические выражения:

```
10 X$="1995"
20 PRINT VAL(X$)/15

RUN
 133 .
```

Если набор символов содержит точку, то последовательность преобразуется в вещественное число и точка будет разделять целую и дробную части:

```
10 PRINT VAL("234.105")+66

RUN
 300.105
```

Все символы, стоящие после цифровой последовательности, отбрасываются.

Еще примеры:

```
20 PRINT VAL("0083")

RUN
 83

30 PRINT VAL("05 МЕСЯЦ")

RUN
 5

40 PRINT VAL("-20.75+E")

RUN
-20.75
```

Если в последовательности символов на первом месте стоит не цифра (и не точка), результат действия на нее функции VAL будет нулевым:

```
50 PRINT VAL("F19.5")

RUN
 0
60 PRINT VAL("ГОД 1995")

RUN
 0
```

Очень полезно функцию VAL использовать для проверки при вводе в операторе INPUT строковой переменной:

```
10 PRINT"В КАКОМ МЕСЯЦЕ ВЫ РОДИЛИСЬ?"
20 PRINT"ВВЕДИТЕ, ПОЖАЛУЙСТА, ЧИСЛО ОТ 1 ДО 12"
30 INPUT M¤
40 IF VAL(M¤)<1 OR VAL(M¤)>12 THEN 30
```

В таком случае сбоя в программе не будет, даже если пользователь введет слово, когда программа ожидает ввода числа. Функция VAL предлагает компьютеру воспринимать строковое значение как число.

□

Если нам понадобится сделать обратное преобразование, т.е. **перевести число в последовательность символов**, состоящих из цифр этого числа, мы можем использовать функцию:

STR¤ (ЧИСЛО)

Использование этой функции позволяет "разрезать" и "склеивать" (подчеркиваем, склеивать, а не складывать!) числа.

Сравните два результата:

```
10 PRINT 2+3
RUN
 5
20 PRINT STR¤(2)+STR¤(3)
RUN
 2 3
```

Не используя арифметических операций, выделим последнюю цифру числа:

```
10 ' ПОСЛЕДНЯЯ ЦИФРА ЧИСЛА
20 Q=256
30 N¤ =STR¤(Q)
40 A¤=MID¤(N¤,LEN (N¤),1)
50 PRINT VAL(A¤)
RUN
 6
```

Результат выведен в числовом виде – перед числом 6 выделена позиция под знак.

Функция STR¤ употребляется и в тех случаях, когда нужно **напечатать положительное число, не выделяя позиции под знак.**

Сравним две команды печати:

```
10 ' ВЫВОД ПОЛОЖИТЕЛЬНОГО ЧИСЛА С ПЕРВОЙ
ПОЗИЦИИ
20 K=17.5
30 PRINT K
40 PRINT MID¤(STR¤(K),2)
RUN
 17.5
17.5
```

Для любого числа Z справедливо тождество:

```
VAL(STR¤(Z))=Z
```

Проверьте свои знания

1. Что вы понимаете под операцией "конкатенация"?
2. Каково назначение функции LEN? Как она работает?
3. Зачем служит функция MID¤? Как используется эта функция в левой и правой частях равенства?
4. Как с помощью функции INSTR осуществляется поиск фрагмента в строке?
5. Каково назначение функции STRING¤? Как она работает?
6. Для чего используется функция VAL? Функция STR¤?

Тренировка

I. Используя функцию LEN и оператор IF...THEN, напишите программу, которая находит самое короткое слово.

II. Используя функцию MID¤, замените часть слова "КОРЗИНА" началом слова "АРТИСТ" так, чтобы получить слово "КАРТИНА".

III. Попытайтесь заставить компьютер выводить на экран слово или фразу в обратном порядке. Для этого используйте функцию MID¤ и цикл с шагом −1. Испытайте программу с использованием литературных палиндромов:

Я ИДУ С МЕЧОМ СУДИЯ (ДЕРЖАВИН)
КОТУ ТАЩАТ УТОК
А РОЗА УПАЛА НА ЛАПУ АЗОРА
ИШАКУ КАЗАК СЕНА НЕС, КАЗАКУ КАШИ
КИТ НА МОРЕ РОМАНТИК

IV. Напишите программу поиска позиций, в которых встречается заданная буква в слове.

V. Не используя функцию INSTR, напишите программу поиска данного фрагмента в строке.

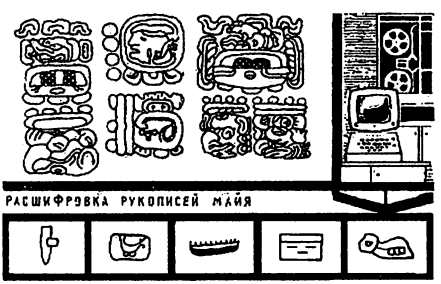
VI. Напишите программу, чтобы при нажатии вами клавиши компьютер выводил на экране фразу: "ПРАЗДНОСТЬ РОЖДАЕТ ПОРОКИ", в противном случае на экране выводились бы пробелы.

VII. Проверьте, является ли шестизначный номер автобусного билета "счастливым" (сумма первых трех цифр равна сумме последних трех), не используя операции умножения.

§ 40. Использование ЭВМ для кодирования сообщений

Когда-то народы были бесписьменными. Кое-где такие народы существуют и в наше время.

Живущие две тысячи лет назад финикийцы изобрели буквы только для гласных звуков. Затем древние греки на их основе сложили свой греческий алфавит, который содержал и согласные и гласные звуки. Впоследствии греческий алфавит "завоевал" весь мир: он помог многим народам создать свою письменность. Так, человек изобрел значки, в которые как бы "одели" звуки, "сшили для них форму". Но не все языки понятны современному человеку. Много лет ученым не удавалось расшифровать язык древнего народа майя (Южная Америка). Лишь недавно с появлением ЭВМ их усилия увенчались успехом. Машина помогла расшифровать и другую письменность – киданьскую (Центральная Азия). Она быстро подсчитала, какие символы и в каких сочетаниях встречаются в словах чаще, какие – реже. В результате в каждом слове киданьского языка удалось выделить корень, суффикс и окончание. После этого не составило труда понять этот язык (он оказался похожим на монгольский).



Ученые при расшифровке письменных знаков индейцев майя применили ЭВМ

Но в жизни приходится не только дешифровать, но и шифровать сообщения.

Вспомните детективные фильмы, там всегда разведчикам приходилось зашифровывать свои сообщения, а в "центре" их дешифровали.

Работа шифровальщиков чисто автоматическая, а значит, ее можно поручить ЭВМ. Существует огромное мно-

жество способов шифрования текстов. Мы рассмотрим лишь некоторые из них.

❐

Один из методов шифровки сообщений состоит в том, что после каждого символа вставляется некоторая буква (каждый раз разная).

Другой способ шифровки заключается в том, что буквы-сообщения поочередно заменяются на предшествующую и последующую буквы алфавита.

Например:

```
10 PRINT"ВВЕДИТЕ СООБЩЕНИЕ"
20 INPUT S¤
30 FOR K=1 TO LEN(S¤)
40 LET X=ASC(MID¤(S¤,K,1))
50 IF X<65 OR X>90 THEN LET N=X ELSE 100
60 IF INT(K/2)=K/2 THEN N=X+1
70 IF INT(K/2)<>K/2 THEN LET N=N-1
80 IF N <65 THEN LET N=N+26
90 IF N >90 THEN LET N=N-26
100 PRINT CHR¤(N)
110 NEXT K
120 STOP
```

Поясним эту программу. Строка 30 делает число повторений цикла, равное числу символов в сообщении S¤. В строке 40 компьютер выбирает каждый раз одну букву и запоминает ее ASCII-код в переменной X. В строке 50 интервал 90<X>65 выбран из расчета принадлежности каждого символа к буквам (возможно, в вашем компьютере этот интервал иной – посмотрите в инструкцию пользователя). В строках 60-70 проводится проверка, является ли переменная цикла K четным или нечетным числом. Для этого K делится на 2, а результат деления с помощью функции INT (см. § 24) приводится к целому числу. Затем компьютер проверяет, равно ли это целое число K/2. Если эта проверка подтверждается, то K является четным числом.

Строки 60-70 как раз и меняют буквы, прибавляя 1 к X, если счетчик цикла K является нечетным числом.

Строки 80-90. Измененный номер буквы (N), если он выходит за любой конец алфавита (из 26 букв), переводят в другой конец (прибавляя или вычитая число 26).

Строка 100 выводит закодированную букву.

Можно эту программу изменить. Для этого исходный алфавит сдвигают на определенное количество (М) букв. Величина сдвига определяется ключевым числом:

```
25 INPUT"ВВЕДИТЕ КЛЮЧЕВОЕ ЧИСЛО";M
60 LET N=K+M
70 ' ЭТУ СТРОКУ В ПРЕДЫДУЩЕЙ ПРОГРАММЕ - УДАЛИТЬ
```

Например, если ключевое слово равно 5, то исходный алфавит сдвинется на пять букв:

Алфавит: A B C D E F G H I J L M N O P Q S T U V W X Y Z

Алфавит кода: F G H I J L M N O P Q S T U V W X Y Z A B C D E

Такой способ шифровки называют **кодированием с ключевым числом.**

Существует еще способ кодирования с помощью так называемого **циклового кода.** Для получения этого кода к ASCII-коду каждой буквы сообщения прибавляют значение переменной цикла (К). Чтобы реализовать эту идею, необходимо в исходной программе строку 70 удалить, а строку 60 заменить на такую:

```
60 LET N=K+K
```

"Пляшущие человечки" — шифр, разгаданный Шерлоком Холмсом в одноименном рассказе Конан Дойла

Знаменитый сыщик сразу понял, что перед ним шифр, и начал искать ключ. Вскоре ключ был найден, и Шерлок Холмс, разгадав значение каждой фигурки-буквы, прочитал странные записки. Мало того, тем же самым шифром он написал письмо преступнику, и преступник попал в руки правосудия.

А чтобы получить программу декодирования циклового кода, нужно снова изменить строку 60:

```
60 LET N=X-K
```

Можно закодировать сообщение с помощью **перестановочного кода**. Чтобы получить этот код, все символы сообщения, включая пробелы, делят на пары, а затем символы в каждой паре меняют местами:

```
10 INPUT"ПРЕДЛОЖЕНИЕ, КОТОРОЕ КОДИРУЕТСЯ";P¤
20 FOR K=1 TO LEN(P¤) STEP 2
30 PRINT MID¤(P¤,K+1,1);
40 PRINT MID¤(P¤,K,1);
50 NEXT K
```

В строке 20 оператор STEP 2 заставляет компьютер изменять переменную цикла К с шагом 2. При каждом повторении цикла компьютер выводит на экран сначала второй символ (К+1) из текущей пары символов, а за ним первый (К) из этой же пары.

Например:

```
RUN
ПРЕДЛОЖЕНИЕ, КОТОРОЕ КОДИРУЕТСЯ?
ЮСТАС-АЛЕКСУ КЭТ ПОД КОЛПАКОМ
СЮАТ-СЛАКЕУСК ТЭП ДОК ЛОАПОК М
```

Чтобы "усилить" шифровку, используют **ключевые фразы**. Под ключевой понимают такую фразу, в которой встречаются все буквы данного алфавита (а также знаки: запятая, точка, дефис, кавычки, восклицательный и вопросительный знаки), например:

```
ЦЕЛЬ ДЭНДИ:"МАЛЕНЬКИЕ НЕПРИЯТНОСТИ НЕ
ДОЛЖНЫ МЕШАТЬ БОЛЬШОМУ УДОВОЛЬСТВИЮ."
НО!"ФАРАОНОВЫ ТОЩИЕ КОРОВЫ,СЪЕВШИЕ ТУЧНЫХ."-
ГДЕ ЛОГИКА?
```

Суть программ шифровки текстов с помощью ключевой фразы заключается в том, что каждый символ секретного текста заменяется на номер этого символа в ключевой фразе.

```
10 INPUT"ВВЕДИТЕ КЛЮЧЕВУЮ ФРАЗУ";K¤
20 INPUT"ВВЕДИТЕ СЕКРЕТНЫЙ ТЕКСТ";T¤
30 FOR I=1 TO LEN(T¤)
40 P¤=MID¤(T¤,I,1)
50 GOSUB 200
60 PRINT H;
70 NEXT I
75 PRINT"ШИФРОВКА ЗАКОНЧЕНА"
80 END

200 REM ПОДПРОГРАММА КОДИРОВАНИЯ БУКВ
210 FOR J=1 TO LEN(K¤)
220 IF MID¤(K¤,J,1)<>P¤ THEN 250
230 H=J
240 J=LEN(K¤)
250 NEXT J
260 RETURN
```

Поясним работу подпрограммы. Она предназначена для определения номера позиции, которую занимает символ P¤. Но если символ в ключевой фразе (K¤) встречается несколько раз? Чтобы переменной H присваивался номер первой, а не последней встретившейся позиции буквы, нужно произвести выход из цикла, как только этот символ впервые встретился (строки 220-240).

Напишем теперь программу **дешифровки**. ЭВМ опять должна запросить ключевую фразу. Затем, запрашивая последовательно числа, кодирующие символы зашифрованного сообщения, она составит из соответствующих символов исходный текст. Например, расшифруем сообщение:

48 29 10 26 3 10 1

Запишем его в символьную переменную T¤. Сначала переменной T¤ присвоим пустое слово, затем берем 48 й символ ключевой фразы (это буква "Ш") и добавляем его к T¤. После этого допишем к T¤ последовательно 29 10 26 3 10 1– символы ключевой фразы. Получим слово "ШТИРЛИЦ".

Но здесь есть трудность. Ведь в этом разобранном примере мы видели шифрованное сообщение целиком и могли легко определить, когда кончать расшифровку. Труд-

ность 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 определиться в том, как машина будет обнаруживать окончание шифрованного сообщения. Закончив шифровку, введем число 0. Договоримся, что ЭВМ будет воспринимать его как сигнал об окончании сообщения.

С помощью этой решетки вы прочтете, что здесь написано

```
10 INPUT"ВВЕДИТЕ КЛЮЧЕВУЮ ФРАЗУ";K¤
20 LET T¤=" "
25 J=1
30 PRINT"ВВЕДИТЕ КОД";J;"СИМВОЛА"
40 INPUT H(J)
80 IF H(J)<>0 THEN 90 ELSE 115
90 T¤=T¤+MID¤(K¤,H(J),1)
100 J=J+1
110 GOTO 30
115 PRINT T¤
120 PRINT"ДЕШИФРОВКА ОКОНЧЕНА"
130 END
```

Проверьте свои знания

1. Какие способы кодировки сообщений вы знаете?
2. В чем суть шифровки сообщений с помощью циклового кода? Перестановочного кода?
3. Что такое ключевая фраза?
4. На чем основаны способы дешифровки сообщений?

§ 41. Порядок и беспорядок

Мы в течение дня много раз что-нибудь упорядочиваем: книги, этикетки. Мы можем упорядочить фамилии наших друзей по алфавиту в нашем дневнике. Нам приходится сортировать данные как по возрастанию, так и по убыванию (например, исторические даты). Небольшие списки данных очень просто отсортировать вручную, однако при большом объеме компьютер выполняет эту работу значительно быстрее и аккуратнее. Нужно только показать ему, как это делать.

Программы, специально предназначенные для выполнения сортировки данных, обычно называют **сортировками**. Существует множество различных программ (а точнее, способов) сортировки, написанных на языке Бейсик.

Мы обсудим только два вида программ сортировки. Один из них называется **сортировкой методом пузырька** (ниже вам станет понятно, почему он получил такое название), а другой – **сортировкой Шелла** (названной так в честь предложившего ее автора).

Прежде чем приступить к этим конкретным видам сортировки, обсудим некоторые общие положения.

Сам процесс сортировки заключается в сравнении двух элементов, выбранных некоторым образом из массива. Если они находятся не в отсортированном порядке относительно друг друга, то меняются местами.

Различные алгоритмы различаются только процедурой выбора элементов для сравнения.

Большинство операционных систем содержат стандартные программы сортировки. Но для изучения эти программы не всегда приемлемы. Дело в том, что стандартные программы сортировки во многих случаях не являются лучшими. Они могут быть эффективны для очень больших наборов данных, но для более коротких (содержащих не свыше 100 элементов) лучшей может оказаться самодельная программа сортировки.

Изучим сначала предложенные методы упорядочения, а затем, возможно, вы сами придумаете другие способы сортировки.

Метод пузырька

При сортировке чисел таким методом компьютер берет из самого начала неупорядоченного списка первые два элемента и сравнивает их. Если порядок расположения этих элементов отличается от требуемого, компьютер меняет их местами и затем сравнивает смежную пару элементов. Подобные сравнения и перестановки производятся компьютером по всему списку так, чтобы более крупные числа постепенно "всплывали", как пузырьки в конце списка.

Представленная ниже программа является программой **сортировки чисел методом пузырька:**

```
10 REM СОРТИРОВКА МЕТОДОМ
ПУЗЫРЬКА ЧИСЕЛ
20 INPUT"СКОЛЬКО ЧИСЕЛ
НУЖНО СОРТИРОВАТЬ";N
30 DIM A(N)
40 FOR I=1 TO N
50 INPUT"ВВЕДИТЕ ЧИСЛО И
НАЖМИТЕ (ВК)";A(I)
70 NEXT I
75 PRINT
76 PRINT
80 FOR I=1 TO N-1
90 FOR J=I+1 TO N
100 IF A(I)>A(J) THEN 104
 ELSE 110
104 E=A(I)
105 A(I)=A(J)
107 A(J)=E
110 NEXT J
120 NEXT I
130 FOR I=1 TO N
140 PRINT A(I);" ";
150 NEXT I
160 END
```

Кто там?

Задать этот вопрос, прежде чём открывать дверь, никогда не вредно. Но нет ли более эффективного способа обезопасить себя? Поразмыслив над этим, японская фирма EST Co. Ltd выпустила несколько вариантов электронного швейцара, заменяющего сразу пять глазков (по числу предполагаемых дверей).

Связь между телекамерами, устанавливаемыми над дверьми, и центральным пунктом — беспроволочная, так что перерезать провода врагу не удастся. Управляет охранным ра-

диотелекомплексом скромная БИС, которую вполне можно считать полноценным компьютером, поскольку она оборудована всем, что подобает иметь приличной ЭВМ: памятью, процессором и даже периферийными устройствами в виде дистанционно управляемых засовов и замков.

Теперь, услышав звонок в дверь или сирену сигнализации, вы подходите к маленькому экрану с изображением пришедшего и, если по его внешнему виду непонятна цель прихода, уточняете ее. Хотя иногда именно за такими дорогими приборами и приходят незваные гости.

Запустим программу на выполнение. Дадим (в строке 20) переменной N значение 5. Компьютер будет сравнивать 5 чисел. С помощью цикла, начинающегося в строке 40, компьютер получит 5 чисел, которые вы введете с помощью клавиатуры, и присвоит их переменным A(I). Например, вы введете серию чисел: 5 – 9 – 2 – 3 – 1. Компьютер присвоит значения: A(1)=5; A(2)=9; A(3)=2; A(4)=3; A(5)=1.

Что делает строка 100? Сначала сравнивает A(1) с A(2), т.е. 5 и 9, и, если необходимо, меняет местами их значения. Переходит на строку 110 и берет следующее значение J. Теперь сравнивает A(1) с A(3) и т.д. Когда J станет равным 5, переходит на строку 120 и берет следующее значение I (равное 2). Теперь сравнивает A(2) с A(3) и т.д.

Мы используем I и J для того, чтобы индексировать переменные и сравнивать первый элемент ряда со вторым и всеми остальными, а затем то же самое делаем со вторым и всеми следующими и т.д.

В этом примере мы упорядочим числа от меньшего к большему. Для упорядочения от большего к меньшему достаточно только изменить знак сравнения ">" (больше) на знак сравнения "<" (меньше) в строке 100.

Следующая программа представляет собой **сортировку слов методом пузырька**. Она очень похожа на предыдущую программу, за исключением того, что переменные для хранения числовых данных теперь стали строковыми переменными.

Для сравнения букв компьютер использует тот же самый метод, что и для сравнения чисел. Внутри компьютера буквы и другие символы представляются в виде чисел, и поэтому, когда вы предлагаете компьютеру сравнить какие-либо символы, он сравнивает их числовые коды. Чтобы сравнить между собой два слова, компьютер сначала сравнивает первые буквы каждого слова и, если они одинаковы, сравнивает вторые буквы, затем третьи и т.д. В строковые переменные можно помещать как буквы, так и цифры, поэтому программу сортировки слов методом пузырька можно использовать для данных, которые содержат слова и числа, например, адреса, списки телефонов, указатели журнальных статей и т.д.

```
10 REM СОРТИРОВКА МЕТОДОМ ПУЗЫРЬКА СЛОВ
20 INPUT"СКОЛЬКО СЛОВ НУЖНО ОТСОРТИРОВАТЬ"; N
30 DIM A¤(N)
40 FOR I=1 TO N
50 INPUT"ВВЕДИТЕ СЛОВО";A¤(I)
70 NEXT I
75 PRINT
76 PRINT
80 FOR I=1 TO N-1
90 FOR J=I+1 TO N
100 IF A¤(I)>A¤(J) THEN 104 ELSE 110
104 E¤=A¤(I)
105 A¤(I)=A¤(J)
107 A¤(J)=E¤
110 NEXT J
120 NEXT I
130 FOR I=1 TO N
140 PRINT A¤(I);" ";
150 NEXT I
160 END
```

Если вы собираетесь отсортировать большой объем данных, то следует учесть, что сортировка методом пузырька очень медленна. Некоторым компьютерам может потребоваться почти одна минута для сортировки этим методом всего 50 элементов.

Сортировка Шелла

Следующая программа является сортировкой Шелла для чисел. Она примерно в три раза быстрее сортировки методом пузырька.

При сортировке данных методом Шелла компьютер делит список подлежащих сортировке элементов пополам и сравнивает все элементы одной половины списка с элементами другой. Затем снова делит список пополам и, используя тот же прием перестановки, как и в предыдущем методе, осуществляет за проход намного больше сравнений элементов и их перемещений по списку.

```
10 REM СОРТИРОВКА ЧИСЕЛ МЕТОДОМ ШЕЛЛА
20 INPUT"СКОЛЬКО ЧИСЕЛ НУЖНО СОРТИРОВАТЬ";N
30 DIM A(N)
40 FOR I=1 TO N
50 PRINT"ЧИСЛО";I
60 INPUT A(I)
70 NEXT I
80 C=N
90 C=INT(C/2)
100 IF C=0 THEN 240
110 D=N-C
120 E=1
130 F=E
140 G=F+C
150 IF A(F)<=A(G) THEN 210
160 TE=A(F)
170 A(F)=A(G)
180 A(G)=TE
190 F=F-C
200 IF F>0 THEN 140
210 E=E+1
220 IF E>D THEN 90
230 GOTO 130
240 PRINT"ОТСОРТИРОВАННЫЙ СПИСОК:"
250 FOR I=1 TO N
260 PRINT A(I);" ";
270 NEXT I
```

В строке 90 встретился новый оператор INT. Этот оператор отбрасывает все цифры после десятичной точки, т.е. оставляет только целую часть числа.

Чтобы преобразовать эту программу для сортировки строк символов, необходимо заменить числовой массив N на строковый N$, (в строках 30, 60, 150-180,260), заменить числовую переменную TE на строковую TE$ (строки 160, 180) и переписать операторы PRINT.

Функция INT отрубывает (отбрасывает) дробную часть десятичной дроби

Если бы вы стали испытывать сортировки – методом пузырька и методом Шелла – на списке из нескольких чисел, вы не заметили бы, насколько быстрее сортировка Шелла. Чтобы действительно испытать программы этих двух методов, можно включить в них генерацию набора случайных чисел, а затем зафиксировать, сколько времени потребуется каждой программе для упорядочения этих чисел. Но перед тем, как это осуществить, мы поподробнее рассмотрим работу **датчика случайных чисел**: RND(X) (randomize – от англ. случайный).

□

В большинстве случаев эта функция "выдает" случайные числа в интервале (0,1) – круглые скобки означают, что интервал открытый, т.е. не содержит своих концов – точек 0 и 1.

Вообще говоря, такие числа принято называть **псевдослучайными** (т. е. почти случайными). Этот термин подчеркивает, что речь идет не об абсолютно случайных числах (между значениями RND(X) существует функциональная зависимость).

Тем не менее псевдослучайные числа, как мы увидим в дальнейшем, достаточно удобны в тех случаях, когда мы сталкиваемся с необходимостью получить в требуемых местах программы "неожиданные", "не прогнозируемые заранее" элементы.

Аналогично гаданию на ромашке датчик RND выбирает случайные числа

Значения функции RND(X) распределены на интервале (0,1) равномерно, т.е., какой бы подынтервал мы ни выбрали, значения функции RND(X) рано или поздно непременно попадают в него.

В дальнейшем мы всегда будем записывать RND(1), хотя встречаются и другие версии Бейсика, в которых аргумент X может быть произвольным. Если вы напишите:

```
NEW
10 PRINT RND(1),RND(1),RND(1)
```

то, когда запустите программу командой RUN, получите приблизительно следующие числа:

```
RUN
.276141   .6644921      .652564
```

или составите такую программу:

```
NEW
10 PRINT RND(1)
20 GOTO 10
```

и запустите командой RUN, будет приблизительно следующее:

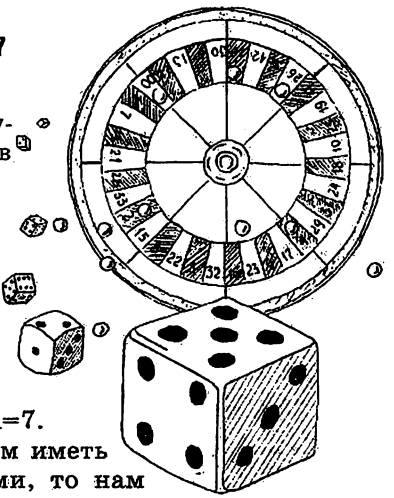
```
RUN
.885978
.436967
.532571
.241439
.914433
ПРЕРЫВАНИЕ В СТРОКЕ 10.
```

Как видите, все эти числа меньше 1. А что же делать, если вы хотите получить числа другого интервала?

Для получения случайных чисел, принадлежащих открытому интервалу (A,B), можно использовать выражение RND(1)*(B—A)+A.

Например, программа:

```
10 FOR I=1 TO 5
20 PRINT RND (1)*2+7
30 NEXT I
```



при выполнении печатает случайные числа, заключенные в интервале (7,9):

```
RUN
8.88697
8.43696
7.24143
8.91443
8.33213
```

В этой программе B=9, A=7.

Теперь, если мы не хотим иметь дело с десятичными знаками, то нам надо в предыдущей программе переписать строку 20 следующим образом:

```
20 PRINT INT(RND(1)*2+7)
```

Строка 20 содержит один набор скобок, заключенный в другой. Это так называемые вложенные скобки.

Самые внутренние скобки содержат выражение RND(1)*2+7. Оно говорит компьютеру: "Выбрать случайное число в интервале между 7 и 9 вместе с десятичными знаками" (это мы уже видели в предыдущей программе). Внешняя пара скобок начинается словами INT, является сокращением для слова INTEGER (от англ. – целый). Компьютер, встретив команду INT, отбрасывает десятичные знаки и берет значение ближайшего наименьшего целого числа.

Если бы мы поместили такую строку 20 в программу и дали команду RUN, то получили бы следующий набор чисел:

```
RUN
 8
 8
 7
 8
 7
```

По результату выполнения программы видно, что числа берутся целые, причем в интервале от 7 до 8. Если же хотим получить числа от 7 до 9, то должны прибавить к нашей формуле слагаемое +1. Тогда получим

```
INT(RND(1)*(B-A+1)+A)
```

А наша строка 20, в соответствии с этой формулой, перепишется так:

```
20 PRINT INT(RND(1)*(2+1)+7)
```

или

```
20 PRINT INT(RND(1)*3+7
```

Обобщив все отмеченное, составим правило использования функции для образования случайных чисел:

---

1. Для получения случайного числа C в интервале (A,B) следует растянуть интервал (0,1) в (B–A) раз и прибавить к результату A, т.е. число C связано с функцией RND(1) следующим образом: C=RND(1)*(B–A)+A.

2. Для получения случайного целого числа C из интервала A,B следует растянуть интервал [0,1] в (B–A+1) раз, прибавить к результату A и взять целую часть суммы:

C=INT(RND(1)*(B–A+1)+A)

А теперь снова вернемся к рассмотрению вопроса быстродействия двух методов сортировки.

В обе программы введем дополнительные строки:

```
50 A(I)=INT(RND(1)*200+1)
60 PRINT A(I)
73 INPUT"ЗАМЕТЬТЕ И НАЖМИТЕ КЛАВИШУ (ВК),
ЧТОБЫ НАЧАТЬ СОРТИРОВКУ";S
```

В строке 50 интервал взят от 1 до 200. Чем длиннее список чисел (шире этот интервал), тем больше будет разница во времени между этими двумя сортировками.

Проверьте!

Проверьте свои знания ?

1. В чем состоит суть методов сортировки: метода пузырька, метода Шелла?
2. Почему программы сортировки с цифрами можно преобразовать в программы сортировки слов?
3. Каким способом, используя функцию RND, интервал (0,1) можно преобразовать в любой другой интервал?
4. Как работает функция INT?

Тренировка

Составьте программу упорядочением библиотеки, чтобы она отсортировывала и классифицировала книги и вы могли легко найти любую из них по разделу, имени автора, по названию.

Примечание: используйте ключи (коды) для разделов и подразделов, пользуясь библиотечной классификацией. Например:

DATA 212,ХЕМИНГУЭЙ, СТАРИК И МОРЕ.

Автор Название

Подраздел 2 (повести испанских и латиноамериканских авторов).

Раздел 21 (художественная литература).

§ 42. Программы анкеты

Сейчас стали весьма распространены самые разные анкеты и тесты. С помощью одних оценивают характер человека, отдельные черты его, предрасположенность к каким-то определенным действиям и видам работы, по другим пытаются "угадать" будущую профессию учащегося (тесты профориентации), наклонности, степень умственного развития (тесты Айзенка). Наконец, по тестам сдают выпускные и вступительные экзамены в школах и других учебных заведениях. Тестирование получило очень широ-

кое распространение при приеме человека на работу, при сдаче экзамена по правилам дорожного движения, при опросе общественного мнения и т.д.

Серьезные анкеты состоят из 500-600 вопросов, поэтому обрабатывать их вручную достаточно сложно, здесь нам на помощь приходит ЭВМ.

С чего начинается составление программы? Конечно, сначала нужно хорошо разобраться в задаче, в тех действиях и операциях, которые должна выполнять программа. Потом надо составить ее план, определить структуру – представить ее составные части.

Программа для проведения анкетного опроса должна состоять из трех частей: ввод данных (задается вопрос и вводится ответ), подсчет суммы баллов и вывод текста заключения).

Как задавать вопросы? Очевидно, с помощью оператора PRINT. А как вводить ответы опрашиваемого? (Последнего в анкетных исследованиях называют **респондентом**.) Обычно это делают с помощью оператора INPUT.

Организуем простенькую программу так, чтобы машина задавала респонденту вопросы, он отвечал на них, а она оценивала эти ответы.

Вопросы будем задавать по курсу физики 10 класса, из раздела "Термодинамика":

```
10 PRINT "НАЗЫВАЕТСЯ ЛИ ПРОЦЕСС ИЗОХОРНЫМ,
ЕСЛИ ТЕРМОДИНАМИЧЕСКАЯ СИСТЕМА ИЗМЕНЯЕТСЯ
ПРИ ПОСТОЯННОМ ОБЪЕМЕ";
20 INPUT U¤
30 IF U¤="ДА" THEN PRINT"ОТВЕТ ВЕРНЫЙ"
40 IF U¤="НЕТ" THEN PRINT"ОТВЕТ НЕВЕРНЫЙ"
```

Но эта программа не застрахована от других (кроме "ДА" и "НЕТ") ответов. Учащийся ведь может ответить как угодно: "НЕ ЗНАЮ", "НЕ МОЖЕТ БЫТЬ", "НАВЕРНОЕ" и т.п. или просто вместо заглавных букв "ДА" и "НЕТ" наберет строчные "да" и "нет" или какую угодно комбинацию "Да", "Нет", "дА", "нЕТ" и т.д. Для этого мы введем новые строки:

```
50 IF U¤<>"ДА" AND U¤ "НЕТ" THEN PRINT"ОТВЕЧАЙТЕ
ТОЛЬКО ЗАГЛАВНЫМИ БУКВАМИ (ДА) ИЛИ (НЕТ)"
55 GOTO 10
```

Обратите внимание, что здесь стоит оператор AND (И), а не OR (ИЛИ). Это означает, что оба условия должны быть истинны. В противном случае (если бы стояло OR), утверждения сравнения не будут истинны и компьютер при правильном ответе (заглавные буквы "ДА" и "НЕТ") все равно отпечатает PRINT в строке 50 (см. § 50).

Теперь нашу программу немного усложним. Пусть респонденту задают не один вопрос, а несколько (к примеру, три). Наша задача не просто оценить их правильность, а по совокупности правильных и неправильных ответов дать конечную оценку отвечающему:

```
10 S=0
20 PRINT"ВЕРНА ЛИ ЗАПИСЬ ПЕРВОГО ЗАКОНА
ТЕРМОДИНАМИКИ: ΔU=A-Q";
30 INPUT U1¤
40 IF U1¤="ДА" THEN PRINT "НЕПРАВИЛЬНО!?" ELSE 50
45 GOTO 60
50 IF U1¤="НЕТ" THEN S=S+1
55 PRINT"ПРАВИЛЬНО!"
60 PRINT"АДИАБАТНЫЙ ЛИ ПРОЦЕСС
В ТЕПЛОИЗОЛИРОВАННОЙ СИСТЕМЕ";
70 INPUT U2¤
80 IF U2¤="ДА" THEN S=S+1 ELSE 90
85 PRINT"ПРАВИЛЬНО!"
90 IF U2¤ ="НЕТ" THEN PRINT "НЕПРАВИЛЬНО!?"
100 PRINT "ВЕРНА ЛИ ЗАПИСЬ ПЕРВОГО
ЗАКОНА ТЕРМОДИНАМИКИ В ПРИМЕНЕНИИ
К ИЗОХОРНОМУ ПРОЦЕССУ:ΔU=A";
110 INPUT U3¤
120 IF U3¤="ДА" THEN PRINT"НЕПРАВИЛЬНО!" ELSE 130
125 GOTO 140
130 IF U3¤= "НЕТ" THEN S=S+1
135 PRINT"ПРАВИЛЬНО!"
140 IF S=3 THEN PRINT"ВЫ ОТВЕТИЛИ НА - 5!!! ТАК
ДЕРЖАТЬ!"
150 IF S<3 THEN PRINT"ВЫ ОТВЕТИЛИ НА - 2?? ПОЗОР!
НУЖНО УЧИТЬ МАТЕРИАЛ!"
160 END
```

В этой программе мы ввели переменную S. В начале (в строке 10) эта числовая переменная обнуляется, а затем

Известно ли вам, что...

Когда речь заходит о труде программиста, то высказываются разные противоречивые требования к этой профессии. Одни говорят, что у программистов действуют более строгие стандарты ясности и точности, чем у математиков, другие указывают на то, что в труде программиста сочетаются рафинированная математика с "грязной инженерной работой", третьи утверждают, что "программирование – это область ... искусства" (Д.Кнут).

А вот как раскрывает противоречия во внутреннем содержании деятельности этой профессии академик Андрей Петрович Ершов (инициатор обучения детей информатике в нашей стране): "Программист должен обладать способностью первоклассного математика к абстракции и логическому мышлению в сочетании с эдисоновским талантом сооружать все что угодно из нуля и единиц. Он должен сочетать

после каждого правильного ответа (см. строки 50, 80, 130) увеличивается на единицу. В строках 140 и 150 она сравнивается с эталоном (у нас это число три), и выставляется конечная оценка респонденту (в этой программе оценки "5" и "2").

Понятно, что если бы в программе было больше вопросов, то эталон сравнения был бы иным. Например, если бы вопросов было пять, то можно оценить знания учащегося по такой схеме:

Пять правильных ответов – оценка "5".

Четыре правильных ответа – оценка "4".

Три правильных ответа – оценка "3".

Два, один или ноль правильных ответов – оценка "2".

А теперь еще немного усовершенствуем нашу программу, используя для этого операторы связки: FOR... NEXT и READ... DATA:

```
10 S=0
20 FOR P=1 TO 5
30 READ G$,X
40 PRINT G$
50 T=X
60 INPUT U
70 IF U<>T THEN PRINT"ИЗВИНИТЕ,
ПРАВИЛЬНЫЙ ОТВЕТ -"; T ELSE 90
80 FOR J=1 TO 2000
85 NEXT J
87 GOTO 120
90 IF U=1 THEN PRINT"ВЫ
ОТВЕТИЛИ ВЕРНО!"
100 S=S+1
110 PRINT
120 FOR J=1 TO 2000
125 NEXT J
```

```
130 NEXT P
140 PRINT
150 IF S=5 THEN PRINT"ВАШИ
ЗНАНИЯ НА ОТЛИЧНО!!!"
160 IF S=4 THEN PRINT "ВАШИ
ЗНАНИЯ НА ХОРОШО!!"
170 IF S=3 THEN PRINT"ВАШИ
ЗНАНИЯ НА УДОВЛЕТВОРИТЕЛЬНО!?"
180 IF S=2 OR S=1 OR S=0 THEN
PRINT"ЗНАНИЯ НА ДВА??"
190 FOR J=1 TO 2000
195 NEXT J
200 PRINT"ДО СВИДАНИЯ! ДО
НОВЫХ ВСТРЕЧ!"
210 DATA"ВОПРОС I: 1-ОТВЕТ 2-
ОТВЕТ 3-ОТВЕТ",1
220 DATA"ВОПРОС II: 1-ОТВЕТ 2-
ОТВЕТ 3-ОТВЕТ",1
230 DATA"ВОПРОС III: 1-ОТВЕТ 2-
ОТВЕТ 3-ОТВЕТ",3
240 DATA"ВОПРОС IV: 1-ОТВЕТ 2-
ОТВЕТ 3-ОТВЕТ",2
250 DATA"ВОПРОС V: 1-ОТВЕТ 2-
ОТВЕТ 3-ОТВЕТ",3
260 END
```

Вопросы и ответы в этой программе обозначили условно:

> "ВОПРОС I: 1-ОТВЕТ 2-ОТВЕТ 3-ОТВЕТ"

При составлении рабочих программ в эти обозначения "ВОПРОС", "ОТВЕТ" вкладывают конкретный смысл.

Каждый раз, когда ответ на вопрос будет правильным, компьютер укажет на это, давая респонденту одно очко, и задаст следующий вопрос. Если же ответ будет неправильным, компьютер скажет об этом, даст номер правильно-

аккуратность бухгалтера с проницательностью разведчика, фантазию автора детективных романов с трезвой практичностью экономиста".

****

Чип **– это микросхема (БИС) без контактов, основа микропроцессоров. Чипы вырезаются из пластины монокристалла сверхчистого кремния после обработки ее по интегральной технологии.**

Биочип **– сверхминиатюрное устройство обработки и хранения информации на основе электронных процессов в биоорганических молекулярных системах или в системах, объединяющих кристаллические и биомолекулярные структуры. Как правило, используется в специализированных нейрокомпьютерах. Для них разрабатываются схемы, содержащие до 1000 млн микропроцессоров. Напомним, что человеческий мозг содержит около 10 млрд нейронов и в 1000 раз большее количество синапсов.**

го ответа и задаст следующий вопрос. В конце опроса компьютер оценит знания учащихся.

Эту программу можно немного упростить, используя подпрограммы. Строки 80 и 85, 120 и 125, 190 и 195 заменим на строки с операторами GOSUB 300, а сам цикл ожидания запишем в строке 300 в виде подпрограммы. Блок проверки правильности тоже можно организовать в виде подпрограммы. Окончательно программа будет выглядеть так:

```
10 S=0
20 FOR P=1 TO 5
30 READ G¤,X
40 PRINT G¤
50 T=X
60 GOSUB 400
80 GOSUB 300
130 NEXT P
140 PRINT
150 IF S=5 THEN PRINT"ВАШИ ЗНАНИЯ НА
ОТЛИЧНО!!!"
160 IF S=4 THEN PRINT"ВАШИ ЗНАНИЯ НА
ХОРОШО!!"
170 IF S=3 THEN PRINT"ВАШИ ЗНАНИЯ НА
УДОВЛЕТВОРИТЕЛЬНО!"
180 IF S=2 OR S=1 OR S=0 THEN PRINT"ЗНАНИЯ НА
ДВА??"
190 GOSUB 300
200 PRINT"ДО СВИДАНИЯ! ДО НОВЫХ ВСТРЕЧ!"
210 DATA"ПЕРЕДАЧУ ЭНЕРГИИ ОТ ОДНИХ ЧАСТИЦ К
ДРУГИМ НАЗЫВАЮТ...
1.ТЕПЛОПЕРЕДАЧЕЙ 2.ИЗЛУЧЕНИЕМ 3.КОНВЕКЦИЕЙ",1
220 DATA"ФОРМУЛА ТЕПЛОТЫ,ИДУЩЕЙ НА
ПАРООБРАЗОВАНИЕ (КОНДЕНСАЦИЮ):
1.Q=LM 2.Q=qM 3.Q=CM(T2-T1)",1
230 DATA"ФИЗИЧЕСКИЙ СМЫСЛ 'C' В ФОРМУЛЕ:
Q=CM(T2-T1):1.ТЕМПЕРАТУРА 2.ТЕПЛОТА 3.УДЕЛЬНАЯ
ТЕПЛОЕМКОСТЬ",3
240 DATA"ЕДИНИЦА УДЕЛЬНОЙ ТЕПЛОТЫ
ПЛАВЛЕНИЯ: 1.ДЖ 2.ДЖ/КГ 3.КГ",2
250 DATA"ЭНЕРГИЯ ТЕЛА ПРИ ПЕРЕВОДЕ ВЕЩЕСТВА
```

```
ИЗ ГАЗООБРАЗНОГО В ЖИДКОЕ... 1.УВЕЛИЧИВАЕТСЯ
2.ОСТАЕТСЯ ПОСТОЯННОЙ 3.УМЕНЬШАЕТСЯ",3
260 END
300 FOR J=1 TO 2000
310 NEXT J
320 RETURN
400 INPUT U
410 IF U<>T THEN PRINT"НЕВЕРНО! ПРАВИЛЬНЫМ
ОТВЕТОМ ЯВЛЯЕТСЯ:";X;"ОТВЕТ" ELSE 420
415 GOTO 450
420 IF U=T THEN PRINT "ВЕРНО!"
430 S=S+1
440 PRINT
450 RETURN
```

В этих программах коды правильных ответов:
I вопрос – 1 правильный;
II вопрос – 1 правильный;
III вопрос – 3 правильный;
IV вопрос – 2 правильный;
V вопрос – 3 правильный.

Для непредсказуемости работы контролирующих программ часто используют случайную выборку вопросов, т.е. случайное извлечение из блока данных символических величин. А мы знаем, что функция RND оперирует только числами. Как же перейти от случайных чисел к выбору случайных вопросов?

Эту задачу в языке Бейсик лучше всего решать с помощью операторов ON...GOTO и ON...GOSUB.

Используя первый из этих операторов, составим тестирующую программу, в которой вопросы будут выбираться самым случайным образом. Для простоты рассмотрения используем блок, состоящий из 8 исходных вопросов (причем записанных условно). Пользователю будет предложено лишь 5 случайных из них (в рабочих программах используют выборку из большого числа вопросов):

```
10 DIM A(5)
130 FOR I=1 TO 5
135 A(I)=INT(RND(1)*8+1)
140 ON A(I) GOTO 200,250,300,350,400,450,500,550
145 NEXT I
```

```
150 IF S=5 THEN PRINT "ВЫ ОТВЕТИЛИ НА 5"
160 IF S=4 THEN PRINT "ВЫ ОТВЕТИЛИ НА 4"
170 IF S=3 THEN PRINT "ВЫ ОТВЕТИЛИ НА 3"
180 IF S=2 OR S=1 OR S=0 THEN PRINT "ВЫ
ОТВЕТИЛИ НА 2"
190 END
200 PRINT"I-Й ВОПРОС"
202 GOSUB 2000
204 GOTO 145
250 PRINT"II-Й ВОПРОС"
252 GOSUB 2050
254 GOTO 145
300 PRINT"III-ВОПРОС"
302 GOSUB 2000
304 GOTO 145
350 PRINT"IV-Й ВОПРОС"
352 GOSUB 2000
354 GOTO 145
400 PRINT"V-Й ВОПРОС"
402 GOSUB 2050
404 GOTO 145
450 PRINTI"VI-Й ВОПРОС"
452 GOSUB 2050
454 GOTO 145
500 PRINT"VII-Й ВОПРОС"
502 GOSUB 2050
504 GOTO 145
550 PRINT"VIII-Й ВОПРОС"
552 GOSUB 2000
554 GOTO 145
2000 INPUT"ОТВЕЧАЙТЕ 'ДА' ИЛИ 'НЕТ'; V1¤
2010 IF V1¤="ДА" THEN PRINT"НЕПРАВИЛЬНО!"
2020 IF V1¤="НЕТ" THEN PRINT"ПРАВИЛЬНО!"
 ELSE 2040
2030 S=S+1
2040 RETURN
2050 INPUT"ОТВЕЧАЙТЕ 'ДА' ИЛИ 'НЕТ'; V2¤
2060 IF V2¤="ДА" THEN PRINT"ПРАВИЛЬНО!"
 ELSE 2080
2070 S=S+1
```

```
2080 IF U2$="НЕТ" THEN PRINT"НЕПРАВИЛЬНО"
 ELSE 2090
2090 RETURN
```

При запуске программы командой RUN даже самый неопытный пользователь заметит, что программа обладает существенным недостатком. Некоторые вопросы повторяются по нескольку раз. Избавиться от этого "недуга" можно, применяя нижеследующий прием. В данной программе исключим строку 135 и перепишем первые 145 строк:

```
10 DIM A(5)
20 A(1)=INT(RND(1)*8+1)
30 K=1
40 FOR I=2 TO 5
50 P=INT(RND(1)*8+1)
55 N=0
60 FOR J=1 TO K
70 IF P<>A(J) THEN 90
80 N=1
85 J=I
90 NEXT J
100 IF N=1 THEN 50
110 A(I)=P
115 K=K+1
120 NEXT I
130 FOR I=1 TO 5
140 ON A(I) GOTO 200,250,300,350,400,450,500,550
145 NEXT I
```

Новая программа гарантирует выдачу на экран любого из приведенных в блоке (строки 200-550) вопросов лишь один раз. Причем при каждом повторном запуске программы вопросы будут вводиться в другой последовательности. Это именно то, к чему мы стремились.

Программист, составляющий программы тестового характера, всегда сталкивается с задачей описания **эталона выполнения**, т.е. с определением оценки, необходимой для выполнения теста. Эту оценку всегда связывают с **коэффициентом условия К**, который определяется по формуле:

$$K=S/M,$$

где S – количество правильно выполненных операций;

M – общее количество операций в одном тесте.

Если в результате выполнения набора тестов $К \geq 0,6 \div 0,7$, то считается, что деятельность на данном уровне усвоена. Для пятибалльной системы К обычно принимает такие значения:

оценка "5" при $1 = К \geq 0,9$ (т.е. – 90% правильных)

оценка "4" при $0,9 > К \geq 0,7$ (90%-70%)

оценка "3" при $0,7 > К \geq 0,5$ (70%-50%)

оценка "2" при $0,5 > К = 0$ (50-0%)

при условии, что степень важности предлагаемых вопросов одинакова. Если степень (вес) важности вопросов различна, то существуют другие (более сложные) методики оценки ответов респондентов.

Тренировка

Составьте тест-программу для испытания своих товарищей на сообразительность. Используйте для этих целей тест, который полушутя, полусерьезно предлагают в Калифорнийском университете (США).

1. Профессор ложится спать в 8 часов вечера, а будильник заводит на 9 часов утра. Сколько будет спать профессор?

2. Может ли мужчина жениться на сестре своей вдовы?

3. Есть ли 7 ноября в Австралии?

4. У Мамеда 10 овец. Все, кроме 9, сдохли. Сколько осталось овец?

5. Вы – пилот самолета, летящего из Гаваны в Москву с двумя пересадками в Алжире. Сколько лет пилоту?

6. Обычно месяц заканчивается 30-м или 31-м числом. В каком месяце есть 28-е число?

7. Вы заходите в малознакомую комнату, которая затемнена. В ней есть две лампы – газовая и бензиновая. Что вы зажжете в первую очередь?

8. Один поезд идет из Киева в Донецк, а другой – из Донецка в Киев. Вышли они одновременно, но скорость первого в три раза больше скорости второго. Какой поезд будет дальше от Киева в момент встречи?

9. Отец с сыном попали в катастрофу. Отец скончался в госпитале. К сыну в палату заходит хирург и говорит, показывая на него: "Это мой сын". Могут ли эти слова быть правдой?

10. Археологи нашли монету, датированную 35-м годом до нашей эры. Возможно ли это?

11. Палку нужно распилить на 12 частей. Сколько потребуется распилов?

12. На руках – 10 пальцев. Сколько пальцев на 10 руках?

13. В каком количестве взял Ной зверей в свой ковчег?

14. Врач прописал больному три укола через каждые полчаса. Сколько потребуется времени, чтобы сделать все уколы?

15. Сколько цифр 9 в ряду чисел от "1" до "100"?

16. Одинокий сторож умер ночью. Дадут ли ему пенсию?

17. Горело 7 свечей. Три погасло. Сколько свечей осталось?

18. Кирпич весит один килограмм плюс еще полкирпича. Сколько весит кирпич?

19. Под каким кустом сидит заяц во время дождя?

Посчитайте число неправильных ответов и определите, кто вы:

0-1 – гений;
2-4 – интеллектуал;
5-6 – нормальный человек;
7-8 – рядовой идиот;
9-10 – рядовой идиот с побочными явлениями;
11-12 – абсолютный идиот;
13-15 – не способен мыслить;
16-19 – необходима изоляция от общества.

Правильные ответы: 1. Один час (будильник не разбирает, где утро, а где вечер). 2. Нет, так как вдова – та, у которой умер муж. 3. Есть. 4. Девять. 5. Вы – пилот самолета (пилоту столько лет, сколько вам). 6. Во всех. 7. Спичку. 8. Одинаково (в момент встречи они находятся в одной точке). 9. Да, если хирург – мать мальчика. 10. Нет, тогда на монетах еще не писали дату изготовления. 11. 11 распилов. 12. 50. 13. Каждой твари – по паре. 14. Один час. 15. 20. 16. Нет, он умер. 17. Три, остальные сгорели. 18. Один килограмм. 19. Под мокрым.

§ 43. Вопросы оптимизации

Принятие оптимального, т.е. наилучшего варианта при решении возникающих в жизни задач приходится делать почти ежедневно.

Представьте себе, что вы кассир и вам необходимо давать сдачу. Мы знаем, что в практике используются монеты (или банкноты) достоинством в 50, 20, 10, 5, 3, 2 и 1

копейки. Какие монеты вы должны взять, чтобы дать сдачу 79 копеек наименьшим количеством монет? Ответ может быть таким: 79=50+20+5+3+1.

Задача состоит в том, чтобы ЭВМ могла быстро подсказывать кассиру, какие монеты брать, если ему необходимо дать сдачу мелочью на сумму S< 100 копеек.

Текст программы может быть таким:

```
10 ' ПОЛУЧИТЕ СДАЧУ
20 INPUT"ВВЕДИТЕ ИСХОДНУЮ СУММУ S";S
30 PRINT"ПОЛУЧИТЕ СДАЧУ"
40 IF S>=50 THEN 45 ELSE 50
45 S=S-50
47 PRINT"50 КОП. 1 ШТ."
50 IF S>=20 THEN 55 ELSE 60
55 I=INT(S/20)
57 PRINT"20 КОП.";I;"ШТ."
59 S=S-I*20
60 IF S>=15 THEN 65 ELSE 70
65 S=S-15
67 PRINT"15 КОП. 1ШТ."
70 IF S>=10 THEN 75 ELSE 80
75 S=S-10
77 PRINT"10 КОП. 1ШТ."
80 IF S>=5 THEN 85 ELSE 90
85 S=S-5
87 PRINT"5 КОП. 1ШТ."
90 IF S>=3 THEN 95 ELSE 100
95 S=S-3
97 PRINT"3 КОП. - 1ШТ."
100 IF S>=2 THEN 105 ELSE 110
105 S=S-2
107 PRINT"2 КОП. 1 ШТ."
110 IF S>=0 THEN PRINT"1 КОП. - 1 ШТ."
```

На фабриках и заводах из листов железа, ткани, кожи, бумаги и других материалов вырезают различные заготовки для деталей. Очень важно, чтобы как можно меньше материала отправлялось в отходы.

Пример

Имеется квадратный лист бумаги со стороной a. Из листа делается коробка следующим образом: по уг-

лам листа вырезается четыре квадрата, чтобы коробка имела наибольшую вместимость. Решить задачу при a=6 см и a=18 см.

Для решения такой задачи обозначают через x сторону вырезаемых квадратиков. Тогда получающаяся коробка будет иметь квадратное основание со стороной $a-2x$ и высотой x. Следовательно, вместимость коробки (обозначим ее $f(x)$ будет равна:

$$f(x)=(a-2x)^2x=4x^3-4ax^2+a^2x.$$

Далее из необходимого условия максимума ($f'(x)=0$) получаем уравнение $12x^2-8ax+a^2=0$. Решаем полученное уравнение при соответствующих a и анализируем корни. Теперь несложно составить саму программу.

Сколько раз, придя в магазин или на почту и увидев там несколько очередей, вы становились в одну из них и обнаруживали, что именно эта очередь совершенно не сдвигается с места из-за какого-то недотепы, стоящего впереди?

Очереди нас раздражают: мы, несомненно, тратим страшно много времени на ожидание обслуживания.

Один из способов избежать нескольких очередей состоит в том, чтобы иметь только одну очередь, из которой клиенты поступают на обслуживание по принципу "первым пришел – первым обслужен". Эта система принята во многих банковских и почтовых отделениях.

Математическая теория, которая занимается оптимальным расчетом таких очередей, называется **"теорией массового обслуживания"**.

Но иногда приходится решать самостоятельно возникающие в жизни задачи. Например:

1. Владелец магазина может закупать некий товар в банках по 1,53 долл. за штуку. Коробка, содержащая 24 банки, обойдется ему в 36 долл. Необходимо написать программу на языке Бейсик, распечатывающую стоимость 24 банок поштучно и в коробке. Сравните эти две стоимости и выведите на дисплей способ лучшей закупки.

2. Комиссионерам, продающим ваш товар, вы платите месячные комиссионные в зависимости от объема продаж по следующей формуле. Если объем продажи меньше 50000 долл., то комиссионные составляют 32/2%. Если объем

50000 или больше, но меньше 100000, то комиссионные увеличиваются до 4%. А если объем достигает 100000 и более, то выплачивается 43/4% комиссионных. Нужно написать программу на языке Бейсик, выплачивающую комиссионные торговому агенту, продавшему товар на 52347.50 долл.

Попробуйте составить эти программы самостоятельно.

В ряде стран ученые работают над созданием так называемого **искусственного интеллекта**. Это, конечно, не искусственный разум. Сейчас нет ни возможностей, ни намерений создать интеллект, подобный человеческому. Речь идет о помощниках человека в разных, но весьма узких областях его умственной деятельности, и прежде всего в решении задач управления.

Американский ученый К.Шеннон предложил шахматную игру как модель для научных исследований. Но почему именно шахматы? Потому, что эта древняя игра, как и многие проблемы управления, строится на решении задачи переборного типа. Результаты такого исследования могут быть использованы для решения сложных практических задач на ЭВМ, связанных с перебором большого количества вариантов, например, в планировании, управлении.

Для точного решения такой задачи необходимо сформировать дерево перебора всех возможностей (в шахматах – вариантов). Для сложных задач полное дерево перебора может быть весьма большим, а иногда и бесконечно большим. Подсчитано, например, что число всех возможных расстановок фигур на шахматной доске достигает 10^{120}.

В качестве курьеза укажем, что с того момента, как человек обрел дар речи, все люди земли произнесли "всего" 10^{16} слов. Поэтому нет другого выхода, как искусственно уменьшить число возможностей, т. е. попытаться найти решение с помощью усеченного дерева перебора, ограничивая длину вариантов.

К.Шеннон предложил два метода: во-первых, строить "усеченное" дерево перебора, включая в него все возможности в пределах ограниченной длины вариантов, и, во-вторых, исключать из дерева перебора возможности, заведомо лишенные смысла. По первому методу (он сравнительно прост) действуют сильнейшие современные шах-

матные программы для ЭВМ, по второму (он сложен) – играют шахматные мастера. Еще не законченная история создания искусственного шахматиста проходит под знаком соревнования между двумя этими направлениями.

§ 44. Искусство решения задач на ЭВМ

С помощью компьютеров решаются различные задачи обработки информации. Можно выделить общие особенности их подготовки к решению на ЭВМ с учетом двух возможных ситуаций.

Первая ситуация, когда для решаемой задачи **имеется готовая программа** (или соответствующий пакет прикладных программ), вторая – когда таковой программы нет и все требуется начинать "с нуля".

Первая ситуация в решении задач достаточно простая (необходимо лишь ознакомиться с инструкцией по эксплуатации программы), а вот вторая, которая чаще всего встречается в жизни, требует специальных навыков и компьютерной культуры.

Вот такой ситуацией, когда возникают многообразные конкретные задачи и связанные с этим проблемы и потребности, мы и займемся.

Прежде чем вы начнете составлять план или сценарий, вам нужно иметь четкое представление о том, что именно должна делать ваша программа. Весьма полезно составить сначала краткое описание программы, предусмотрев по возможности все случаи, которые могут возникнуть в процессе решения задачи. ЭВМ не может сама "осмыслить" процесс решения задачи и полученные результаты. Машина является лишь инструментом для автоматизации трудоемких процессов.

Использование ЭВМ не уменьшает необходимость полного понимания сущности решаемой задачи. Поэтому под ее решением на компьютере обычно подразумевают нечто большее, чем та работа, которая непосредственно выполняется на ЭВМ.

Подготовка и решение задачи с помощью ЭВМ состоит из нескольких **этапов**. Последовательность указанных эта-

пов, выполненных творчески, определяет искусство подготовки и решения задач на ЭВМ.

□

Первый этап – **постановка задачи и ее содержательный анализ.** Задание определяет общий подход к решению задачи. В нем формируются условия задачи: 1. Что дано? 2. Что необходимо? 3. Какие данные допустимы? 4. Какие результаты и в каком виде должны быть получены?

Точность и четкость постановки задачи – половина успеха.

На этом этапе нужно провести содержательный анализ, направленный на уточнение цели решения задачи, ее смысловых компонентов, исходных данных. Затем – ответить на вопрос: при каких условиях возможно получение требуемых результатов, а при каких нет. И, наконец, важно определить, какие результаты будут считаться достоверными и правильными.

Если у нас нет четкого понимания, что должно быть конечным результатом, то можно получить не то, что предполагалось. Поэтому умение четко формулировать, что требуется, и является залогом успеха.

□□

Второй этап – **формализация задачи**, выбор метода математической модели ее решения. На этом этапе развернутое содержательное описание задачи заменяется свернутой формулой (уравнением, неравенством или другим соотношением), в которой смысловые компоненты обозначаются соответствующими символами. Иными словами, на этом этапе конкретно решается поставленная задача.

Но правильность результатов решения прежде всего зависит от правильности выбранного **метода решения.** Именно трудноуловимые ошибки в алгоритмах и программах возникают из-за ошибок в выбранном методе решения. Обоснованность методов – гарантия правильности создаваемых алгоритмов и программ.

Отметим, что при решении задач на ЭВМ наибольший интерес для пользователей представляют математические модели различных процессов.

□□□

Третий этап – **составление алгоритма** на основе выбранного метода.

Нужно отметить тесную взаимосвязь между составлением алгоритма и выбранным методом, т.к. алгоритм в большей степени определяется выбранным методом, хотя один и тот же метод в свою очередь может быть реализован при помощи различных алгоритмов.

При разработке алгоритма решения сложной задачи следует использовать метод структурного подхода (см. § 32), который включает в себя **метод пошаговой детализации**. Здесь нужно следить, чтобы на каждом шаге структура алгоритма оставалась простой и ясной. Следует максимально использовать существующие базовые или разработанные ранее алгоритмы для отдельных фрагментов (блоков) алгоритма.

□□□□

Четвертый этап – **составление программы**, описание алгоритма на языке программирования.

Собственно программирование – написание программ (при наличии алгоритмов решения задач) – это просто кодирование алгоритмов на выбранном языке программирования.

При составлении программ на языке Бейсик нужно следить за тем, чтобы переменные, являющиеся исходными данными, вводились оператором INPUT, а результаты выводились оператором PRINT. В программе нужно предусмотреть печать необходимых заголовков и пояснений к выводимым результатам, а также печать указаний для выполнения операторов ввода.

Работа по составлению программы на Бейсике не должна вызывать трудностей. Если вы испытываете трудности, значит плохо был выполнен этап разработки алгоритма.

□□□□□

Пятый этап – **тестирование и отладка программы** – это проверка правильности работы программы и исправление обнаруженных ошибок.

Транслятор в компьютере осуществляет автоматически перевод языка программы на язык машинных команд. В процессе трансляции выявляются синтаксические ошибки в исходном тексте (листинге), затем формируется объект-

Что может компьютер

Лучше меньше, да лучше

Ценность книг не принято измерять их весом. И все же солидные фолианты всегда занимают на полках почетное место.

Величина компьютера также не является показателем его возможностей, однако японская фирма Fujitsu с радостью сообщает, что с ее конвейеров сошла новая модель переносного компьютера Notebook PC размером с книгу, преодолевшая килограммовый барьер.

Несмотря на более чем скромный вес, малютка имеет характеристики стандарт-

ный модуль, который после редактирования преобразовывается в загрузочный модуль, готовый для выполнения в ЭВМ.

Для выполнения тестирования необходимо подготовить тесты. **Тест** – это специально подобранные исходные данные в совокупности с теми результатами, которые должна выдавать программа при обработке этих данных. Разработка тестов – трудоемкая работа, часто требующая выполнения ручных просчетов. При составлении тестов нужно стремиться обеспечить проверку всех ветвей программы.

□□□□□□

Шестой этап – **счет программы, анализ результатов.** При окончании отладки программы из нее следует удалить все отладочные средства. С помощью средств операционной системы программа записывается на языке машинных команд, чтобы не повторять процесс трансляции и редактирования многократно.

После окончания счета необходимо провести обработку и осмысление результатов счета. После этого этапа, возможно, потребуется пересмотр самого подхода к решению задачи и возврат к первому этапу для повторного выполнения всех этапов с учетом приобретенного опыта.

А теперь перейдем к общим рекомендациям по **технологии** подготовки и решения задач на ЭВМ с использованием языка Бейсик.

Нередко приходится слышать, что программисту, хорошо знающему основы программирования, не обязательно для каждой задачи составлять блок-схему алгоритма. Он может сразу составлять программу и отлаживать ее на ком-

пьютере. С такой точкой зрения можно согласиться лишь тогда, когда программист достаточно подготовлен. Однако во всех случаях алгоритм составляется если не в явном виде, то в виде умозрительной модели.

Чем сложнее и длиннее алгоритм, тем острее ощущается потребность в специальных методах разработки и организации работы.

1. Самым простым методом является метод составления **скелетного алгоритма** или **скелетной программы**. Скелетную программу можно сразу записать с клавиатуры или сначала написать на бумаге, если это удобнее. Как только скелетная программа начнет работать, можно начать писать каждую подпрограмму как отдельный модуль, а затем вставлять ее в программу и испытывать. Некоторые подпрограммы можно в свою очередь подразделять на модули, модули разбивать на отдельные сегменты и т.д. Конечная цель состоит в том, чтобы полученные в результате окончательного разбиения элементарные модули были достаточно малы и их можно было бы разрабатывать, программировать и отлаживать.

2. Старайтесь создавать программу **универсальной**, т.е. не зависящей от конкретного набора данных (см. свойства алгоритмов § 15). Если данные представлены в виде массивов, то в качестве исходных следует рассматривать не только сами данные, но и их количество, для того чтобы одна и та же программа могла быть использована для обработки массивов различных размеров.

Универсальная программа должна обрабатывать вырожденные случаи и печатать сообщения об ошибке. Например:

ного компьютера. Кроме того, "книга" имеет два дополнительных слота для расширения объема ОЗУ.

В верхнюю часть корпуса очень естественно встроен жидкокристаллический дисплей, который может отобразить 640 двухцветных точек в каждой из 400 горизонтальных линий.

Все это аппаратное многообразие конструкторы умудрились разместить в коробке толщиной с обычную видеокассету, которую можно полностью накрыть стандартным листом писчей бумаги. Ну а вес 990 г, как уже говорилось, стал особой гордостью разработчиков. И это несмотря на то, что в том же объеме разместилась стандартная версия довольно большой операционной системы MS-DOS ROM V.3.22, которая может работать от встроенной в компьютер батареи восемь часов подряд.

Известно ли вам, что...

НЕЙРОКОМПЬЮТЕР – это система нечисловой информационно-логической обработки данных, реализуемых на базе новых архитектурных принципов исследования структуры и процессов функционирования человеческого мозга, нейронных сетей низших типов животных, методов получения биологических проводников электрического тока, создания искусственных нейронных сетей в виде специализированных электронных схем, состоящих из электронных аналогов нейронов и синапсов.

Например, японская фирма Fujitsu изготовила новый нейронный суперкомпьютер с параллельной архитектурой, который может

"На ноль делить нельзя", "Вы вышли за предел допустимых значений" и т.д.

3. Вместо констант лучше **использовать переменные**. Если в программе используются константы, то при их изменении нужно изменять в исходной программе каждый оператор, содержащий прежнюю константу. Эта процедура отнимает много времени и часто вызывает ошибки.

4. В программе следует предусмотреть **контроль вводимых данных** (в частности, программа не должна выполняться, если данные выходят за пределы допустимого диапазона).

5. Программа должна иметь комментарии (операторы REM), позволяющие легко проследить за логической взаимосвязью и функциями отдельных ее частей.

При написании программы следует заботиться о ее структуре, чтобы программа была **удобочитаемой**. В частности, в программе должны быть хорошо видны циклы. Для этого операторы FOR и NEXT нужно размещать в различных строках, не содержащих других операторов.

6. Когда программа получается слишком длинной, становится довольно сложно проследить на всем протяжении такой программы все выполняемые ею действия и понять, как она работает. Дело упрощается, если выбрать разумный порядок расположения подпрограмм. Вот две рекомендации:

а) Сначала запишите все рабочие подпрограммы, а следом за ними – сервисные подпрограммы. Если же их перемешать, то будет гораздо труднее выделить основные части программы.

б) В пределах этих двух групп поместите вместе те подпрограммы, которые

выполняют аналогичные действия, или разместите подпрограммы в том порядке, в каком они выполняются компьютером.

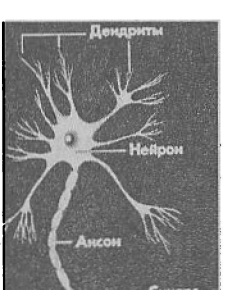
7. Если у вас имеется подпрограмма для решения задачи, которую нужно **исполнить очень быстро** (например, подпрограмма рисования для мультипликаций), поместите ее в самом начале программы. Вам потребуется поставить перед подпрограммой оператор GOTO, чтобы заставить компьютер обойти подпрограмму при первом проходе программы.

Объем программы колеблется от единиц до сотен строк. С некоторого момента размер программы начинает превышать объем динамической памяти. Такую программу необходимо разбивать на сегменты. Каждый сегмент в этом случае хранится на диске как самостоятельная программа, но выполняет лишь часть общей задачи. Когда текущий сегмент завершает работу, он инициирует загрузку очередного программного сегмента, предназначенного для выполнения следующего шага задачи, и передает управление этому новому сегменту. Описанная структура программы называется **оверлейной**, а отдельный программный сегмент – **оверлеем** (от анг. overley – перекрытие, наложение); каждый вновь поступающий сегмент как бы накладывается на сегмент, находящийся в данный момент в памяти. Для такой организации работы программы служит оператор CHAIN (см. описание его в инструкции к своему компьютеру).

Часто при тестировании, особенно для больших и сложных программ,

самообучаться (причем в 400 раз быстрее своего предшественника), осуществляет 500 миллионов связей между ячейками в секунду. Новая плата может научиться быть персональным компьютером всего за сутки, а освоить диалект сети NETtalk всего за 30 секунд.

В США тоже созданы микросхемы, специально предназначенные для использования в нейрокомпьютерах. Такая микросхема содержит на одном кристалле до 1000 микропроцессоров и формируется на биочипах (см. с.291).



очень трудно определить, где именно в программе возникает ошибка. Для локализации ошибок полезно использовать специальные вспомогательные средства, одним из которых является оператор TRON (см. описание в инструкции пользователя), приводящий в действие **трассировочную программу**. Последняя в процессе работы основной программы выводит номера ее строк в той последовательности, в какой они реально исполнялись. Цепочка этих номеров составляет след (или трассу – отсюда название) работы программы.

А теперь приведем **приемы экономии места** в памяти и **приемы увеличения скорости** выполнения программ.

Многие из них усложняют чтение и отладку программ, поэтому их следует применять только тогда, когда вы уже закончили программу и она выполняется свободно, без каких-либо ошибок. Сохраните копию этой программы на внешнем носителе памяти и, если возможно, распечатайте копию программы на бумаге для продолжения работы с ней.

Эти приемы можно применять, если, например, нужно добавить в программу побольше данных, а памяти не хватает, или нужно улучшить графику или несколько ускорить исполнение программы.

I. В большинстве версий Бейсика можно исключить отдельные служебные слова, например, LET, THEN, GOTO. Обратитесь к своему руководству пользователя, т.к. для разных диалектов языка эти правила различаются, а затем просмотрите свою программу:

```
1 IF X=N THEN PRINT "НЕТ"
2 IF Y=Z THEN GOTO 190
3 IF K=N THEN GOTO 195
4 LET B=5
5 IF M<C THEN LET S=S+1
```

II. Сократите (где это возможно) имена всех переменных с шести букв до одной:

```
ОТШЕТ; LIMET; FAMЮ; CELUH%
```

III. Удалите все операторы REM. Они не влияют на исполнение программы. Однако следует убедиться, что ни

один оператор GOSUB или GOTO не передает управление той строке, которую вы намерены удалить:

```
100 REM НАЧАЛО ПОДПРОГРАММЫ
110 PRINT
```

IV. Удалите все лишние пробелы, где это не вызывает ошибку. Каждый пробел занимает один байт памяти, поэтому можно сохранить довольно много места в памяти. Однако будьте осторожны, т.к. в некоторых операторах пробелы обеспечивают нужную четкость записи, например:

V. Переменную цикла можно не включать в операторе NEXT, компьютер автоматически восстановит пропущенную переменную в виде NEXT I:

```
10 FOR I=1 TO 10
20 PRINT I
30 NEXT
```

VI. Объединяйте в одну многооператорную строку как можно больше строк. Однако нельзя объединять в одной строке операторы IF... THEN или ON...GOSUB с другими операторами, которые не зависят от них. Проверьте значение максимальной длины строки для своего компьютера (обычно она составляет 255 символов).

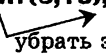
```
100 PRINT"СЧИТЫВАНИЕ
МАССИВА"
110 FOR I=1 TO 100
120 READ M¤
130 NEXT
140 PRINT
150 GOSUB 300
```

аналогично:

```
100 PRINT"СЧИТЫВАНИЕ МАССИВА":
FOR I=1 TO 100:READ M¤:NEXT:
PRINT:GOSUB 300
```

VII. Символ ";" где он не играет нужной "связующей" роли, можно убрать:

```
10 PRINT;"СИМВОЛ"
20 PRINT AT(5,10);"СМЕЩЕНИЕ"
```



VIII. В некоторых версиях Бейсика можно использовать сокращенную запись ключевых слов языка. Например, оператор PRINT можно заменить символом "?" (знак вопроса):

10 PRINT"ЗАМЕНА" ← аналогично → **10 ?"ЗАМЕНА"**

Однако сами по себе эти сокращения не экономят место в памяти, т. к. компьютер всегда хранит укороченные варианты ключевых слов.

IX. Для экономии памяти в массивах можно использовать элемент с нулевым индексом:

```
10 REM 5 ЭЛЕМЕНТОВ ДАННЫХ
20 DIM D¤(5)
30 FOR I=1 TO 5
40 READ D¤(I)
50 NEXT I
60...
```

аналогично

```
10 REM 5 ЭЛЕМЕНТОВ ДАННЫХ
20 DIM D¤(5)
30 FOR I=0 TO 4
40 READ D¤(I)
50 NEXT
60...
```

X. Вместо обычных числовых переменных используйте целые (в конце имени целой переменной ставится знак "%"). Они занимают гораздо меньше места в памяти, чем обычные действительные переменные, и намного быстрее обрабатываются компьютером:

ОБЫЧНЫЙ ЦИКЛ:

```
FOR K=1 TO 1000
PRINT K
NEXT
```

ЦИКЛ С ЦЕЛОЙ ПЕРЕМЕННОЙ:

```
FOR K%=1 TO 1000
PRINT K%
NEXT
```

XI. В качестве счетчика цикла всегда используйте целые переменные:

$$S\% = S\% + 1$$

Такие циклы используются вдвое быстрее.

❐

Приведем примеры **хранения данных**

В программах при считывании данных в массив они оказываются записанными в памяти (ОЗУ) дважды – один раз в виде строк DATA в области, где хранится вся программа, и второй раз в массиве в той области памяти, где хранятся все массивы и переменные. Именно поэтому программы, имеющие большой объем данных (например, игры), занимают много места в оперативной памяти.

Вместо размещения данных в программе в строках DATA можно хранить их на магнитном диске и читать оттуда прямо в массив, когда программа запущена на выполнение.

Вам следует узнать, какие операторы используются в вашем компьютере для загрузки в память и сохранения данных. Они различны у каждого компьютера, хотя обычно похожи на операторы, приведенные в схеме:

```
OPEN ПОИМЕНОВАННЫЙ
ФАЙЛ
FOR I=1 TO ЧИСЛО
ЭЛЕМЕНТОВ ДАННЫХ
PRINT # МАССИВ(I)
NEXT I
ЗАКРЫТЬ ФАЙЛ
```

Цикл чтения данных из массива МАССИВ в память компьютера и записи их на внешний носитель (магнитный диск)

PRINT # – оператор записи файла данных на магнитный диск. Прежде чем записать данные на диск, вы должны открыть файл (оператор OPEN), а когда закончите запись, нужно закрыть этот файл (оператор CLOSE).

```
OPEN ПОИМЕНОВАННЫЙ
ФАЙЛ
FOR I=1 TO ЧИСЛО
ЭЛЕМЕНТОВ ДАННЫХ
INPUT # МАССИВ(I)
NEXT I
ЗАКРЫТЬ ФАЙЛ
```

Цикл чтения записанных на диске данных в массив МАССИВ в компьютерной памяти

INPUT # – оператор загрузки файла данных в оперативную память с диска. Как в случае сохранения данных, нужно открыть файл перед загрузкой данных, а потом закрыть его.

Приведем схему создания **файла данных**:

1. В подпрограмме инициализации задайте размер массива и добавьте строки чтения из файла данных X. Запи-

шите программу на внешний носитель (на магнитный диск).

2. Наберите команду NEW, чтобы стереть программу из памяти (ОЗУ) компьютера.

3. Напишите подпрограмму, как показано выше, чтобы записать данные в файл.

4. Задайте размер массива для размещения данных.

5. Читайте данные в массив.

6. Запишите данные в файл X.

7. Исполните эту программу.

А теперь покажем все вышесказанное на конкретном примере:

Создадим программу инициализации:

```
1000 DIM F¤(3)
1010 FOR J=1 TO 3
1020 READ F¤(J)
1030 PRINT F¤(J)
1040 NEXT J
1050 DATA ШЕВЧЕНКО,МАРЧУК,КОСТЕНКО
1060 RETURN
```

Создадим программу чтения данных из массива в память компьютера (ОЗУ) и записи их на магнитный диск:

```
10 GOSUB 1000
20 OPEN"FAM" FOR OUTPUT: 'ПОИМЕНОВАННЫЙ ФАЙЛ
30 FOR J=1 TO 3
40 PRINT # F¤(J)
50 NEXT J
60 CLOSE: 'ЗАКРЫТЬ ФАЙЛ
```

Дадим команду NEW и очистим ОЗУ ЭВМ.

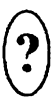
В основной программе создаем блок считывания данных с магнитного диска:

```
100 OPEN"FAM" FOR INPUT: 'ПОИМЕНОВАНЫЙ ФАЙЛ
110 FOR J=1 TO 3
120 INPUT # F¤(J)
125 PRINT F¤(J)
130 NEXT J
140 CLOSE: 'ЗАКРЫТЬ ФАЙЛ
```

Вот те шаги, следуя которым можно организовать хранение данных для программы на магнитном диске, а не в

строках DATA в самой программе. После выполнения этих шагов вы получите на диске копию своей программы вместе с файлами данных, содержащую все данные для этой программы. Когда вы загрузите в память и запустите программу, данные будут автоматически загружены в массив.

Проверьте свои знания



1. Перечислите этапы подготовки и решения задач на ЭВМ. Дайте им краткую характеристику.
2. Приведите рекомендации по технологии подготовки и решения задач на ЭВМ.
3. Что вы понимаете под оверлейной структурой организации программ?
4. Что такое трассировочная программа?
5. Какие вы знаете приемы экономии места в памяти ЭВМ? Увеличения скорости выполнения программ
6. Расскажите методику организации хранения данных для программ на магнитном диске.

ПОДВЕДЕМ ИТОГИ

Программирование есть процесс записи готового разработанного, обдуманного и проверенного алгоритма в виде текста программы.

Программа на языке Бейсик – последовательность операторов, записанных в пронумерованных строках.

Нельзя продвигаться дальше в изучении языка Бейсик (и не только его), если нет уверенности в том, что все рассмотренное раннее изучено и закреплено. Язык программирования обладает малой избыточностью, в нем нет ничего лишнего (в основных средствах), он обладает повышенной, образно говоря, "чувствительностью к синтаксическим ошибкам".

Основные **структуры алгоритмов** – это ограниченный набор стандартных способов соединения блоков для выполнения типичных последовательностей действий:

СЛЕДОВАНИЕ
РАЗВИЛКА (ПОЛНАЯ И НЕПОЛНАЯ)
ЦИКЛ (ДО И ПОСЛЕ)

Системные команды не заносятся в память ЭВМ и выполняются сразу после ввода:

RUN – выполнять программу;
LIST – просмотреть текст программы;
DELETE – удалить программные строки;
NEW – уничтожить весь текст программы;
SAVE – записать текст программы на внешний носитель;
LOAD – загрузить программу с внешнего носителя.

Оператор – это конкретное указание машине, оформленное либо в виде математической формулы, либо в виде обозначения какого-то действия, для выполнения которого требуется выполнить целый ряд машинных операций.

LET – оператор присваивания значений числовым и символьным переменным. Служебное слово LET, как правило, можно опускать.

PRINT – оператор печати на экран дисплея (или принтера) результатов вычислений или обработки символьной информации.

Информация может выводиться в **зонном** или **плотном** формате. Оператор PRINT используется совместно с различными функциями (AT, TAB, SPC и т.д.).

GOTO – оператор перехода на какой-то номер строки в программе.

Если таких переходов (скачков), когда один оператор (GOTO) обходит другой, много, то в совокупности они создают большие трудности в чтении программы. Старайтесь как можно меньше использовать этот оператор в программе во избежание как алгоритмических, так и логических ошибок.

IF ... THEN – условные операторы, служат для изменения порядка выполнения операторов в зависимости от какого-либо условия.

Можно сказать, что оператор IF сравнивает данные и выполняет (THEN) в зависимости от результата.

Располагая операторами LET, PRINT, GOTO, IF, можно записать программу, соответствующую любому алгоритму. Это не означает, что изучаемые далее средства языка

являются второстепенными – нет, это важные средства: с их помощью программы записываются кратко, удобно для обозначения; обеспечивается оперативное взаимодействие человека, решающего задачу, и ЭВМ.

FOR – NEXT – операторы цикла. Их по-другому называют **контролируемый цикл**. При их запуске компьютер выполняет задание определенное количество раз.

INPUT – оператор ввода информации с клавиатуры (во время выполнения программы). Этот оператор в Бейсике позволяет реализовать диалоговый режим работы компьютера с человеком.

READ-DATA – операторы чтения данных внутри программы.

Использование операторов READ-DATA позволяет формировать самые разнообразные массивы из исходных чисел и слов. От подготовки данных часто во многом зависит простота решения задачи.

☝ **Массив** – совокупность величин, при которой доступ к любой из них обеспечивается заданием имени массива и соответствующего для этой величины значения индекса (указателя), определяющего положение величины в массиве. Иногда массив называют **таблицей**.

DIM – оператор описания массивов. Он относится к неисполняемым операторам и служит для резервирования места в оперативной памяти компьютера.

GOSUB-RETURN – операторы вызова подпрограммы и возврата из нее в основную программу. По-другому эти операторы называют операторами "передачи управления с возвратом".

☝ **Подпрограммой** называют обособленную группу операторов, которую можно выполнять, многократно обращаясь к ней из различных мест программы.

Итак, мы представили вам основную часть средств языка Бейсик, т. е. только тот минимум, который вам необходим для усвоения азов программирования. Изучив дополнительные средства (а мы ни словом не обмолвились ни о матричных операторах, ни об операторах графики, ни об операторах работы с памятью и т.д.), вы получите мощное "оружие" для решения различного рода возникающих в жизни задач.

Сейчас многие работают с языком Бейсик, но зачастую стесняются этого. Вы же, пройдя наш курс "начал" языка и перейдя на современные версии (а сейчас в ходу версии BASIC с приставками GW, Q, QUISUAL, VISUAL, TURBO и др.), с гордостью будете произносить его имя и слова благодарности прародителям языка Бейсик – сотрудникам Дармутского колледжа (США) Дж.Кемени и Т.Куртцу.

Известно ли вам, что...

Что такое Notebook-компьютер?

Это название все чаще переводится на русский язык как "записная книжка", хотя размеры большинства представителей этого класса больше соответствуют термину "блокнот".

К таким компьютерам обычно относят портативные переносные станции, помещающиеся в обычном кейсе типа "дипломат", хотя многие известные фирмы выпускают такие модели, которые помещаются в дамской сумочке, имеющие размеры обычной книги или даже маленького калькулятора, но умеют делать многое из того, что делает настольный персональный компьютер, требующий сетевого питания (см. с. 304).

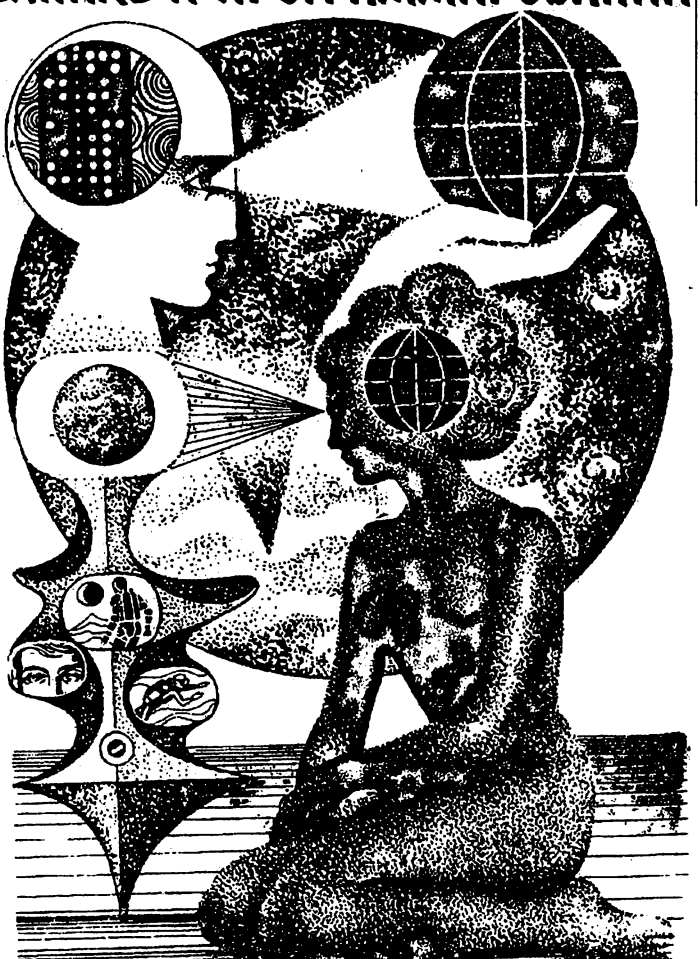
Дисплей такой ЭВМ обычно жидкокристаллический. В комплект может входит принтер, радиомодем с рацией, позволяющий включаться в информационную систему.

Такая мобильность позволяет работать с ним не только на работе, но и в гостях, а главное — в дороге. Он выполняет множество функций: персонального компьютера, программируемого телефона, телефакса, языкового процессора, электронного калькулятора, часов. Японские фирмы "Канон" и "Шарп" выпускают notebook с рядом дополнительных встроенных функций. Телефонный справочник, расписание движения пригородного транспорта, календарь, электронный секретарь, который вовремя напомнит о деловых встречах и поездках, – вот их неполный перечень. Кроме того, терминал можно использовать и для запоминания текстовой и числовой информации, что дает возможность поработать в дороге над статьей или путевыми заметками. С помощью дополнительных разъемов можно к нему подключать блоки, превращающие его либо в энциклопедический словарь, либо в один из словарей иностранных языков, либо в игровой персональный компьютер. Никаких ограничений здесь нет: очередная карта – и у вас в руках помощник в планировании семейного бюджета, еще одна – и компьютер превратится в инструктор по работе с самим собой.

Ну а, вернувшись в офис, можно с помощью стандартного интерфейса подключить автономный терминал к "большому компьютеру" и передать в него накопившуюся за день информацию для дальнейшей обработки.

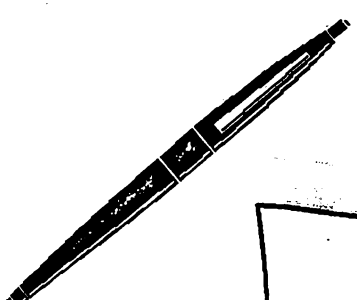
Глава VI

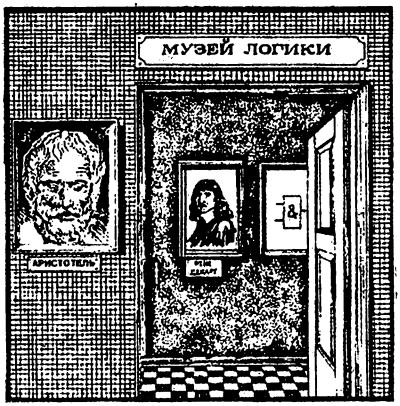
ЛОГИКА В ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКЕ И ПРОГРАММИРОВАНИИ



"Где начало того конца, которым оканчивается начало?"

Козьма Прутков





Если теория алгоритмов – в некотором смысле мать современных ЭВМ и программирования, то логика – их отец.

Умение рассуждать, логически мыслить, давать ответы на поставленные вопросы играет очень важную роль в жизни человека. Выделение логических задач носит до некоторой степени условный характер. Трудно определить, какую задачу следует назвать логической. Кажется, любая задача является таковой, так как для ее решения требуются определенные логические рассуждения. И это верно, но все же по традиции для тренировки именно логического мышления человеком придумано множество задач, в которых речь идет об объектах, вообще говоря, произвольной природы. Именно такими задачами и правилами их решения мы и займемся в этой главе.

Но какое отношение логика имеет к вычислительной технике и программированию? Оказывается, самое непосредственное. Именно логика является теоретической основой современных ЭВМ и сложных управляющих систем. Она приобретает важное прикладное значение – особенно в области разработки специальных языков для баз данных и представления знаний. Используя методы и средства логической науки, ученые разрабатывают эффективные языки программирования.

Например, основой так называемого **доказательного программирования** является формальная логика. Общая

идея здесь, как говорится, лежит на поверхности: если можно конструктивно, используя интуицию, доказать, что существуют объекты, удовлетворяющие некоторому данному условию, то, построив доказательство, можно построить по нему и программу вычисления соответствующего условия (функции).

Опять же, в основе так называемого **логического программирования** лежат структуры логических доказательств.

Но особое значение логическая наука стала приобретать в вопросах, касающихся проблемы **искусственного интеллекта.** Именно здесь разработчикам пришлось создать новую область логических исследований – **логический анализ.** Попытаемся очертить лишь некоторые контуры этого нового, перспективного, развивающегося направления.

Искусственный интеллект предполагает различные типы рассуждений. Они могут быть обычными или монотонными, но обязательно предполагают добавление новой информации. Логический анализ предполагает, что новая информация не отменяет, не делает неверными следствия, полученные ранее. Однако в практике рассуждений мы нередко допускаем некоторую полноту исходной информации. Добавление новой информации к исходной отменяет это допущение, и то, что ранее принималось как следствие, может не быть таковым при дополнительной информации. Такой логический анализ называют **очерчиванием.** Читая эти строки, вы наверняка ничего не поняли. Но мы привели это объяснение из области проблемы искусственного интеллекта умышленно, чтобы читатель почувствовал, что проблема эта достаточно трудная. В ней много нерешенных вопросов, которые предстоит разработать в будущем.

§ 45. Логика и математика

ЛОГИКА – это наука правильно рассуждать, наука о формах и законах человеческого мышления.

Логика – наука древняя. Ее основоположником считают древнегреческого мыслителя Аристотеля, жившего в

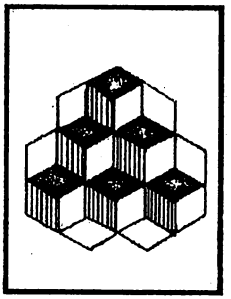
384-322 годах до н.э. Именно он подверг анализу человеческое мышление, такие его формы, как **понятие, суждение, умозаключение,** и рассмотрел мышление со стороны строения, структуры, то есть с формальной стороны. Так возникла **формальная логика** – наука, пытавшаяся найти ответ на вопрос, как мы рассуждаем, изучающая логические операции и правила мышления.

Ко времени зарождения логики математика уже прошла значительный путь развития. В течение многих веков логика помогала математике стать строгой, последовательной наукой. Постепенно взаимная связь между математикой и логикой привела к тому, что логика оказалась под влиянием математики.

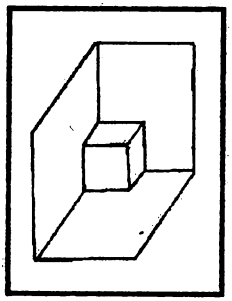
После падения античной цивилизации развитие математики, и особенно логики, замедлилось, потому что новые логические идеи нередко вступали в противоречие с формами мышления церкви. Любопытно отметить: первое, что было восстановлено из античной науки, – это именно логика Аристотеля.

Если обратиться к эпохе Возрождения, к истокам науки нового времени, нетрудно установить, что и в этом случае первыми восстанавливались и использовались именно разработанные в античности логические методы. С этого начиналась философия и математика Рене Декарта (1596-1650). Он считал, что человеческий разум может постигнуть истину, если будет исходить из достоверных положений, сводить сложные идеи к простым, переходить от известного и доказанного к неизвестному, избегая каких-либо пропусков в логических звеньях исследований. Фактически Декарт рекомендовал науке о мышлении – логике – руководствоваться общепринятыми в математике принципами.

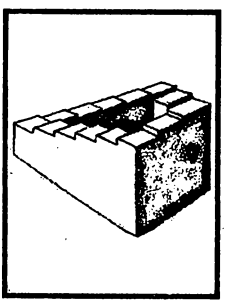
Кто изображен на портрете: прекрасная незнакомка или старая ведьма?



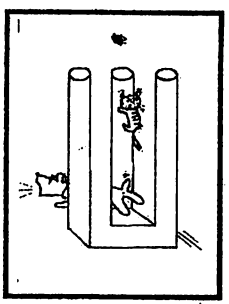
Сколько, по-вашему, здесь кубиков: 6 или 7?



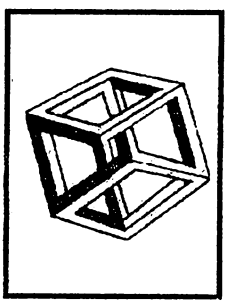
Что вы здесь видите: куб, стоящий в углу комнаты, куб, прикрепленный извне к большому блоку, или выемку в форме куба в большом блоке?



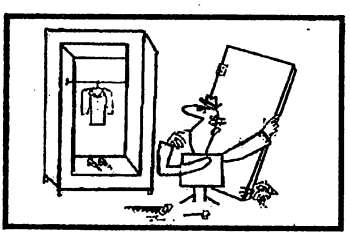
Лестница ведет вверх или вниз?



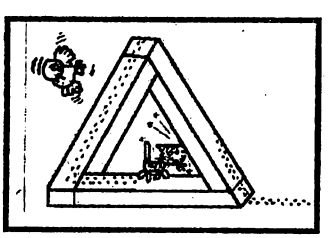
Это проем или выступ?



Не куб! Куб!



Вот так дверь!



По какой стороне идет кот?

Основоположником **математической логики** считают великого немецкого математика и философа Вильгельма Лейбница. Это он в XVII веке пытался построить первые логические исчисления: арифметические и буквенно-алгебраические. Это он впервые высказал мысль о возможности применения двоичной системы счисления в вычислительной математике.

Но этим идеям Лейбница суждено было получить дальнейшее развитие лишь в середине XIX века в трудах другого великого математика Джорджа Буля, отца писательницы Э.Войнич – автора романа "Овод". Он вывел для логических построений особую алгебру (алгебру логики). В отличие от обычной, в ней символами обозначают **не числа, а высказывания.**

ГЛАВНАЯ ЗАДАЧА ЛОГИКИ состоит в том, чтобы ВЫЯВИТЬ, какие способы рассуждения правильные, а какие нет.

Для иллюстрации сказанного приведем задачу, в которой используется неправильный способ рассуждения.

Трое фермеров, приехав в районный центр, решили сообща пообедать. Когда они закончили с обедом, буфетчица сказала, что с них причитается 30 долларов.

Каждый из обедавших достал 10 долларов, и они рассчитались. Когда фермеры были уже у выхода, буфетчица сообразила, что обсчитала их на 5 долларов. Тогда она позвала своего сынишку, который вертелся тут же, и сказала:

– Видишь тех трех мужчин? Догони их и верни им эти деньги. – И дала ему три бумажки по 1 доллару и одну двухдолларовую.

Смышленый парнишка, наблюдавший, как фермеры расплачивались, быстро смекнул, что они никак не смогут поровну поделить на троих 5 долларов. Он отдал им только три по доллару, а двухдолларовую оставил себе. Фермеры разделили 3 доллара – каждому по доллару – и подсчитали, что обед им обошелся по 9 долларов с брата, а всего, следовательно, они израсходовали 27 долларов. Кроме того, как мы знаем, 2 доллара осталось у мальчика. Всего получается 29 долларов. Но ведь они отдали буфетчице 30 долларов.

Куда пропал 1 доллар?

Мы предлагаем читателю самостоятельно разобраться в предложенной задаче. Приведем еще примеры.

Существует предание, что Александрийскую библиотеку сжег калиф Омар. Но прежде чем сжечь ее, он обосновал свое деяние с помощью следующего рассуждения: "*Если ваши книги согласны с кораном, то они вредны. Но вредные или излишние книги следует уничтожить. Значит, ваши книги следует уничтожить*".

Рассуждение есть переход от некоторых предложений, утверждений, называемых **посылками**, к утверждению, которое называется **умозаключением**. В приведенном примере первые три предложения являются посылками, четвертое – умозаключением.

Допустим, нас не интересует ни то, что в действительности не Омар сжег библиотеку, ни то, что он вообще так не рассуждал, ни истинность или ложность самого по себе заключения. Но зададимся вопросом, правильно ли, независимо от всего этого, само рассуждение. Как будет видно из дальнейшего, такое рассуждение с логической точки зрения совершенно правильно. Другое дело, что не истинны посылки, которые в нем используются. Но в данном случае нас интересует только правильность самого рассуждения, независимо от истинности посылок и заключения.

Возьмем еще пример. Предположим, некто строит следующее рассуждение: "*Дикари раскрашивают свое тело. Некоторые современные женщины раскрашивают свое тело. Следовательно, некоторые современные женщины – дикари*". Правильно ли это рассуждение?

Нетрудно установить, что данное рассуждение неправильно, несмотря на то, что используемые посылки и сделанное из них заключение можно признать истинными.

Итак, задача логики – описать и исследовать те способы рассуждений, которые являются правильными.

Проверьте свои знания ?

1. Какую роль играют знания логики в вычислительной технике и программировании? Где она имеет прямое приложение?

2. Что такое логика? Какими формами человеческого мышления она занимается?

3. Приведите краткую историю развития математической логики.

4. Какова главная задача логики?

§ 46. Понятие, суждение и умозаключение

Логика рассматривает три различные формы, в которых осуществляется мышление: понятие, суждение и умозаключение.

ПОНЯТИЕ – мысль, в которой обобщаются и выделяются предметы некоторого класса по определенным общим и в совокупности специфическим для них признакам.

В понятии "схватывается" сущность предметов, их внутреннее содержание.

СУЖДЕНИЕМ (ВЫСКАЗЫВАНИЕМ) называется всякое утверждение (или всякое предложение), о котором можно судить, истинно оно или ложно.

"*Этот вписанный угол, опирающийся на диаметр*" есть понятие; если мы говорим не обр всех углах, то понятие единичное. "*Все вписанные углы, опирающиеся на диаметр – прямые*" – это уже суждение, поскольку в нем высказывается, каковы свойства объекта суждения.

Если из двух суждений выводится третье, то этот процесс называется **умозаключением.**

Возьмем первое суждение:

"*Академик Ершов русифицировал язык Паскаль*".

Второе суждение:

"*Язык Паскаль – структурный язык*".

Тогда вывод из этих суждений:

"*Академик Ершов русифицировал структурный язык*" – будет умозаключением.

Поговорим более подробно о суждениях.

Почта, телефон, спутник связи...

Средства связи не напрасно во все времена и во всех странах привлекали внимание революционеров. Но приходит время, когда привычный ряд "почта, телефон, телеграф" следует продолжить словами "... и 30 спутников связи". Именно такого количества низкоорбитальных спутников достаточно, чтобы в поле зрения любого потенциального абонента, находящегося на поверхности Земли, всегда была хотя бы одна приемопередающая антенна.

Каждая космическая антенна позволяет связаться абонентским ком-

Будем обозначать суждения большими буквами латинского алфавита: A, B, C, D, ...

Можно интересоваться расположением слов в суждении, его звучанием, наличием определенного ритма в нем и т.д. Алгебра логики этими вещами не интересуется. Единственной существенной для нас характеристикой каждого высказывания есть **истинность** или **ложность**; эту характеристику назовем **значением истинности** (или истинным значением) данного суждения.

Условимся значение истинности суждения обозначать числом 1, если суждение истинно, и числом 0, если суждение ложно. Будем истинное суждение приравнивать числу 1, а ложное – числу 0. Сразу оговоримся, что символы 1 и 0 по написанию совпадают с обычными арифметическими единицей и нулем только внешне. Смысл, который вкладывается в эти обозначения, совсем иной. Логическая 1 не есть одна штука чего-то реального или абстрактного, это знак того, что совершилось какое-то событие, причем событие истинное, а логический 0 – что совершилось событие, причем событие ложное. Например, суждение "*Киев – столица Украины*" – истинное. Если его обозначить буквой A, то можно записать:

$$A = 1.$$

Суждение "*Высота гор на Земле превосходит 15 км*" – ложное. Если его обозначить буквой D, то можно записать:

$$D = 0.$$

Те утверждения (или те предложения), о которых нельзя сказать, истинны они или ложны, не являются суждениями. Например, утверждения: "*Эта*

книга – информатика", "*Метеорологический прогноз*" – не являются суждениями. Не будут суждениями предложения вопросительные и восклицательные: "*Как мелодичны вы, песни, Украины!*", "*Был ли казак Сагайдачный гетьманом?*".

Суждениями не будут и утверждения вида: "5+X=12", "X+Z<1", "*Число* Y *кратно* 3" и др. Ведь мы не знаем, чему равны значения X, Y, Z. Поэтому не можем знать, являются ли эти выражения достоверными, а раз нам это неизвестно, они не могут быть суждениями.

Такие выражения о переменных (объектах) называют **предикатами**.

Предикаты становятся суждениями, если переменной (или переменным — если их несколько) придать некоторое числовое значение (конечно, из области допустимых значений) или применить логическую операцию, которая устанавливает область истинности (ее называют **квантор**):

∀X (читается "для всех X"), ∃X (читается "существуют такие X" или "для некоторых X").

Например:

"5+X=12, если X=7".

"*Число* Y *кратно 3, когда сумма цифр числа* Y *делится на 3 без остатка*".

"(∃X) (5+X=12)" (читается: существует такое значение X, что 5+ X=12).

Суждения подразделяются на общие и частные.

ЧАСТНЫЕ суждения выражают конкретные (частные) факты.

Примеры частных суждений: "7–2>3", "*Луна – спутник Земли*", "*Этот четырехугольник ромб*".

пьютерам, находящимся на расстоянии до 6 тыс. км друг от друга. Компьютеры могут быть стационарные, переносные и даже встроенные в радиотелефон. В последнем случае на процессор возлагается только задача установления связи, а диалог будет происходить с помощью обычного телефона.

С помощью спутников, облетающих земной шар за полтора часа, сообщение можно послать и на более далекое расстояние по ходу их движения. Правда, скорость доставки вашего оцифрованного послания будет равна скорости спутника, но это тоже, согласитесь, не очень медленно. А если учесть, что прием и передача адресату вашей электронной посылки на спутнике полностью автоматизированы, то единственным неудобством такой системы связи является невозможность передать с ее помощью что-то материальное.

ОБЩИЕ суждения характеризуют свойства групп объектов или явлений.

Примеры общих суждений: "*В любом прямоугольном треугольнике есть угол в 90°*", "$x^2>0$", "*Всякий человек – млекопитающее*".

Может оказаться, что два суждения А и В одновременно истинны или одновременно ложны; тогда назовем их **равносильными (эквивалентными)** и условимся писать:

$$A \equiv B.$$

Эту запись читают так: "Суждение А эквивалентно суждению В", или "А тогда и только тогда, когда В", или "А необходимо и достаточно для В".

Например, суждения:

А= "*этот треугольник равносторонний*";

В= "*этот треугольник равноугольный*" –

будут равносильными, так что А≡В.

В программировании логическую эквивалентность обозначают символами "EQV". Таблица истинности для эквивалентности такова:

Таблица 10

A	B	A≡B	A	B	A≡B
1	1	1	0	1	0
1	0	0	0	0	1

Различают суждения простые и сложные.

Суждение считается ПРОСТЫМ, если никакая его часть не является суждением.

СЛОЖНЫЕ суждения характеризуются тем, что образованы из нескольких суждений с помощью определенных способов соединения суждений; простые суждения этим свойством не обладают.

Например, суждение: "*Париж – столица Албании*" – простое. А суждение "*Неверно, что Париж – столица Албании*" – сложное, потому что его часть является тоже суждением.

Сложные суждения чаще всего образуются как **составные**. Они получаются из простых или элементарных суждений с использованием связок "и", "или", "если..., то...", "не". Более подробно о них мы поговорим ниже.

Проверьте свои знания

1. Что мы называем понятием, суждением, умозаключением? Приведите примеры суждений.
2. Как из двух суждений получить умозаключение?
3. Как обозначают значение истинности суждений?
4. Какие утверждения нельзя отнести к суждениям?
5. Какие два суждения называют равносильными?
6. Какие суждения являются общими? Приведите примеры частных и общих суждений.
7. Что такое простое и сложное суждения? Приведите пример, как из простых суждений образовать сложное.

Тренировка

I. Какие из перечисленных ниже предложений являются суждениями и каково значение их истинности:

1) "сижу и смотрю";
2) "сумма внутренних углов треугольника равна двум прямым углам";
3) "верно ли, что $\pi=3,1415926...$?";
4) "$44>88$";
5) "математическое доказательство";
6) "существует такое значение x, что $2x^2-5x+3=0$;
7) "не лізь по перед батька в пекло!";
8) "для $\forall x$ выражение $x^2\geq 0$";
9) "$z+5=45$";
10) "$20+30+40+10=1000$"?

II. Из представленных двух суждений получите третье в виде умозаключения:

A="Если сумма цифр трехзначного числа равна 7";
B = "Цифры десятков и единиц одинаковы".

III. Укажите, какие из суждений являются частными, а какие общими:

1) $(x+y)(x-y)=x^2-y^2$;
2) "Любой ромб является параллелограммом";

3) "$a^3=a^2$, если $a=1$";

4) "Крышку уличного люка делают круглой, а не квадратной потому, что она не может соскользнуть в люк, если поставить ее на ребро";

5) $3^2 + 4^2 = 5^2$;

6) Если $|A| = |B|$, то $A=B$;

7) "Квадрат любого четного числа делится на 4";

8) "Меркурий – спутник Марса";

9) "Джордано Бруно – ученик Галилео Галилея";

10) "Не существует целого числа, куб которого оканчивался бы цифрой 2".

Укажите значение истинности для каждого суждения.

IV. Будут ли нижеприведенные суждения равносильными? Если да, то почему?

A = "В этом четырехугольнике один из углов прямой и диагонали равны";

B = "В этом параллелограмме все углы прямые";

C = "В этом ромбе один угол равен 90°".

V. Из сложных суждений выделите простые и обозначьте их буквами:

1. Если три стороны одного треугольника соответственно равны трем сторонам другого, то такие треугольники равны.

2. "Есть мера вещей и существуют известные границы" (афоризм Горация).

3. "Разрешаются от бремени горы, а рождается и смешная мышь" (из Горация).

4. Если сумма цифр числа через одну равна сумме остальных цифр через одну или разность этих сумм делится на 11, то и данное число делится на 11.

5. "Шахтер" выиграл встречу у "Динамо", а встреча с "Таврией" и "Спартаком" закончилась вничью.

6. Студент запланировал выполнить следующие дела: подготовиться к зачету, побывать на тренировке, почитать интересную книгу, поиграть в шахматы.

7. Если завтра будет туман, мы не сможем вылететь на соревнования.

§ 47. Вывод умозаключений

Путь вывода умозаключений лежит через рассуждения, доказательства, умение ставить вопросы и давать на них четкие ответы.

РАССУЖДЕНИЕ – это цепочка взаимосвязанных суждений, фактов и общих положений, получаемых из других суждений по определенным правилам вывода.

Примеры таких правил:

"*Если треугольник равносторонний, то у него все углы равны между собой*", "*Если король под шахом и ему некуда ходить, то – мат*", "*Если идет дождь, то необходимо открыть зонт*".

Любое правило вывода умозаключений состоит из двух суждений (простых или сложных). Мы уже говорили, что одно из них называется **предпосылкой** или **условием**, а второе – **следствием, заключением** или **выводом**.

Например, в рассуждениях:

"*Если треугольник равносторонний, то у него все углы 60°*", суждение "*У него все углы равны 60°*" – это заключение, суждение "*Треугольник равносторонний*" – предпосылка.

В правиле "*Если идет дождь, то необходимо открыть зонт*" второе суждение "*необходимо открыть зонт*" является следствием того, что "*идет дождь*".

Приведем пример более сложных рассуждений.

Некогда жил жестокий правитель, который не желал никого впускать в свои владения. У моста через пограничную реку был поставлен часовой, вооруженный с головы до ног, и ему приказано было спрашивать каждого путника: "Зачем идешь?".

Если путник в ответ говорил неправду, часовой обязан был схватить его и тут же повесить. Если же путник отвечал правду, ему и тогда не было спасения: часовой должен был немедленно утопить его в реке.

Таков был суровый закон жестокосердного правителя, и неудивительно, что никто не решался приблизиться к его владениям.

Но вот нашелся мудрец, который, несмотря на это, спокойно подошел к охраняемому мосту у запретной границы.

– Зачем идешь? – сурово остановил его часовой, готовясь казнить смельчака, безрассудно идущего на верную гибель.

Но ответ был таков, что озадаченный часовой, строго исполняя жестокий закон своего господина, не мог ничего поделать с мудрецом.

Что же ответил мудрец?

На вопрос часового "Зачем идешь?" логика рассуждений в лице мудреца дает такой ответ:

– Я иду, чтобы быть повешенным вот на этой виселице.

Такой ответ поставил часового в тупик. Что он должен сделать с мудрецом? Повесить? Но тогда выйдет, что мудрец сказал правду, за правдивый же ответ было приказано не вешать, а топить. Но и топить нельзя: в таком случае окажется, что мудрец солгал, а за ложное показание предписывалось повесить.

Так часовой и не мог ничего поделать со сметливым мудрецом.

Существуют определенные приемы вывода умозаключений, которые в ряде случаев облегчают поиск правильных рассуждений, доказательств или способов решения задач.

Наиболее ценные из них – это аналогия, индукция и дедукция.

УМОЗАКЛЮЧЕНИЕ ПО АНАЛОГИИ – это знание, полученное из рассмотрения какого-либо объекта, переносимое на менее изученный, сходный по существенным свойствам и качествам объект.

"Аналогия" – греческое слово, в переводе оно означает "сходство".

Например, из суждения: *"Солнечная система – это планеты, вращающиеся по орбитам, в центре которых находится Солнце"*, получаем умозаключение по аналогии:

"Атом – это электроны, вращающиеся по орбитам, в центре которых находится ядро".

И еще пример. Даны суждения:

"Квадрат – это равносторонний и равноугольный параллелограмм".

"Квадрат – это равносторонний прямоугольник".

По аналогии можно в качестве третьего определения привести следующее умозаключение:

"Квадрат – это равноугольный ромб".

Но подмеченная аналогия не всегда может служить доказательством!

Суждения, сформулированные по аналогии с истинными, могут оказаться ложными. Например, по аналогии с истинными суждениями: *"Если сумма цифр числа делится на 3, то и само число делится на 3"*; *"Если сумма*

чисел делится на 9, то и само число делится на 9" – можно сформулировать и суждение: "*Если сумма цифр делится на 27, то и само число делится на 27*". Но это суждение явно ошибочно. Чтобы убедиться в этом, рассмотрите число 2799 – сумма его цифр делится на 27, а само число на 27 не делится.

Рассмотрим другие важные приемы, облегчающие поиск верных умозаключений, – индукцию и дедукцию.

ИНДУКЦИЯ – это правило вывода умозаключений при переходе от частных суждений к общим.

Само слово "индукция" в переводе с латинского означает "наведение".

Этому правилу обычно противопоставляют дедукцию – "выведение".

ДЕДУКЦИЯ – это правило вывода умозаключений при переходе от общих суждений к частным.

Проверьте свои знания ?

1. Что такое рассуждение?
2. Что мы понимаем под предпосылкой и заключением в сложном суждении вывода умозаключения? Приведите примеры таких суждений, выделив соответствующие части.
3. В чем состоит вывод умозаключения по аналогии? Всегда ли вывод по аналогии является истинным? Приведите примеры таких выводов.
4. В чем состоит принцип индукции, дедукции?

Тренировка

I. В следующих суждениях выделите предпосылки и заключения. Определите, истинны они или ложны:

1) произведение двух чисел равно 0, если хотя бы один из сомножителей равен 0;

2) если $A \cdot B > 0$, то $A > 0$ и $B > 0$;

3) если в треугольнике все стороны равны, то и все углы равны;

4) если на улице сыро, то прошел дождь.

II. 1) Дано суждение: "Каждый равносторонний треугольник является также равноугольным". Сформулируйте аналогичное суждение для шестиугольника;

2) Дано истинное суждение: "Существует правильный многоугольник с любым наперед заданным числом сторон $N \geq 3$".

Сформулируйте аналогичное суждение для правильного многогранника. Истинно ли оно?

III. Турист шел к озеру. У перекрестка сидели двое парней, каждый из которых знал, какая дорога ведет к озеру. На вопросы они отвечали только "да" и "нет". Один из них всегда говорил правду, другой всегда лгал. Все это знал турист, но не знал, какая из двух дорог ведет к озеру.

Проверьте свои способности логически мыслить: вам предлагается решить задачу, используя три варианта:

1-й. Турист задал два вопроса одному из парней.

2-й. Турист задал один и тот же вопрос каждому из парней.

3-й. Турист задал один вопрос одному из парней.

Что это были за вопросы, которые позволили туристу найти нужную дорогу?

Примечание. На обдумывание каждого варианта отводится не более 5 мин.; 1-й вариант оценивается оценкой "3"; 2-й вариант – "4"; 3-й вариант – "5".

IV. Три мудреца вступили в спор: кто из троих более мудрый? Спор помог решить случайный прохожий, предложивший им испытание на сообразительность.

– Вы видите у меня, – сказал он, – пять колпаков: три черных и два белых. Закройте глаза!

С этими словами он надел каждому по черному колпаку, а два белых спрятал в мешок.

– Можете открыть глаза, – сказал прохожий. – Кто угадает, какого цвета колпак украшает его голову, тот вправе считать себя самым мудрым.

Долго сидели мудрецы, глядя друг на друга...

Наконец, один воскликнул:

– На мне черный!

Как он догадался?

V. Саша сказал:

– У Димы больше тысячи книг.

– Нет, – возразил Коля, – книг у него меньше.

– Одна-то книга у него наверняка есть, – сказала Оля.

Если истинно только одно из этих утверждений, то сколько же книг у Димы?

§ 48. Алгебра суждений

Для составления сложных суждений, как правило, используют простые суждения, соединяя их знаками логических операций: **“и”**, **“или”**, **“нет”**, **“если..., то...”**. Значение истинности сложных суждений полностью определяется значениями истинности составляющих элементарных суждений. Это дает возможность заниматься своеобразным исчислением суждений, т.е. определять значение истинности формулы на основании значений истинности составляющих суждений.

Парадокс с карточкой математика П.Жирдена. Какое бы значение истинности вы ни приписали любому из них, оно будет противоречить другому утверждению. Ни одно из утверждений не содержит ссылки на себя, но, взятые вместе, эти два утверждения воспроизводят парадокс.

I. **Отрицание.** Имея суждение A, можно образовать новое суждение, которое читается как “не A” или “неверно, что A”. Для обозначения отрицания суждения употребляют символ “ ¬ ” и пишут ¬A. В программировании операцию отрицания обозначают “NOT” (от английского “не”). Иногда обозначают отрицание еще так: $\bar{A}$. Мы будем пользоваться последним обозначением.

Таблицу истинности отрицания A можно представить так:

Таблица 11

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

Так как возможны только два значения переменной, то всегда:

$$\bar{1} = 0 \text{ и } \bar{0} = 1.$$

Пусть суждение A = “*Мы любим информатику*”, тогда отрицанием будет: $\bar{A}$= “*Мы не любим информатику*”. От-

Рис.39. Отрицание Ā

рицание Ā имеет значение "истинно", если исходное суждение А ложно. И наоборот, Ā имеет значение "ложно", если исходное суждение А истинно. Последнее можно подтвердить рисунком (рис. 39).

Эту операцию называют также **инверсией** (или **логическим "не"**).

II. **Конъюнкция.** Конъюнкция двух высказываний А и В соответствует союзу "и". Она обозначается символами "Λ" или "&" (читается "ЭТ"). В программировании эту операцию обозначают "AND" (от английского "и"). Мы будет пользоваться символом "*".

Запись А*В читается так: "Конъюнкция суждений А,В", или "А конъюнкция В", или "суждение А и суждение В", или, совсем коротко, "А и В" (рис. 40).

Таблица истинности конъюнкции двух суждений А и В такова:

Таблица 12

A	B	A*B	A	B	A*B
1	1	1	0	1	0
1	0	0	0	0	0

Из таблицы истинности следует, что операция конъюнкции (операция "и") – это **логическое умножение**, которое ничем не отличается от традиционного умножения в обычной алгебре. Например, пусть есть суждения:

А= "*Сегодня солнечный день*",

В = "*Остап пошел купаться*".

Тогда конъюнкция А*В есть суждение:

Х = "*Сегодня солнечный день, и Остап пошел купаться*".

Это новое суждение Х будет истинным, если одновременно и суждение А истинно (сегодня действительно солнечный день), и суждение В истинно (Остап на самом деле

пошел купаться); если же хотя бы одно из этих суждений ложно (если неправда, что сегодня солнечный день, или неправда, что Остап пошел купаться, или то и другое неправда), то мы, естественно, уже будем считать, что построенное выше составное суждение X ошибочно, ложно.

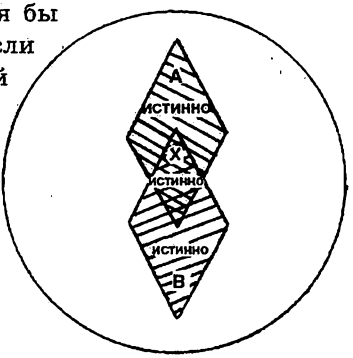


Рис. 40. Конъюнкция X=A*B

В обыденной речи иногда в роли союза "и" выступают союзы "а", "но". Например, "*Богдан был победителем, а Степан занял второе место*". В этом суждении союз "а" выступает в роли союза "и". Это же предложение можно сформулировать и так: "*Богдан был победителем, и Степан занял второе место*".

Итак:

Связка "и" в составных суждениях всегда предполагает одновременную истинность составляющих суждений.

III. **Дизъюнкция** двух суждений A и B соответствует союзу "или" и обозначается символом "V" или символом "+". В программировании эта операция обозначается союзом "OR" (от английского "или"). Мы будем пользоваться символом "+".

Запись A+B может быть прочитана так: "Дизъюнкция суждений A, B", или "A дизъюнкция B", или "суждение A или суждение B", или, совсем коротко, "A или B".

Таблица истинности дизъюнкции двух суждений A и B такова:

Таблица 13

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

Из таблицы истинности следует, что операция дизъюнкции (операция "или") – **логическое сложение** – немного,

но все же отличается от обычного алгебраического сложения. А именно: отличается лишь первой строкой: 1+1=1. Результат этот также не совпадает со сложением двоичных чисел (там было: 1+1=10). Это следствие того, что 1 является не числом "один", а только символом, смысл которого был пояснен выше. Если имеются две истинные величины, то результатом их сложения будет истинная величина, но не может быть ни дважды истинно, ни полуистинно! Именно поэтому 1+1=1.

Например, пусть даны суждения

А="*снег пойдет ночью*",

В="*снег пойдет утром*".

Тогда суждение X = A + B = "*Снег пойдет ночью или утром*".

В этом примере связка "или" играет **объединяющую роль** (рис. 41).

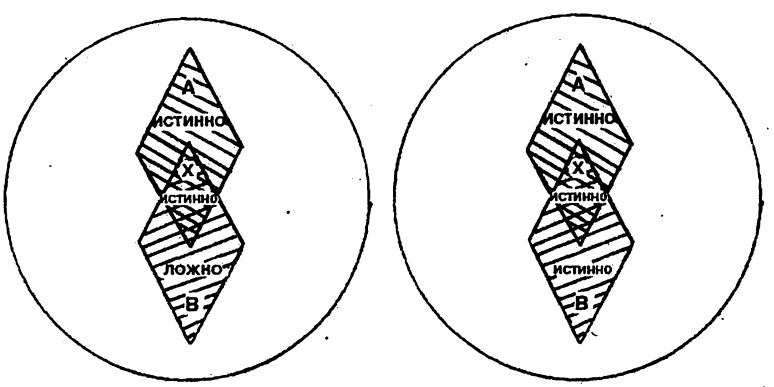


Рис. 41. Объединяющая дизъюнкция X=A+B

Приведем другой пример. Даны суждения:

А="*Он придет сегодня*",

В="*Он придет завтра*".

Суждение X=A+B="*Он придет сегодня или завтра*". В последнем случае связка "или" играет только **разъединяющую роль** (ее можно заменить разделяющим "либо"). Возможны только два варианта:

1. "*Он придет сегодня*" либо
2. "*Он придет завтра*".

Такая дизъюнкция называется **строгой** или **разделенной дизъюнкцией**, она истинна в том и только в том случае, когда одно из суждений истинно, а другое ложно. В этом случае A+B читается "строго A или B" или "либо A, либо B". В программировании такое исключающее "или" обозначают – "XOR" (рис. 42).

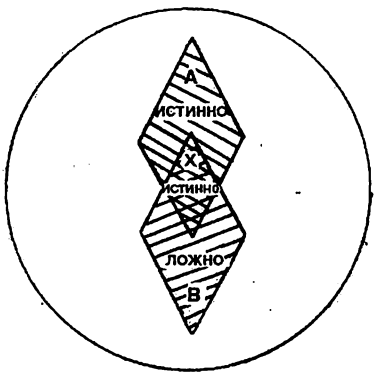


Рис. 42. Строгая дизъюнкция A+B

Строгая дизъюнкция может быть выражена через уже рассмотренные нами объединяющую дизъюнкцию и конъюнкцию:

$$X=(A*\bar{B})+(\bar{A}*B)$$

Скобки, как всегда, регулируют порядок выполнения логических операций. Такая дизъюнкция будет иметь такую же таблицу истинности, как и объединяющая, если в ней первую строку 1+1=1 заменить на строку 1+1=0.

Во всех машинных приложениях и математических рассуждениях предполагается единственная трактовка дизъюнкции – **объединяющая**. В ней связка "или" понимается только в более широкой объединяющей роли. Например, в суждении: "*На кружок по информатике могут ходить или те ученики, которые желают этого, или те, которых пригласил преподаватель*". Связка "или" означает:

1) либо "те ученики, которые желают этого";

2) либо "те ученики, которых пригласил преподаватель".

Итак, общее правило:

Составное суждение со связкой "или" считается истинным, если истинно хотя бы одно из составных суждений, и считается ложным, если ложны все его составляющие.

IV. **Импликация** двух суждений соответствует союзу "если..., то..." и обозначается символами "⇒" или "⊃". В

программировании саму логическую операцию обозначают "IMP", а союз "если..., то..." заменяют связкой "IF..., THEN... (см. последующие параграфы).

Запись A⇒B может быть прочитана так: "Если A, то B", или "Если верно суждение A, то верно и суждение B", а также более коротко, "Из A следует B", "A влечет B", "B следует из A" и др. Иногда ее называют **логическим следованием.**

Таблица истинности импликаций двух высказываний такова:

Таблица 14

A	B	A⇒B	A	B	A⇒B
1	1	1	0	1	1
1	0	0	0	0	1

Эта таблица показывает, что из истинного высказывания A следует только истинное высказывание B. Ложное же высказывание A имплицирует как ложное, так и истинное высказывание B (говорят: "Из лжи – все, что угодно").

По сути дела, импликация лежит в основе процесса вывода умозаключений. Поэтому в импликации A⇒B суждение A называют **условием** или **посылкой импликации**, а B – **заключением** или **следствием импликации** (см. предыдущий параграф).

Например, пусть истинно следующее суждение:

A = "*треугольник равносторонний*".

Тогда, как известно, истинно суждение:

B = "*треугольник равноугольный*",

поэтому импликация:

X = A⇒B = "*если треугольник равносторонний, то он равноугольный*" истинна (см. первую строчку таблицы).

Последние две строчки таблицы могут вызвать некоторые сомнения. Ведь в этих случаях, согласно таблице, считаются истинными суждения: "Если A ложно, то B истинно" и "Если A ложно, то B тоже ложно".

Однако такая договоренность отражает тот известный факт, что из ложного суждения можно (с помощью правильных рассуждений) вывести как правильное, так и ошибочное заключение.

А теперь о приоритетности перечисленных операций:

Порядок выполнения логических операций следующий: первыми выполняются операции "НЕ", когда они охватывают две или более переменных, затем выполняются операции "И" и последними "ИЛИ" (см. табл. 8).

Таким образом, сохраняется тот же порядок выполнения, как в обычной алгебре: сначала все умножения, а потом сложения.

Если сравнить обычную алгебру и алгебру логики, то на последнюю нельзя распространить все арифметические действия, а только два – сложение и умножение. В алгебре логики **нет вычитания**, поэтому **нет деления**. Поскольку нет деления, при преобразовании логических функций **нельзя сокращать** общий множитель.

А теперь, зная таблицы истинности для эквивалентности, отрицания, конъюнкции, дизъюнкции и импликации, а также учитывая приоритетность перечисленных операций, мы можем научиться составлять таблицы истинности для сложных формул. Учтем лишь, что операции в скобках выполняются в первую очередь.

Например, необходимо составить таблицу истинности для формулы $(A*B+C)+(\overline{A}*\overline{C})$. Первыми выполняются операции в скобках. В самих же скобках – в первую очередь операции отрицания, затем логическое умножение и последней – логическое сложение.

Известно ли вам, что...

Самый молодой миллиардер в мире

Уильям Гейтс, основатель фирмы по производству программного обеспечения Microsoft. Ему было 33 года, когда он в списке богатейших людей мира занимал 131-е место. Вот что может ЭВМ!

Своим успехом Гейтс обязан в равной, пожалуй, мере везению и таланту. Научился программировать в 13 лет (талант? Безусловно! Повезло? А как же! Двадцать семь лет назад и в США ЭВМ были не очень-то доступны); будучи школьником,

когда в продаже появился первый персональный компьютер "Альтаир 8800" – машина с ОЗУ 126 байт, без клавиатуры и дисплея, без внешней памяти. Герою нашей истории вместе с его другом П.Алленом пришло в голову написать для "Альтаира" интерпретатор Бейсика. Эта программа имела полный успех, была первой коммерческой программой для персонального компьютера и положила начало коммерческо-программистской деятельности, принесшей 15 лет спустя такие внушительные плоды.

Сейчас фирма Microsoft занимает первое место среди производителей программного обеспечения, получая прибыль в тысячи миллионов долларов.

Сначала составляем таблицу с колонками (столбцами) для суждений A,B,C (перебирая варианты):

A	B	C	A	B	C
1	1	1	1	1	0
1	0	1	1	0	0
0	1	1	0	1	0
0	0	1	0	0	0

Теперь выполним действия в первой скобке. Сначала логическое умножение A*B (в помощь привлекаем табл. 12):

A	B	C	A*B	A	B	C	A*B
1	1	1	1	1	1	0	1
1	0	1	0	1	0	0	0
0	1	1	0	0	1	0	0
0	0	1	0	0	0	0	0

Затем – логическое сложение (A*B)+C (в помощь привлекаем табл. 13):

A	B	C	A*B	(A*B)+C
1	1	1	1	1
1	0	1	0	1
0	1	1	0	1
0	0	1	0	1
1	1	0	1	1
1	0	0	0	0
0	1	0	0	0
0	0	0	0	0

Теперь выполним действия во второй скобке. Сначала – логические отрицания: $\overline{A}$ и $\overline{C}$ (в помощь привлекаем табл. 11):

A	B	C	A*B	(A*B)+C	$\bar{A}$	$\bar{C}$
1	1	1	1	1	0	0
1	0	1	0	1	0	0
0	1	1	0	1	1	0
0	0	1	0	1	1	0
1	1	0	1	1	0	1
1	0	0	0	0	0	1
0	1	0	0	0	1	1
0	0	0	0	0	1	1

Затем – логическое умножение A*C (в помощь снова привлекаем табл. 12):

A	B	C	A*B	(A*B)+C	$\bar{A}$	$\bar{C}$	$\bar{A}*\bar{C}$
1	1	1	1	1	0	0	0
1	0	1	0	1	0	0	0
0	1	1	0	1	1	0	0
0	0	1	0	1	1	0	0
1	1	0	1	1	0	1	0
1	0	0	0	0	0	1	0
0	1	0	0	0	1	1	1
0	0	0	0	0	1	1	1

И, наконец, произведем логическое сложение двух скобок (A*B+C)+ ($\bar{A}*\bar{B}$) (снова в помощь привлекаем табл. 13):

A	B	C	A*B	(A*B)+C	$\bar{A}$	$\bar{C}$	$\bar{A}*\bar{C}$	A*B+C)+ +($\bar{A}*\bar{C}$)
1	1	1	1	1	0	0	0	1
1	0	1	0	1	0	0	0	1
0	1	1	0	1	1	0	0	1
0	0	1	0	1	1	0	0	1
1	1	0	1	1	0	1	0	1
1	0	0	0	0	0	1	0	0
0	1	0	0	0	1	1	1	1
0	0	0	0	0	1	1	1	1

A	B	C	$(A*B+C)+(\bar{A}*\bar{C})$	A	B	C	$(A*\bar{B}+C)+(\bar{A}*\bar{C})$
1	1	1	1	1	1	0	1
1	0	1	1	1	0	0	0
0	1	1	1	0	1	0	1
0	0	1	1	0	0	0	1

Последняя колонка и будет таблицей истинности исходной формулы.

А теперь приведем пример более сложной формулы (в ней используются все логические операции):

Докажем эквивалентность формул: $A*(A\Rightarrow B)\Rightarrow B)$ и $((A\equiv B)*(A*\bar{B}))$

A	B	$A\Rightarrow B$	$A*(A\Rightarrow B)$	$A*(A\Rightarrow B)\Rightarrow B$
1	1	1	1	1
1	0	0	0	1
0	1	1	0	1
0	0	1	0	1

A	B	$A\equiv B$	$\bar{B}$	$A*\bar{B}$	$(A\equiv B)*(A*\bar{B})$	$\overline{(A\equiv B)*(A*\bar{B})}$
1	1	1	0	0	0	1
1	0	0	1	1	0	1
0	1	0	0	0	0	1
0	0	1	1	0	0	1

Сравним последние колонки – они одинаковы, следовательно, эквивалентность формул доказана.

Проверьте свои знания ?

1. С какими основными операциями работает алгебра суждений?
2. Что такое логическое отрицание? Приведите таблицу истинности отрицания.
3. Что такое конъюнкция? Можно ли утверждать, что конъюнкция – это и есть логическое умножение?
4. Приведите таблицу истинности конъюнкции.

5. Что такое дизъюнкция? Можно ли утверждать, что дизъюнкция – это и есть логическое сложение?

6. Приведите таблицу истинности дизъюнкции.

7. Какие два вида дизъюнкции вы знаете? Приведите примеры.

8. Что такое импликация? Приведите таблицу истинности импликации.

9. Какова приоритетность выполнения логических операций?

Тренировка

I. Выделите в сложных суждениях простые и обозначьте их буквами. Представьте эти сложные суждения в виде формул:

1) утром мы пойдем на рыбалку, позагораем, но уху варить не будем;

2) число 156 делится на 3 и 4, но не делится на 7;

3) после школы я поступлю или на экономический, или на юридический, или на факультет иностранных языков;

4) на каникулах я поеду на лечение в Трускавец или Нимиров, а мой класс поедет на экскурсию в Ялту или Севастополь;

5) если завтра не будет дождя, то мы пойдет купаться на речку или пойдем собирать грибы в лес;

6) число делится на 6 тогда и только тогда, когда оно делится на 2 и на 3;

7) для того чтобы параллелограмм был квадратом, необходимо и достаточно, чтобы он был ромбом и имел прямой угол или был прямоугольником и имел равные смежные стороны.

II. Сложные суждения составляются из следующих простых:

A= "ДОКТОР УОТСОН – ОТСТАВНОЙ ОФИЦЕР";

B= "ДОКТОР УОТСОН – ДРУГ ЗНАМЕНИТОГО СЫЩИКА";

C= "ДОКТОР УОТСОН ОКОНЧИЛ ЛОНДОНСКИЙ УНИВЕРСИТЕТ".

Прочтите формулы сложных суждений, используя смысл каждого из простых суждений:

а) $A+B*C$ в) $\overline{C}*\overline{B}*\overline{A}$ д) $(C+A)\Rightarrow B$

б) $B*C*\overline{A}$ г) $B\Rightarrow(C*A)$ е) $(\overline{A}*\overline{C})\Rightarrow B$

III. Известно, что дизъюнкция $\overline{A}+B+\overline{C}$ ложна. Что можно сказать о значении истинности самих высказываний A, B, C?

IV. Составьте таблицы истинности для следующих формул:

1) $A*\overline{B}$ 4) $A*(B+\overline{A})$ 7) $(A\Rightarrow B)+\overline{B}$ 10) $\overline{(A*\overline{B})}\equiv(\overline{A}+(A*B))$

2) $A*B*\overline{C}$ 5) $(A*B)+\overline{C}$ 8) $A*\overline{B}*(A\Rightarrow B)$ 11) $(A\equiv B)*(A*\overline{B})+(\overline{A}*B)$

3) $\overline{A}*\overline{B}*\overline{C}$ 6) $A+(A*\overline{B})$ 9) $(A*\overline{B})\Rightarrow(\overline{B}\Rightarrow A)$ 12) $(\overline{A}\Rightarrow(B\Rightarrow\overline{C}))\equiv(A*B*\overline{C})$

V. Составлением таблиц покажите равносильность следующих формул:

1) $\overline{(A\Rightarrow B)}$ и $A*\overline{B}$ 2) $A*B$ и $A*B$ 3) $A+B*C$ и $(A+B)*(A+C)$

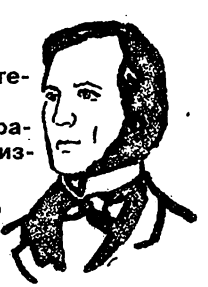
§ 49. Булева алгебра

В 1847 г. английский математик Джордж Буль, преподаватель Коркского университета, разработал алгебру логики (мы с ее основными действиями познакомились в предыдущем параграфе). Почти 100 лет эта "алгебра высказываний" не была известна широкому кругу пользователей. Лишь в 1938 году выдающийся американский математик и инженер Клод Шеннон обнаружил, что алгебра логики приложима к любым переменным, которые могут принимать только два значения. Например, к состоянию контактов: включено – выключено или напряжению (или току): есть – нет, которыми представляется информация в ЭВМ.

В результате алгебра логики явилась математической основой теории электрических и электронных переключательных схем, используемых в ЭВМ, поэтому ее предпочитают называть не алгеброй логики, а Булевой алгеброй – по имени ее создателя.

Приведем **Булевы алгебры.** Если условимся писать вместо знаков дизъюнкции и конъюнкции соответственно более привычные знаки "+" и "*", а вместо знака эквивалентности – знак "=", то мы получим, что для любых

Джордж Буль (1815-1864)



Английский математик, основоположник математической логики.

Не имея специального математического образования, все же за свои научные труды был избран профессором математики (в Ирландии).

В работах "Математический анализ логики", "Исследование законов мышления" предпринял попытку построить формальную логику в виде некоторого "исчисления", "алгебры".

Логические исчисления, построенные в соответствии с идеями Буля, находят сейчас широкое применение в вычислительной технике, в частности, в теории релейно-контактных схем.

Именем Буля названы так называемые Булевы алгебры – особые алгебраические системы, для элементов которых определены две операции.

ЧТО может КОМПЬЮТЕР

Управлять сердцем

Когда речь заходит о компьютерах, люди старшего поколения, как правило, сразу же пытаются ретироваться. А вот один американец в возрасте 71 года из штата Теннесси носит компьютер в сердце.

Программируемый кардиостимулятор, разработанный в Алабамском госпитале, – первый из имплантируемых приборов подобного типа, в котором применено компьютерное управление режимами работы стимулятора. Новый аппарат не только обеспечивает ра-

суждений справедливы следующие зависимости:

1. A + B = B + A

(Коммутативность сложения или переместительный закон).

2. A * B = B * A

(Коммутативность умножения или сочетательный закон).

3. (A + B) + C = A + (B + C)

(Ассоциативность сложения или распределительный закон сложения).

4. (A * B) * C = A * (B * C)

(Ассоциативность умножения или распределительный закон умножения).

5. A *.(B + C) = A*B + A*C

(Дистрибутивность умножения относительно сложения).

6. A + B*C = (A + B)*(A + C)

(Дистрибутивность сложения относительно умножения).

7. A + A = A

(Идемпотентность сложения).

8. A * A = A

(Идемпотентность умножения).

9. A*(B + $\overline{B}$)=A (или A*(A + B)= A; (A + B)*$\overline{B}$= A * $\overline{B}$)

10. A + B *$\overline{B}$ = A (или A+(A * B) = A; (A*B) + $\overline{B}$ = A + $\overline{B}$)

(Правила поглощения).

11. $\overline{A + B}$ = $\overline{A}$*$\overline{B}$ (или A + B = $\overline{\overline{A}*\overline{B}}$)

12. $\overline{A*B}$ = $\overline{A}$ + $\overline{B}$ (или A*B = $\overline{\overline{A} + \overline{B}}$)

(Правила де Моргана).

13. $\overline{\overline{A}}$ = A

(Двойное отрицание или закон отрицания отрицания).

Суждения, истинность которых постоянна и не зависит от истинности входящих в них простых суждений, а определяется только их структурой, называются ТОЖДЕСТВЕННЫМИ или ТАВТОЛОГИЯМИ.

Кроме того, если обозначить через 0 суждение, которое наверняка ложно, а через 1 – суждение, которое заведомо истинно, то имеют место еще такие зависимости:

14. $A + \bar{A} = 1$

(А или не А всегда истинно; закон исключения третьего).

15. $A * \bar{A} = 0$

(А и не А всегда ложно; закон непротиворечивости).

16. $1 + A = 1$

(Истина или А равносильно истине – тавтология тавтологии).

17. $1 * A = A$

(Истина и А равносильно истине – тавтология тавтологии).

18. $0 + A = A$

(Противоречие или А равносильно А).

19. $0 * A = 0$

(Противоречие и А есть противоречие).

Овладев этими основными свойствами суждений, можно упрощать формулы логики суждений уже формально, подобно тому, как в алгебре выполняются тождественные преобразования.

Пример. Упростите суждение $A*((\bar{B}+\bar{C})+B*C)+\bar{A}$

боту сердца в заданном режиме, но и подстраивается под его ритм, обеспечивая тем самым наиболее комфортабельные условия для жизни пациента. Кроме того, стимулятор постоянно собирает информацию о работе сердца, которая позднее, на приеме у лечащего врача, может быть считана и обработана с помощью более мощного стационарного компьютера.

Проанализировав работу сердца пациента за последний период, врач с помощью своей кабинетной ЭВМ может подстроить программу компьютера вживленного стимулятора посылкой радиокоманды, задав либо более мягкий, адаптивный режим работы стимулятора, либо более жесткий и независимый от ритма сердца режим работы в заданном темпе.

$$A*((\underbrace{\bar{B}+\bar{C}}_{\text{по 12}})+B*C)+\bar{A} = A*(\underbrace{\overline{B*C}+B*C}_{\text{по 14}})+\bar{A} = \underbrace{A*1}_{\text{по 17}}+\bar{A} = \underbrace{A+\bar{A}}_{\text{по 14}} = 1$$

Имеем $A*((\bar{B}+\bar{C})+B*C)+\bar{A} = 1$.

Для практических применений алгебры суждений важным является то, что любую формулу можно преобразовать так, что:

Таблица 15

Не будут использованы	Будут использованы
Знаки логического сложения	Знаки отрицания и логического умножения
Знаки логического умножения	Знаки отрицания и логического сложения

Пример. Суждение $\overline{A+B}$ преобразовать в эквивалентное ему суждение, не содержащее знака логического сложения:

$$\underbrace{\overline{\bar{A}+B}}_{\text{по 11}} = \underbrace{\bar{\bar{A}}}_{\text{по 13}} * \bar{B} = A * \bar{B}.$$

Пример. Суждение $A*\overline{B*C}$ преобразовать в эквивалентное ему суждение, не содержащее знаков логического умножения:

$$A * \underbrace{\overline{B*C}}_{\text{по 12}} = \underbrace{A * (\bar{B}+\bar{C})}_{\text{по 5}} = \underbrace{A*\bar{B}}_{\text{по 13}} + \underbrace{A*\bar{C}}_{\text{по 13}} = \underbrace{\overline{\overline{A*\bar{B}}}}_{\text{по 12}} + \underbrace{\overline{\overline{A*\bar{C}}}}_{\text{по 12}} =$$

$$= \underbrace{\overline{\overline{A}+\overline{B}}}_{\text{по 13}} + \underbrace{\overline{\overline{A}+\overline{C}}}_{\text{по 13}} = \overline{A} + \underbrace{\overline{\overline{B}}}_{\text{по 13}} + \overline{A} + \underbrace{\overline{\overline{C}}}_{\text{по 13}} = \overline{A} + B + \overline{A} + C.$$

Проверьте свои знания ?

1. Кто является основоположником алгебры высказываний?

2. Почему долгое время Булева алгебра не была востребована специалистами?

3. Почему алгебру логики считают основой теории электрических и электронных переключательных схем?

4. В чем состоит удобство пользования Булевой алгебры?
работы с суждениями при ис- 5. Что такое тавтология?

Тренировка

I. С помощью таблиц истинности докажите:

а) эквивалентность правил де Моргана:

1. $A + B = \overline{\bar{A} * \bar{B}}$ 2. $A * B = \overline{\bar{A} + \bar{B}}$;

б) что импликацию можно заменить отрицанием и дизъюнкцией:

$(A \Rightarrow B) = \bar{A} + B$ (формула 20);

в) что эквивалентность можно заменить конъюнкцией, дизъюнкцией и отрицанием:

$(A \equiv B) = ((A * B) + (\bar{A} * \bar{C}))$ (формула 21).

II. Используя Булевы алгебры и формулы из предыдущего номера тренировки, упростите формулы сложных суждений:

а) $(P \Rightarrow \bar{C}) * (C \Rightarrow P)$;

б) $A * \bar{B} * \bar{C} + \bar{B} * \bar{C} + A$;

в) $(A * B) + (\bar{A} * B) + (\bar{A} * \bar{B})$;

г) $(\bar{A} * P * C) + (A * \bar{P} * C) + (A * P * \bar{C}) + (A * P * C)$.

III. Преобразуйте в равносильные формулы так, чтобы использовались только логическое сложение и отрицание:

а) $(X * Y) \Rightarrow (\bar{X} \Rightarrow Z)$;

б) $(\bar{A} * P) \Rightarrow (\bar{P} * A)$.

IV. Преобразуйте в равносильные формулы так, чтобы использовались только логическое умножение и отрицание:

а) $A + P + \bar{C}$;

б) $(X + Y) \Rightarrow (\bar{X} \Rightarrow Z)$.

V. На допросе свидетель А сказал, что показания В неверны. Следователь рассуждает так: "Если свидетель А сказал неправду, тогда В говорит правду, или свидетель А говорит правду, и В говорит неправду".
Запишите рассуждения следователя в виде формулы.

VI. Ответственный за дежурство в классе составляет график на неделю.

Андрей может дежурить или во вторник или в пятницу; Борис может дежурить только в понедельник или пятницу, а Володя свободен лишь по средам и четвергам; Гена сказал, что если Андрей не будет дежурить во вторник, то он подежурит в четверг, а Дима категорически заявил, что если Гена будет дежурить в четверг, то он сможет подежурить лишь во вторник.

Составьте расписание дежурства так, чтобы оно было приемлемо для всех ребят.

§ 50. Решение задач с помощью алгебры суждений

Задача 1

Один из трех братьев поставил на скатерть кляксу.
– Кто испачкал скатерть? – спросила бабушка.
– Витя не ставил кляксу, – сказал Алеша. – Это сделал Боря.
– Ну, а ты что скажешь? – спросила бабушка Борю.
– Это Витя поставил кляксу, – сказал Боря. – А Алеша не пачкал скатерть.
– Так я и знала, что вы друг на дружку сваливать будете, – рассердилась бабушка. – Ну, а каков твой ответ? – спросила она Витю.
– Не сердись, бабуля! Я знаю, что Боря не мог это сделать. А я сегодня не готовил уроки, – сказал Витя.
Оказалось, что двое мальчиков в каждом из двух своих заявлений сказали правду, а один оба раза сказал неправду. Кто поставил на скатерть кляксу?

Приведем два способа решения:

1. Сначала решим ее так называемым здравым рассуждением, без привлечения каких-либо специальных математических теорий.

В задаче сказано, что двое мальчиков сказали правду (в каждом из двух заявлений), а один сказал неправду (оба раза).

Если сказанное Витей: "Я сегодня не готовил уроки" – неправда, тогда правдиво и первое высказывание Вити: "Боря не мог это сделать". Из этого следует, что Алеша сказал неправду. А он произнес суждение: "Витя не ставил кляксу". Следовательно, кляксу поставил Витя.

Но такое простое решение получилось потому, что мы в начале решения попали, как говорят, "в яблочко", т.е. наше первоначальное допущение оказалось верным. В противном случае решение задачи резко усложнилось бы.

2. Приведем второй способ решения, используя математический аппарат Булевой алгебры:

Сделаем обозначения. Пусть суждение:

A = "АЛЕША ПОСТАВИЛ КЛЯКСУ", тогда
$\overline{A}$ = "АЛЕША КЛЯКСУ НЕ СТАВИЛ".
Аналогичный смысл других суждений:
W = "ВИТЯ ПОСТАВИЛ КЛЯКСУ",
$\overline{W}$ = "ВИТЯ КЛЯКСУ НЕ СТАВИЛ" и
B = "БОРЯ ПОСТАВИЛ КЛЯКСУ".
$\overline{B}$ = "БОРЯ НЕ СТАВИЛ КЛЯКСУ".

Запишем теперь суждения мальчиков формулами. Алеша сказал, что Витя не ставил кляксу, это сделал Боря. Это суждение запишется формулой:

$$K = \overline{W} * B.$$

Аналогично запишем высказывание Бори, а именно:

$$L = W * \overline{A}.$$

Витя сказал, что Боря не ставил кляксу и что он не готовил уроки. Но последнее совершенно не значит, что Витя не мог поставить кляксу. Поэтому суждение Вити запишется так:

$$M = \overline{B} * (\underbrace{W + \overline{W}}_{\text{по 14-й равно 1}}) = \overline{B} * 1 = \overline{B}.$$

Итак

$$M = \overline{B}.$$

По условию задачи двое мальчиков оба раза сказали правду, а один мальчик оба раза сказал неправду. Поэтому среди записанных нами трех формул K, L, M две истинны, а одна ложна. Мы не знаем, какая формула ложна. Но мы утверждаем, что если из этих формул образовать попарные дизъюнкции, то, поскольку в каждую дизъюнкцию будет входить по крайней мере одна истинная формула, эти дизъюнкции будут истинными. Образуем их, получив новые формулы:

$$X = K + L = (\overline{W} * B) + (W * \overline{A})$$

$$Y = K + M = \underbrace{(\overline{W}*B)+\overline{B}}_{\text{по 1-й формуле}} = \underbrace{\overline{B} + (\overline{W}*B)}_{\text{по 6-й}} = (\overline{B}+\overline{W}) * \underbrace{(\overline{B}+B)}_{\text{по 14-й равно 1}} =$$

$$= \underbrace{(\overline{B} + \overline{W})}_{\text{по 1-й}} * 1 = \overline{W} + \overline{B}$$

$$Z = L + M = (W * \bar{A}) + \bar{B}.$$

Найдем конъюнкцию формул X и Y. Она, конечно же, истинна:

$$X * Y = ((\bar{W} * B) + (W * \bar{A})) * (\bar{W} + \bar{B}) =$$
$$\underbrace{(\bar{W} * B * \bar{W})}_{\text{по 8-й}} + \underbrace{(W * \bar{A} * \bar{W})}_{\text{по 15-й равно 0}} + \underbrace{(\bar{W} * B * \bar{B})}_{\text{по 15-й равно 0}} + (W * \bar{A} * \bar{B})$$
$$= (\bar{W} * B) + (W * \bar{A} * \bar{B}).$$

Теперь найдем конъюнкцию трех формул X, Y, Z:

$$X * Y * Z = (\bar{W} * B + (W * \bar{A} * \bar{B})) * ((W * \bar{A}) + \bar{B}) =$$
$$= \underbrace{\bar{W} * B * W}_{\text{по 15-й равно 0}} * \bar{A} + \underbrace{\bar{W} * B * \bar{B}}_{\text{по 15-й равно 0}} + \underbrace{W * \bar{A} * \bar{B} * W}_{\text{по 2-й}} * \bar{A} + W *$$
$$* \bar{A} * \bar{B} * \bar{B} = \underbrace{W * W}_{\text{по 8-й равно само себе}} * \underbrace{\bar{A} * \bar{A}}_{\text{по 8-й}} * \bar{B} + W * \bar{A} * \underbrace{\bar{B} * \bar{B}}_{\text{по 8-й}} =$$
$$= W * \bar{A} * \bar{B}.$$

Итак

$$X * Y * Z = W * \bar{A} * \bar{B}.$$

Из этой истинной конъюнкции следует, что Виктор кляксу ставил, а Алексей и Борис нет.

Задача 2

Следователь допросил трех лиц A, B и C, подозреваемых в совершении преступления. На допросе A сказал, что показания B неверны. B сказал, что показания C неверны, наконец, C сказал, что и A говорит неправду, и B говорит неправду. Может ли следователь на основании этих показаний установить, кто из допрошенных говорит правду?

Решение

Обозначим буквами D, E, F соответственно, что свидетели A, B, C говорят правду, а $\bar{D}$, $\bar{E}$, $\bar{F}$ будут означать, что они говорят неправду. Переведем задачу на язык символов.

Допрошенный A мог сказать правду, а мог сказать и неправду. Поэтому его показания можно записать такой формулой:

$$A = (D * \overline{E}) + (\overline{D} * E) = 1$$

(читается так: "А говорит правду, а допрошенный В говорит неправду, или А говорит неправду, и тогда В говорит правду"). Это суждение, конечно, является истинным.

Эти рассуждения повторимы относительно сказанного свидетелем В, поэтому запись суждения аналогична:

$$B = (E * \overline{F}) + (\overline{E} * F) = 1.$$

О суждении третьего допрошенного судить несколько труднее. Рассмотрим, что скрывается в его показаниях.

С может говорить правду, в таком случае неправду говорят и А и В. Эту часть суждения запишем так:

$$(F * \overline{D} * \overline{E}),$$

но С может сказать и неправду, в таком случае по крайней мере один из А или В говорит правду.

Это запишем так:

$$(\overline{F} * (D + E)).$$

Объединяя знаком дизъюнкции эти суждения, получим содержание сказанного С:

$$C = (F * \overline{D} * \overline{E}) + (F * (D + E)) = 1.$$

Из полученных трех истинных суждений составляем конъюнкцию, которая будет заведомо истинной:

$$A * B = (D * \overline{E} * F) + (\overline{D} * E * \overline{F}) = 1;$$
$$A * B * C = \overline{D} * E * \overline{F} = 1.$$

Выходит, что А и С говорили неправду, а В сказал правду.

Тренировка

1. Неприятная история. В одном из классов школы разбито окно. Выбить стекло мог только кто-нибудь из четырех

учеников: Леня, Дима, Толя и Миша. При опросе учеников каждый из них дал по три показания.

Леня. 1. Я не виноват. 2. Я даже не подходил к окну. 3. Миша знает, кто это сделал.

Дима. 1. Стекло разбил не я. 2. С Мишей я не был знаком до поступления в школу. 3. Это сделал Толя.

Толя. 1. Стекло разбил не я. 2. Это сделал Миша. 3. Дима говорит неправду, утверждая, что я разбил стекло.

Миша. 1. Я не виноват. 2. Стекло разбил Леня. 3. Дима может поручиться за меня, т.к. знает меня со дня рождения.

При дальнейших расспросах каждый ученик заявил, что сделал два верных заявления и одно ложное. Попробуйте с помощью математической логики найти виновного.

II. Один из четырех мальчиков испортил выключатель. На вопрос, кто это сделал, получены такие ответы:

1. Это сделал или Миша, или Коля.
2. Это сделал или Витя, или Коля.
3. Это не могли сделать ни Толя, ни Миша.
4. Это сделал или Витя, или Миша.

Можно ли по этим данным установить, кто виновен в поломке выключателя, если из четырех суждений три истинны?

III. На марафонском беге было сделано два прогноза о местах, которые займут спортсмены Василенко, Левченко и Симченко, реально претендующие на призовые места:

1. "Симченко будет первым, Василенко – вторым, а Левченко – третьим";
2. "Победит Василенко, Левченко придет вторым, а Симченко будет третьим".

После окончания состязания оказалось, что три фаворита действительно заняли три первых места, но оба предсказания оказались ложными. Ни в одном из предсказаний ни одно из мест не было названо правильно. Какое место занял каждый из спортсменов?

IV. Предстоят спортивные соревнования между четырьмя девятыми классами одной школы. В учительской живо обсуждаются возможные результаты, высказываются прогнозы.

– Первое место займет 9А, а второе – 9Б, – сказал учитель математики.

– Да что вы! – сказал учитель географии. – Я недавно ходил с ними в поход и знаю их возможности. 9А займет второе место, а 9Г – только третье.

– А я думаю, что на втором месте будет 9В, – сказала завуч школы, – а 9Г будет на последнем месте.

Оказалось, что прогнозы их сбылись только наполовину. Какое место занял каждый класс?

Известно ли вам, что...

к одной из самых древних "вычислительных машин" каменного века относят величественные сооружения Стоунхенджа, расположенные на юго-западе Англии на Солсберийской равнине. Этим каменным великанам около 4000 лет. Речь идет о кромлехах – строениях в форме колец, состоящих из вертикально вкопанных в землю каменных монолитов по 40-50 тонн каждый в виде трилистов (это две вертикальные глыбы, на которые положена третья), а также других камнях и лунках, концентрически расположенных вокруг них.

Английский астроном Джеральд Хокинс, проанализировав на ЭВМ астрономические данные о положении небесных тел и геометрические закономерности каменного памят-

§ 51. Логические операции в программировании

В условные выражения, управляющие выбором в операторах IF... THEN, могут входить кроме знакомых нам арифметических операций операции нового класса – логические. Логические операции стоят последними в таблице приоритетов операций (см. § 24), откуда следует, что они выполняются только тогда, когда все остальные операции в выражении уже выполнены.

Мы уже знаем, что к логическим операциям относят три основные операции:

NOT (НЕ)
AND (И)
OR (ИЛИ)

и три дополнительные (использующиеся в расширенных версиях Бейсика):

XOR (исключающее или)
EQV (эквивалентность)
IMP (импликация).

Эти операции надо применять в логических выражениях, при вычислении которых получаются значения либо "истина" (true), либо "ложь" (false).

Каждая операция (кроме одноместной операции NOT) составляет значения двух логических выражений и получает новое значение либо "истина", либо "ложь", в соответствии с правилами логики (см. предыдущие параграфы).

Мы уже знаем, что, если проверка условий в операторе IF, описываемом этими операциями, истинна, компьютер будет выполнять оператор, записан-

ный после служебного слова THEN. В противном же случае (если ложна) он переходит на выполнение следующей стоящей за этим оператором строки.

Например:

```
1. IF A<20 AND A>0 AND B<20 AND B>0
THEN A,B
2. IF Y$<>Z$ OR Y$=" " THEN PRINT
"КОНЕЦ"
3. IF Y=2 AND X>0 AND X<5 OR
M/4=20 THEN GOSUB 100
```

Как показано выше, используя логические операции AND и OR, вы можете записывать более сложные проверки. Но обязательно нужно помнить, что

При использвании операции AND оператор после слова THEN будет выполняться только тогда, когда обе проверки истинны.

При использовании операции OR оператор после THEN выполняется в том случае, если истинна хотя бы одна проверка.

Служебное слово ELSE (иначе) дает возможность сказать компьютеру, что он должен делать, если проверка не истинна, не повторяя оператор IF...THEN.

Например, показанная ниже строка 10

```
10 IF S=5 THEN PRINT "ПРАВИЛЬНО"
ELSE PRINT "НЕПРАВИЛЬНО"
```

выполняет то же самое действие, что и пара строк 20 и 30, но она короче:

```
20 IF S=5 THEN PRINT "ПРАВИЛЬНО"
30 IF S<>5 THEN PRINT "НЕПРАВИЛЬНО"
```

ника, пришел к выводу, что Стоунхендж – не только место ритуальных церемоний и погребений, а прежде всего каменная астрономическая обсерватория, работающая на вычислительных принципах. Здесь с удивительной точностью люди вели счет календарным дням, отмечали начало времен года, предсказывали наступление солнечных и лунных затмений. Это объясняется тем, что направления, заданные трилитами, с высокой точностью совпадают с азимутами восходов и заходов Солнца и Луны в характерные дни года. Солнечные и лунные затмения жрецы предсказывали безошибочно, перекладывая "памятные" камни из одной лунки в другую. Это, по сути своей, была натуральная "вычислительная машина каменного века".

Используя ELSE, можно записать вместе много проверок IF...THEN, но будьте осторожны, иначе получите программу, трудную для понимания. Если в одном операторе используется много операций AND и OR, то, возможно, потребуются скобки.

Пример:

```
1. IF X=10 AND (Y=1 OR Y=2) THEN PRINT "ИСТИНА"
2. IF X=10 AND Y=1 OR Y=2 THEN PRINT "ИСТИНА"
```

В первом примере компьютер будет выполнять оператор после слова THEN только в том случае, если X=10 и при этом Y = 1 или Y=2. Во втором примере условие истинно тогда, когда X=10 и одновременно Y=1, или же всякий раз, когда Y=2.

Приведем пример:

```
100 IF Y>=0 AND Y+ABS(X)<=1 THEN PRINT"ТОЧКА
ПРИНАДЛЕЖИТ ФИГУРЕ" ELSE PRINT "ТОЧКА
НЕ ПРИНАДЛЕЖИТ ФИГУРЕ"
```

В основу этой строки положено сравнение двух логических значений, находящихся по обе стороны от операции AND. Смысл таких выражений легко понять, если запись на языке программирования превратить в обычную математическую запись, а затем представить полученные уравнения в виде графика (рис. 43):

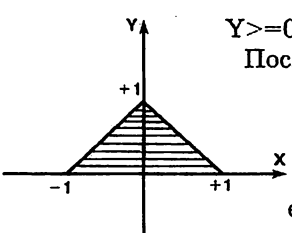


Рис. 43

Y>=0 AND Y+ABS(X)<=1 $Y \geq 0$ и $Y+|X| \leq 1$.

Последнее уравнение распадается на два:

$Y=1+X \quad (X<0)$

$Y=1-X \quad (X \geq 0)$.

Итак, строка 100 говорит, что если заданная точка принадлежит фигуре, то компьютер пишет результат на экране: "ТОЧКА ПРИНАДЛЕЖИТ ФИГУРЕ", в противном случае "ТОЧКА НЕ ПРИНАДЛЕЖИТ ФИГУРЕ".

Вы убедились, что пользователю легче разобраться в логическом выражении, если вместо абстрактных сравнений в него входят наглядные значения переменных или знакомые повседневные понятия.

Например, с помощью логических выражений можно определить свою позицию по отношению к кандидатам во время выборной компании в совет школы. В этом случае позиции преподавателей и учащихся можно представить в качестве выражений или, как их принято называть, **операндов** (тех значений, которые сравниваются или комбинируются).

Для каждого пункта платформы первым операндом является утверждение P = "ПРЕПОДАВАТЕЛИ ПОДДЕРЖИВАЮТ ЭТОТ ПУНКТ", а вторым – U = "УЧАЩИЕСЯ ПОДДЕРЖИВАЮТ ЭТОТ ПУНКТ".

Тогда с помощью логического выражения можно определить ложность или истинность утверждения: Q = "Я ПОДДЕРЖИВАЮ ЭТОТ ПУНКТ".

Используя операцию NOT, операндом которой является позиция преподавателей, вы утверждаете, что ваши мнения противоположные (NOT P). Выражение с операцией AND говорит о том, что вы поддерживаете пункт платформы только в том случае, если его поддерживают и преподаватели, и учащиеся (P AND U). Выражение с операцией OR означает, что вы поддерживаете пункт платформы, если его поддерживает хотя бы одна из сторон (P OR U). В случае операции XOR вы поддерживаете пункт платформы, если его поддерживает только какая-нибудь из сторон, но не обе (P XOR U). Операция EQV противоположна XOR: она говорит о том, что вы поддерживаете пункт платформы, если сразу обе стороны поддерживают или отвергают его (P EQV U). Выражение с операцией IMP утверждает, что вы поддерживаете пункт платформы всегда, кроме случая, когда этот пункт поддерживают преподаватели и отвергают учащиеся.

Проверьте свои знания ?

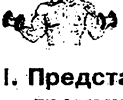
1. Что делает компьютер, если при проверке в операторе IF...THEN логическая операция истинна? Ложна?

2. Назовите правила использования операций AND и OR в операции IF...THEN.

3. Когда применяют служебное слово ELSE в операторе IF...THEN? Можно ли обойтись без него?

4. Что необходимо сделать, чтобы облегчить суть происходящего при выполнении логических операций?

5. Что такое операнд?



Тренировка

I. Представьте в виде логических выражений на языке программирования следующие суждения:

а) $A+D<1 \vee C-D \geq 0$;
б) $(X+Y)^2+C<7{,}5 \wedge X \geq 0$;
в) $\sin X<\cos X \vee N<1$;
г) $\neg(A \wedge \neg B) \equiv \neg A \vee (A \wedge B)$;
д) $\neg(A \wedge \neg B) \supset \neg B \supset A$;
е) $\neg(A \supset (B \supset C)) \equiv A \wedge B \wedge \neg C$.

II. На рис. 44 заштрихована область, ограниченная прямой Y=X и параболой $Y^2=X$. Составьте логическое выражение, принимающее значение "истина" (TRUE) во всех тех и только тех точках плоскости XOY, которые принадлежат заштрихованной области, включая ее границы.

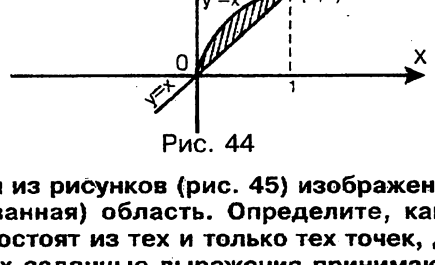


Рис. 44

III. На каждом из рисунков (рис. 45) изображена некоторая (заштрихованная) область. Определите, какие из этих областей состоят из тех и только тех точек, для координат которых заданные выражения принимают истинные значения:

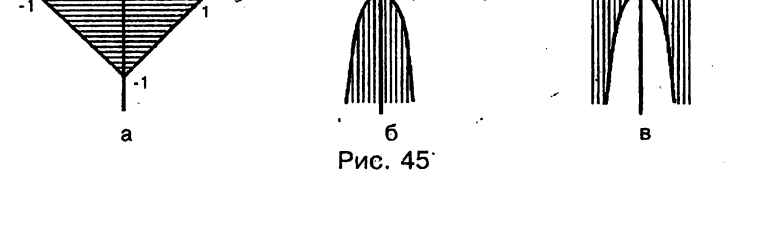


Рис. 45

а) $Y \le -X^2$ или $Y > 0$;
б) $|Y| + |X| \le 1$;
в) NOT ($Y \le -X^2$) или $Y \ge 0$ или $X = 0$.

IV. Составьте логическое выражение на Бейсике, принимающее значение "истина" (TRUE) во всех точках заштрихованной области, изображенной на рис. 46 и включающей свои граничные точки.

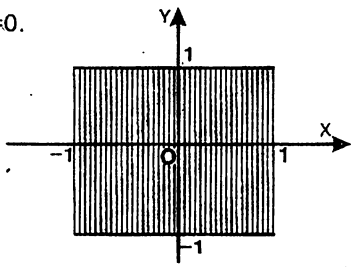


Рис. 46

§ 52. Программирование логических задач

Теперь, зная логические операции, мы можем решить любую задачу, записав ее на языке программирования. Для контроля правильности решения можно воспользоваться и другим способом – решить ее с помощью арифметической модели, хотя этот способ более громоздкий и более трудоемкий.

Именно с него мы и начнем.

В таблице приведены возможные варианты арифметических моделей основных логических операций:

Таблица 16

Название операции	Запись на языке логики	Арифметическая модель
Логическое отрицание	$\overline{A}$	1 – A
Логическое умножение	A∧B	A * B
Логическое сложение	A∨B	A+B–A*B

Составив таблицы истинности, легко убедиться в том, что формулы, выписанные в правом крайнем столбце, дают такие же результаты, как формулы, определяющие основные логические операции. При этом, конечно, все переменные могут принимать только два значения 0 и 1.

Располагая арифметическими моделями отдельных логических операций, можно построить арифметическую модель любой формулы алгебры суждений.

Пример

Пусть дана логическая формула $\overline{A+B+C}$. *Постройте ее арифметическую модель.*

Предварительно данную формулу преобразуем так, чтобы она содержала только логическое умножение и отрицание: получим $A+B+C=\bar{A}*\bar{B}*\bar{C}$.

Модель такого произведения имеет вид:

$$(1 - A) * (1 - B) * (1 - C).$$

Еще пример. *Пусть в текст конкретной программы следует включить вычисление значения логического выражения* $(A+\bar{B}*C)$, *результат необходимо присвоить переменной D.*

Решение может быть таким:

```
15 LET D=(1-(A+(1-B)-A*(1-B))*C=(1-A)*B*C .
```

Решение может быть и таким:

```
15 LET D = NOT (A OR NOT B) AND C .
```

Первое решение чисто арифметическое, но D получит те же значения, что и при вычислении логического выражения.

□

Среди часто встречающихся задач в алгебре суждений важной является задача вычисления всех возможных значений конкретного логического выражения. Это задача построения полной таблицы истинности.

Решение задачи осуществляется в два этапа:

– построение арифметической модели данного логического суждения – этот этап выполняет человек;

– работа с арифметической моделью – этот этап выполняет ЭВМ по программе.

Что касается построения самих арифметических моделей, то здесь следует иметь в виду, что каждое суждение следует записывать исключительно с помощью логического умножения и отрицания.

Решим задачу:

Четыре ученицы – Анита, Бригитта, Криста и Дана – закончили между собой соревнование. На вопрос, кто какое место занял, получены такие суждения:

1. "Анита победила, а Бригитта заняла второе место".
2. "Анита заняла второе место, а Криста третье".
3. "Дана заняла второе место, а Криста четвертое".
Как выяснилось позднее, в каждом из высказываний одно утверждение правильно, а другое ложно. Какое место заняла каждая из девочек?

Решение 1. Решим сначала с помощью ариметической модели.

Введем обозначения суждений (буквой обозначим имя участницы, цифрой – ее место).

А1 – "АНИТА ЗАНЯЛА ПЕРВОЕ МЕСТО";
В2 = "БРИГИТТА ЗАНЯЛА ВТОРОЕ МЕСТО";
А2 = "АНИТА ЗАНЯЛА ВТОРОЕ МЕСТО";
К3 = "КРИСТА ЗАНЯЛА ТРЕТЬЕ МЕСТО";
D2 = "ДАНА ЗАНЯЛА ВТОРОЕ МЕСТО";
К4 – "КРИСТА ЗАНЯЛА ЧЕТВЕРТОЕ МЕСТО".

Выпишем суждения, учитывая, что каждое из них истинно лишь один раз:

$$(A1*\overline{B2}) + (\overline{A1}*B2) = 1$$
$$(A2*\overline{K3}) + (\overline{A2}*K3) = 1$$
$$(D2*\overline{K4}) + (\overline{D2}*K4) = 1$$

Составляем из полученных суждений логическое произведение:

$$((A1*\overline{B2})+(\overline{A1}*B2))*((A2*\overline{K3})+(\overline{A2}*K3))*((D2*\overline{K4})+(\overline{D2}*K4)).$$

Учитываем, что

$$A1*A2=0 \text{ (или } (\overline{A1}+\overline{A2})=1)$$

и

$$K3*K4=0 \text{ (или } (\overline{K3}+\overline{K4})=1),$$

т.к. одна ученица может занять лишь одно место, и

$$B2*A2=0 \text{ (или } (\overline{B2}+\overline{A2})=1)$$
$$B2*D2=0 \text{ (или } (\overline{B2}+\overline{D2})=1)$$
$$A2*D2=0 \text{ (или } (\overline{A2}+\overline{D2})=1),$$

т.к. только одна из учениц заняла второе место.

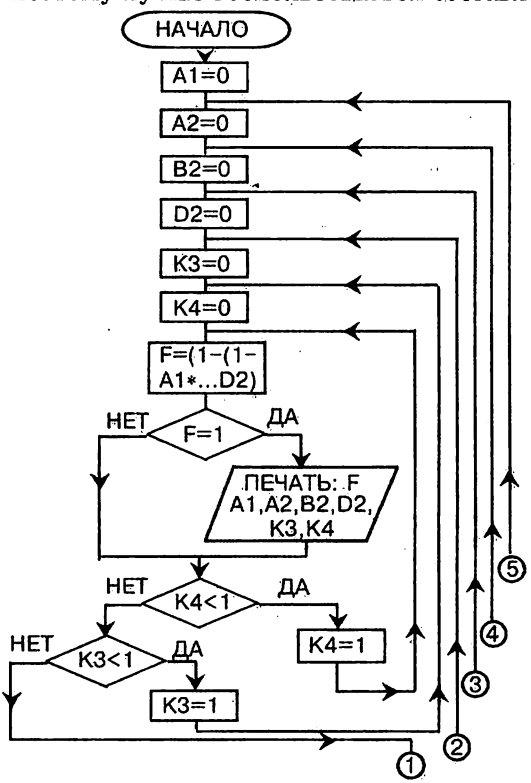
Получаем сложное суждение:

$$((A1*\overline{B2})+(\overline{A1}*B2))*((A2*\overline{K3})+(\overline{A2}*K3))*((D2*\overline{K4})+(\overline{D2}*K4))$$
$$*(\overline{A1}+\overline{A2})*(\overline{K3}+\overline{K4})*(\overline{B2}+\overline{A2})*(\overline{B2}+\overline{D2})*(\overline{A2}+\overline{D2}) = F.$$

Соответствующая арифметическая модель получается достаточно громоздкая, но составлять ее нетрудно, т.к. каждый сомножитель этого выражения преобразуется по одному и тому же правилу:

$$F = (1-(1-A1*(1-B2))*(1-B2*(1-A1)))*(1-(1-A2*(1-K3))*$$
$$*(1-K3*(1-A2)))*(1-(1-D2*(1-K4))*(1-K4*(1-D2)))*$$
$$*(1-A1*A2)*(1-K3*K4)*(1-B2*A2)*(1-B2*D2)*(1-A2*D2)$$

Решить задачу теперь – это значит найти, при каких значениях переменных A1, A2, B2, D2, K3, K4 арифметическое выражение равно единице. Иначе говоря, необходимо заполнить таблицу истинности и найти в ней ту единственную строку, в которой F=1. А мы это уже умеем делать. Правда, решение этой задачи имеет громоздкий характер, поэтому лучше воспользоваться составленной программой:




```
10 FOR A1=0 TO 1
20 FOR A2=0 TO 1
30 FOR B2=0 TO 1
40 FOR D2=0 TO 1
50 FOR K3=0 TO 1
60 FOR K4=0 TO 1
70 F=(1-(1-A1*
*(1-B2))*(1-B2*
(1-A1)))*(1-(1-A2*
*(1-K3))*(1-K3*
*(1-A2)))*(1-(1-
D2*(1-K4))*(1-K4*
*(1-D2)))*(1-A1*
*A2)*(1-K3*K4)
*(1-B2 *A2)*(1-B2
*D2)*(1-A2*D2)
80 IF F=1
THEN 170
90 NEXT K4
100 NEXT K3
110 NEXT D2
120 NEXT B2
130 NEXT A2
140 NEXT A1
150 PRINT -
"ЗАДАЧА РЕШЕНИЙ
НЕ ИМЕЕТ"
```

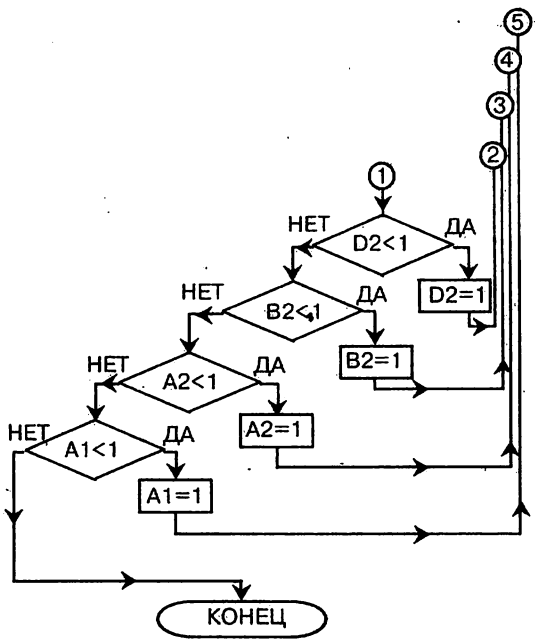


Рис. 47

Решение логических задач с помощью арифметической модели громоздко и достаточно запутанно. Но мы его все же привели, поскольку этот метод чаще всего используют как проверочный для основного метода.

Решение 2. Введем те же обозначения суждений, что и в решении 1.

Понятно, что ученицы не могут в соревновании занимать два места по итогам соревнования и одно место не могут занимать две и более участницы. Такие суждения, если они нам встретятся, будут ложными. Так как в каждом утверждении одно суждение истинное, а второе ложное, то их дизъюнкция будет истинной. Запишем это:

1. A1 ; B2
2. A2 ; K3
3. D2 ; K4

ЧТО может КОМПЬЮТЕР

Компьютер–видеомагнитофон

Резкая активизация деловой жизни в нашей стране привела к расширению функции средств массовой информации. Телевидение, например, начало использовать такую форму, как телебиржа. Этот очень емкий и оперативный способ передачи информации использует стандартный видеосигнал, позволяющий в каждом телевизионном кадре разместить текстовую информацию об одном объекте купли-продажи. Таким образом, всего за десять секунд эфир-

1. A1+K3=1
2. A2+B2=1
3. D2+K4=1

Из этих истинных суждений образуем конъюнкцию, которая также будет истинной:

первое на второе:

$$A1*K3+A2*B2=1;$$

но, т.к. второе место не могут одновременно занять и Анита и Бригитта, A2*B2=0, остается A1*K3=1.

А теперь:

1. A1*K3
3. D2+K4.

Первое на третье:

$$A1*K3*D2+A1*K3*K4=1,$$

(A1*K3*K4 равно 0)

но Криста одновременно не может занять и 3-е и 4-е места, тогда A1*K3*K4=0, остается лишь A1*D2*K3=1.

Мы получили истинную конъюнкцию, из которой следует истинность каждого из составляющих элементарных суждений, т.е.

$$A1=1;\ D2=1;\ K3=1.$$

Итак, первое место заняла Анита, второе – Дана, третье – Криста, а следовательно, Бригитта – четвертое.

Но весь этот результат не обязательно просчитывать, достаточно составить логические выражения и поставить их в уже составленную нами программу. Введем строки:

```
61 F1=(A1 AND NOT B2) OR (NOT A1
AND B2)
```

```
62 F2=(A2 AND NOT K3) OR (NOT A2
AND K3)
63 F3=(D2 AND NOT K4) OR (NOT D2
AND K4)
64 F4=(NOT A1 OR NOT A2) AND (NOT
K3 OR NOT K4)
65 F5=(NOT B2 OR NOT A2) AND (NOT
B2 OR NOT D2)
66 F6=(NOT A2 OR NOT D2)
67 F7=F1 AND F2 AND F3
68 F8=F4 AND F5 AND F6
70 F=F7 AND F8
```

Тогда, дав команду RUN на выполнение, мы на экране получим тот же результат.

Итак, можно сделать вывод, что при разработке программ возможно использование арифметических моделей логических операций. В этом случае арифметика помогает логике. Начинающие программисты избегают использования в своих программах элементов логики, а зря – можно указать много примеров хорошего совмещения в одной программе арифметических и логических операций.

Тренировка

Шесть спортсменов – Адамов, Белов, Ветров, Глебов, Дронов и Ершов – в проходившем соревновании заняли первые шесть мест, причем ни одно место не было разделено между ними. О том, кто какое место занял, были получены такие высказывания:

1. "Кажется, первым был Адамов, а вторым – Дронов".

ного времени перед глазами изумленного телезрителя промелькнет двести пятьдесят полных описаний разнообразных товаров.

Ознакомиться с ними в таком темпе невозможно, даже обладая навыками скорочтения. Но, записав информацию на видеомагнитофон, впоследствии можно внимательно изучить ее в режиме покадрового просмотра.

Новая цифровая видеосистема, созданная японскими фирмами, позволяет обойтись без видеомагнитофона. Это устройство позволяет запоминать на компакт-диске и обрабатывать видео- и аудиоинформацию и воспроизводить ее при необходимости на экране телевизора. Кроме того, с помощью новой системы можно формировать титры и различные видеозаставки для домашнего употребления.

2. "Нет, на первом месте был Ершов, а на втором – Глебов".

3. "Вот так болельщики! Ведь Глебов был на третьем месте, а Белов – на четвертом".

4. "И вовсе было не так: Белов был пятым, а Адамов – вторым".

5. "Вы все перепутали: пятым был Дронов, а перед ним – Ветров".

Известно, что в высказывании каждого болельщика одно утверждение истинное, а второе ложное. Определите, какое место занял каждый из спортсменов.

Составьте программу для решения.

ПОДВЕДЕМ ИТОГИ

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

К помощи логики прибегает человек, распутывая противоречивые показания, составляя различные расписания и во многих других случаях.

В основе теории создания и работы дискретных преобразователей информации (вентили, сумматоры, триггеры и т.д.) лежат аппарат алгебры логики, сведения о двоичной арифметике и теории кодирования (в самой элементарной ее части).

- Логика оперирует **суждениями** (высказываниями), которые могут быть либо **истинными**, либо **ложными**.
- Суждения могут быть **частные** и **общие**, **простые** и **сложные**.
- В алгебре высказываний над простыми суждениями определены следующие операции:
- **Логическое умножение (конъюнкция)** – соединение двух простых суждений в одно сложное с помощью союза "И".

☝ **Логическое сложение (дизъюнкция)** – соединение двух простых суждений в одно сложное с помощью союза "ИЛИ". Дизъюнкция бывает **объединяющая и разъединяющая.**

☝ **Логическое следование (импликация)** – соединение двух простых суждений в одно с использованием оборота речи " ЕСЛИ.., ТО...".

☝ **Эквивалентность** – соединение двух простых суждений в одно с использованием оборота речи или, как принято говорить, связки "... ТОГДА И ТОЛЬКО ТОГДА...".

Все эти четыре операции называют двухместными, т.е. они выполняются над двумя суждениями. В алгебре высказываний определена и широко используется одна одноместная операция:

☝ **Логическое отрицание** – присоединение частицы "НЕ" к сказуемому данного простого суждения. Иногда используют присоединение слов "неверно, что..." к исходному суждению, в результате чего получают новое суждение, являющееся результатом логического отрицания.

☝ Если из двух простых суждений выводится третье, то этот процесс (и результат его) называют **умозаключением.**

☝ Существует несколько способов вывода умозаключения:

– **аналогия;**

– **индукция;**

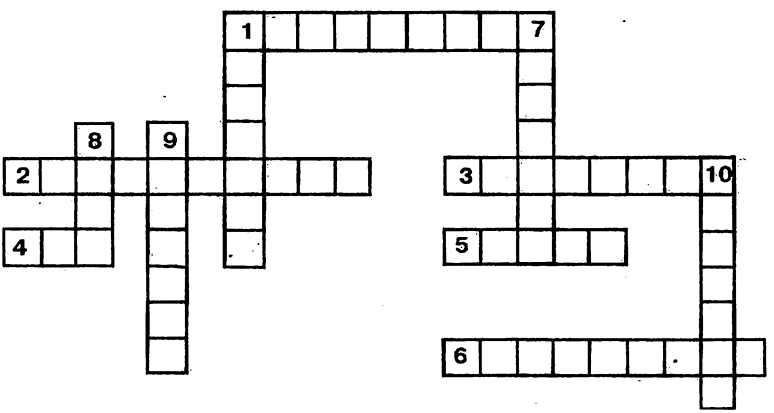
– **дедукция.**

☝ Математической основой алгебры высказываний служит **Булева алгебра.**

В жизни некоторые суждения и связи между ними бывают столь противоречивыми, что такие твердые логические орешки не под силу раскусить даже вдумчивому математику. Тогда на помощь в решении таких логических задач привлекают ЭВМ. Необходимо подчеркнуть, что умение использовать логические операции (OR, AND, NOT и др.) повышает эффективность программирования. Именно

формируя условия в операторе условной передачи управления (IF...THEN), программист использует логические операции.

Кроссворд "Принципы устройства ЭВМ"



По горизонтали: 1. Центральное устройство ЭВМ. 2. Устройство для ввода информации. 3. Устройство для записи и чтения информации с внешнего носителя. 4. Минимальная единица информации. 5. Номер ячейки памяти. 6. Машина для автоматической обработки информации.

По вертикали: 1. Печатающее устройство. 7. Кратковременное записывающее устройство. 8. 8 Бит. 9. Внешний носитель информации. 10. Устройство вывода информации для восприятия человеком.

Ответы:
По горизонтали: 1. Процессор. 2. Клавиатура. 3. Дисковод. 4. Бит. 5. Адрес. 6. Компьютер.
По вертикали: 1. Принтер. 7. Регистр. 8. Байт. 9. Дискета. 10. Дисплей.

ГЛАВА VII

ЗАКЛЮЧЕНИЕ

"Наш мир есть машина,
и притом величайшая, эффективнейшая, прочнейшая,
прекраснейшая".

Анри Монантейль,
1599 г.

§ 53. Аппаратное обеспечение персональных компьютеров

В этом параграфе мы хотели бы представить вам необходимую информацию о компьютерной аппаратуре, достаточную для того, чтобы разобраться в терминологии и общепринятых обозначениях, связанных с нею. Напомним, что оборудование, составляющее ПК, называется **аппаратным обеспечением**. Здесь мы опишем различные типы аппаратных средств, входящих в состав компьютерной системы, с кратким экскурсом в историю их создания.

В том или ином виде эти средства присутствуют во всех компьютерах.

Глушков Виктор Михайлович (1923-1982)

Советский украинский ученый в области кибернетики, вычислительной техники и систем управления.

Родился в г. Ростове-на-Дону. Был директором вычислительного центра и им же организованного Института кибернетики в Украине.

Ему принадлежит идея создания универсальной управляемой ЭВМ и операционной системы реального времени.

Под его руководством созданы первая в стране (тогда СССР) управляющая ЭВМ "Днепр", серия ЭВМ "Мир". Он автор общей теории автоматов и дискретных преобразований. Глушков В.М. первым предложил принципиально новый подход к решению оптимизационных задач в планировании и управлении, создал новые методы прогнозирования.

Процессоры

В ряду гениальных изобретений человека программируемый логический блок по имени "микропроцессор" занимает видное место. "Отлитая" в кремнии идея управляемого логического вычислительного блока была представлена общественности 15 ноября 1971 года.

Импульс был дан в Японии (исходил от японского инженера Масатоси Сима из исследовательского отдела фирмы "Busicom"), но историю **микропроцессоров** (МП) открыли американские фирмы. Началась история с того времени, когда фирма INTEL выпустила МП серии 4004 – "интегральное микропрограммируемое вычислительное устройство", представляющее собой однокристальный центральный процессор, имеющий в своем составе 4-разрядный сумматор и стек. Он был реализован на 2300 транзисторах и мог выполнять 45 различных команд. Последующие поколения МП, представляющие собой 8-, 16- и 32-разрядные приборы, появились соответственно в 1972, 1974 и 1981 годах. И, наконец, пользователям был представлен 64-разрядный МП, содержащий несколько миллионов транзисторов (1993 год).

При оценке параметров микропроцессорной серии наибольшую роль играет **разрядность** прибора, которая задает объем обрабатываемых данных, – чем выше разрядность, тем выше производительность и шире возможности адресации. Быстродействие МП определяется также его **тактовой частотой.**

Итак, в 1972 году появился "всесильный" 8-разрядный МП фирмы INTEL серии 8080. Он содержал уже 4800 транзисторов, выполняя 75 команд, адресовал 64 кбайт памяти. Все посчитали, что предел достигнут. Ни одна программа, думало большинство в то время, не может быть такой большой! Этот МП больше походил на вычислительную машину, чем все те, что были до него; с точки зрения аппаратуры, был проще в применении.

Затем в 1976 году фирма ZILOG создала МП под названием Z80 с тактовой частотой 4 МГц (он был также 8-разрядный), а в 1978 году та же фирма INTEL выпустила МП нового поколения INTEL 8086 – он уже был 16-разрядным. В 1982 году он был улучшен и получил название INTEL

80286 (иногда его просто называют 286), теперь уже с тактовой частотой 6 и 8 МГц. Корпорация IBM, выпускающая на их основе ПК, создала свои знаменитые модели (соответственно IBM PC XT (eXtended Technology) и IBM PC AT (Advanced Technology).

Незадолго до этого (соответственно в 1974 и 1980 гг.) фирма MOTOROLA выпустила МП серии: 6800 и 68000, ставшие "сердцем" машин известного семейства APPLE.

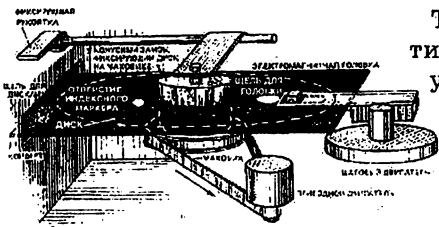
Дальнейшее развитие процессоров – это МП серии INTEL-80386SX (386), ранее известный как P9. Этот процессор можно рассматривать как быструю микросхему 80286 (производительность его в 3-4 раза выше), которая может запускать уже 32-битные программы. Затем появилась серия INTEL-80386DX, выпускающаяся в четырех модификациях – 16, 20, 25 и 33 МГц.

А в 1980 году увидел свет МП серии INTEL-80486DX (или просто 486), имеющий теперь производительность еще в 3-4 раза выше своего предшественника.

Венцом этого парада новинок стала серия INTEL-586, названная "PENTIUM", кодовое название P5 (1993 г.) . А осенью 1995 г. поступил в продажу PENTIUM PRO, использующийся преимущественно в серверных системах. На 1997 год намечено выпустить новый МП фирмы Intel, совместно с Hewlett-Packard, называемый пока условно P7. Его быстродействие, вероятно, достигнет миллиарда операций в секунду!

Дисководы

Дисководы – это устройства, обеспечивающие запись и считывание программ и данных с использованием магнитных дисков. Данные на дисках сохраняются при отключении электропитания. Два наиболее распространенных типа дисководов предназначены для гибких дисков (флоппи-дисков) и винчестерских (жестких) дисков. Для персональных компьютеров фирмы IBM емкость флоппи-дисков колеблется от 300 кбайт (IBM PC AT) до 2 Мбайт. Емкость винчестерских дисков значительно больше: в интервале от 10 Мбайт до 2 Гбайт (и даже выше).

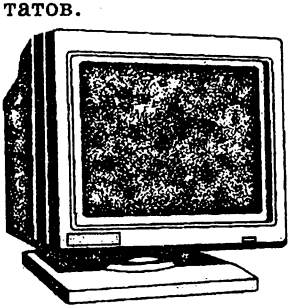


Устройство дисковода для флопи-дисков

Технические характеристики дисководов активно улучшаются. С каждым новшеством хранение информации становится дешевле, а размеры устройства меньше.

Дисплеи

Дисплей (монитор) является основным устройством, используемым для отображения данных и вывода результатов.



Мониторы бывают черно-белыми (их также называют монохромными) и цветными. Они могут работать в одном из двух режимов: текстовом и графическом. Мониторы соединяются с персональным компьютером через специальную дисплейную плату – адаптер, которая управляет высвечиванием на экране информации.

АДАПТЕР – это обобщенное название электронных устройств для сопряжения различных компонентов автоматической аппаратуры. В компьютере адаптер представляет собой специальную микросхему, предназначенную для согласования центральных и периферийных (внешних) устройств и для управления работой последних.

Четкость изображения на дисплее зависит от числа и плотности расположения точек растра. Чем больше таких точек, тем лучше **разрешение**. Например, выражение "разрешающая способность 640*480" означает, что монитор в данном режиме выводит 640 точек по горизонтали и 480 точек по вертикали. Следует заметить, что разрешающая способность не зависит от размера экрана монитора.

Существует несколько стандартов на дисплейную графику IBM PC (табл. 17):

1. Монохромный дисплей с графическим адаптером HERCULES.

2. Цветной монитор RGB (где R-READ – красный, G-GREEN – зеленый, B-BLUE – синий) с платой CGA (COLOR GRAPHICS ADAPTER – цветной графический адаптер (среднего разрешения).

3. С платой EGA (ENHANED GRAPHICS ADAPTER – усовершенствованный графический адаптер высокого разрешения).

С платой MCGA (MULTI-COLOR GRAFICS ADAPTER – мульти-цветной графический адаптер).

4. С платой VGA (VIDEO GRAFICS ADAPTER – видеографический адаптер);

С платой PGA (PROFESSION GRAPHCS ADAPTER – профессиональный графический адаптер).

5. С платой SVGA (SUPER VIDEO GRAPHICS ADAPTER – супервидеографический адаптер).

6. С платой UVGA (ULTRA VIDEO GRAPHICS ADAPTER – ультравидеографический адаптер).

Таблица 17

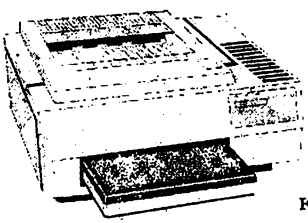
Монитор	Текстовый режим	Графический режим
1. NERCULES	80*25 2 цвета	720*348 2 цвета
2. CGA	80*25 16 цветов	640*200 4 цвета 720*348 2 цвета
3. EGA	80*25 16 цветов	640*350 16 цветов
MGBA	80*43 16 цветов	640*350 16 цветов
4. VGA	80*25 16 цветов 80*50 16 цветов	640*480 16 цветов
5. SVGA		800*600 256 цветов
6. UVGA		1600*1200

Мониторы с адаптерами первых стандартов используются еще относительно редко, т.к. они не обладают надлежащей разрешающей способностью, что приводит к быстрому утомлению глаз.

Важной характеристикой монитора и его адаптера является также скорость работы. В текстовом режиме все мониторы работают достаточно быстро, но при выводе гра-

фических изображений с высокой разрешающей способностью скорость работы может быть весьма существенна. В приложениях с интенсивным использованием графики (обработкой изображений, анимацией, видеоизображением и т.д.) необходимо использование "быстрых" мониторов и адаптеров.

Печатающие устройства



Печатающие устройства позволяют выводить информацию на бумагу. В настоящее время выпускаются печатающие устройства для персональных компьютеров четырех основных видов: матричные, с высоким качеством печати, струйные и лазерные.

Матричные и высококачественные печатающие устройства используют ударную технику печати: изображение на бумаге формируется за счет прижатия к ней красящей ленты. Струйные печатающие устройства для получения символов "выстреливают" на бумагу чернильную струю. Лазерные же устройства "выстреливают" на чувствительную поверхность пучок лазерного излучения. При этом участки, на которые попал луч лазера, обугливаются. К получившейся матрице прижимается лист бумаги, и на нем остается четкий отпечаток.

Матричное печатающее устройство имеет печатающую головку, представляющую собой матрицу из отдельных иголочек. Таким образом, на бумаге образуются символы, состоящие из точек-отпечатков, оставляемых ударами иголочек по красящей ленте (рис. 48).

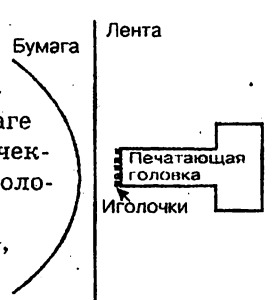


Рис. 48. Матричный принтер

Чем больше точек образует символ, тем выше качество печати матричного устройства. Четкость печати новейших моделей почти не уступает той, что дают устройства для высококачественной печати. При этом сами уст-

ройства матричной печати значительно дешевле и обладают большим быстродействием (100-500 символов в секунду при матричной печати против 10-60 символов в секунду при печати высокого качества). Матричные печатающие устройства также могут быть использованы для получения на бумаге точечных копий графической информации, отображенной на экране дисплея.

Устройство для высококачественной печати иначе называют устройством с лепестковым шрифтоносителем ("ромашка") – по форме печатающей головки. Она позволяет печатать тексты, которые почти не отличаются по качеству от выполненных на пишущей машинке. Это наилучшее качество, доступное для компьютера. Печатающая головка в этом устройстве сменная. Она представляет собой диск с радикально расходящимися лепестками. На каждом лепестке закреплена литера буквы или знака. Специальный молоточек ударяет по лепестку, прижимает его к красящей ленте и бумаге. Для получения отпечатков различных литер диск с лепестками вращается. Большинство таких устройств может печатать 15 символов в секунду, некоторые могут делать 30-40 символов.

Интересны **струйные печатающие устройства**, при использовании которых изображение возникает в результате распыления чернил по бумаге (рис. 49).

При этом подходе пишущее устройство не находится в постоянном соприкосновении с твердой поверхностью, а потому изнашивается не скоро и работает почти бесшумно. Качество печати зависит только от того, насколько точно чернила нанесены на бумагу. Струйные устройства имеют примерно те же качество и скорость печати, что и низкоскоростные матричные. Эта техника, однако, находится на самом раннем этапе развития, так что не удивляйтесь, если в ближайшем будущем она обеспечит и высококачественную печать, и повышенное быстродействие. Однако

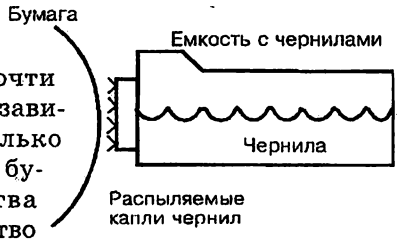


Рис. 49. Струйный принтер

ей придется выдерживать конкуренцию с лазерными печатающими устройствами.

Лазерная печать использует маломощный полупроводниковый лазер, формирующий изображение на светочувствительном фотоприемном барабане (рис. 50).

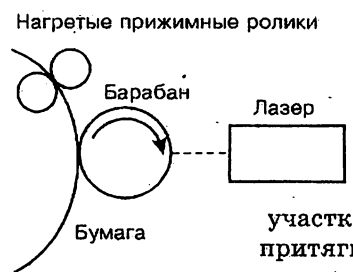


Рис. 50. Лазерный принтер

Барабану предварительно сообщается статический заряд. Когда луч лазера попадает на барабан, заряд стекает с поверхности. Освещаемые лазером участки барабана притягивают (или не притягивают – в другом варианте построения изображения) частицы порошкообразного тонера (тонер как раз и есть довольно липкий порошок), и те пристают к барабану в нужных местах, создавая изображение на нем.

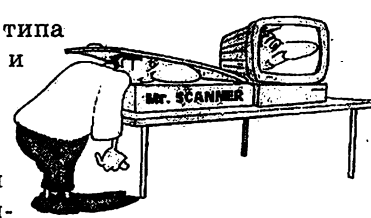
Когда изображение на барабане построено и он покрыт тонером, подающий механизм берет лист и отправляет его в принтер. Дальнейшие процедуры скрыты от глаз пользователя внутри принтера. Лист заряжается таким образом, чтобы тонер с барабана притягивался к бумаге, затем он проводится под барабаном. После этого изображение в виде частичек тонера оказывается на бумаге. Теперь дело за малым – закрепить на бумаге легкий порошок, улетающий при слабом дуновении. Фиксирующий узел прогревает тонер до температуры плавления, а специальные резиновые валики прижимают расплавленный тонер к бумаге. Комбинация нагревания и прижатия позволяет намертво вваривать тонер в бумагу для получения прочного изображения.

Замечательными особенностями этих устройств являются очень высокая скорость и многообразие возможностей печати. Скорость здесь измеряется уже не символами в секунду, а страницами в минуту. Эта техника обещает нам возможность печатать документы различным шрифтом и исполнять рисунки не хуже, чем типографские. Сегодня лучшие модели обеспечивают разрешение печати до 1200 точек на один сантиметр.

Сканеры

Сканером называется устройство, позволяющее вводить в компьютер образы изображений, представленных в виде текста, рисунков, фотографий или другой графической информации.

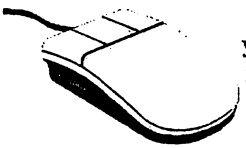
Сканер изображений и **систему оптического распознавания символов (ОРС)** можно представить себе как своего рода фотоаппарат, который делает "снимок" того, что изображено на бумаге. Однако после щелчка такого фотоаппарата изображение не попадает на фотопленку, а преобразуется в поток цифровых сигналов и через интерфейс – в файл особого формата. Затем этот файл обрабатывается программным обеспечением, т.е. его можно редактировать, распечатывать и т.п.



Известны два основных типа сканеров: **ручной** (hand held) и **настольный** (desttop). Ручной сканер чем-то напоминает увеличенную в размерах электробритву. Для того чтобы ввести в компьютер какой-либо документ при помощи этого устройства, надо без резких движений провести сканирующей головкой (находящейся там, где обычно у электробритвы "ножи") по сканируемому изображению. Ширина вводимого изображения для ручных сканеров не превышает обычно 15 см, а длина ограничивается объемом памяти вашего компьютера.

Настольные сканеры чем-то напоминают копировальные машины – "ксероксы". Для сканирования изображения (чего-нибудь) необходимо открыть крышку сканера, положить сканируемый лист изображением вниз и закрыть ее. Все дальнейшее управление процессом сканирования осуществляется с клавиатуры – при помощи специальной программы. Подобная конструкция позволяет сканировать не только отдельные листы, но и страницы журналов или книг.

"Мышь"



"Мышь" – это специальное устройство-указатель, или манипулятор, позволяющее просто и быстро передвигать курсор по экрану. Название объясняется наличием у устройства длинного "хвоста" – провода, соединяющего его с компьютером. "Мышь" стала особенно популярна с появлением программ, ориентированных на полноэкранный диалог с пользователем в графическом режиме интерфейса. "Мышь" позволяет передвигать курсор в нужное место экрана и фиксировать выбор нажатием одной из кнопок на своей поверхности.

Существуют два основных варианта конструкции "мыши": механический и оптический. Механическое устройство использует свободно вращающийся шарик, который располагается на "дне мыши" или на ее поверхности (в таком случае она называется "трекбол"). Шарик в результате трения проворачивается, когда "мышь" двигают по плоской поверхности (или вращают его рукой – в трекболе). Схемы "мыши" воспринимают это, подсчитывают число оборотов и передают информацию компьютеру.

Модем

Потребность в передаче информации между компьютерами возникает достаточно часто, именно по этой причине компьютеры снабжают коммуникационными портами.

Если компьютеры находятся в одном помещении (или в одном здании), то можно их соединить локальной сетью, а если они находятся по разные стороны улицы или в разных городах и даже в разных странах, то соединить их кабелем будет затруднительно.

Но все же работа эта уже проделана – весь мир опутан телефонными линиями! К сожалению, телефонные каналы были непредусмотрительно разработаны только для передачи сигналов с частотой звукового диапазона. Наш компьютер работает только с нулями и единицами.

Следовательно, дело за малым – необходимо устройство, которое могло бы превращать цифровые биты в ана-

логовые сигналы для передачи по телефонным линиям, а аналоговые сигналы – обратно в цифровые биты для последующей обработки компьютером. Преобразование дискретной информации в звуки называется модуляцией, обратный процесс – демодуляцией.

Поэтому прибор, который выполняет модуляцию и демодуляцию, называется **модемом**.

Вы просто подключаете модем к компьютеру и телефонной розетке – и связь обеспечена. Правда, для этого еще необходимо соответствующее программное обеспечение.

§ 54. Мультимедиа. CD ROM технологии

Под МУЛЬТИМЕДИА понимают такой способ и такие средства донесения информации до потребителя, при которых используется несколько сред, например, компьютерная графика, фотография, фрагменты видео, текст, звуковое сопровождение.

Такой подход предъявляет высокие требования к носителю информации по объему и быстродействию, а также к средствам вывода информации. Таким требованиям, с учетом доступной цены, на сегодня отвечает комплекс мультимедиа, который включает в себя следующие компоненты (рис. 51):

– персональный компьютер;
– проигрыватель компакт-дисков (CD ROM DRIVE);
– звуковая плата с наушниками или динамиками;
(видеоплата с видеокамерой);
– компакт-диски (CD ROM) с записанными на них программами.

Известно ли вам, что...

для снятия зрительного утомления при работе с экраном дисплея разработано несколько комплексов упражнений. Вот один из них.

Сядьте в удобную позу, позвоночник прямой, глаза открыты, взгляд устремлен прямо. Выполнять упражнения легко, без напряжения в течение 3-5 мин.

1. Взгляд направить влево-вправо, вправо-прямо, вверх-прямо, вниз-прямо, без задержки в отведенном положении. Повторять от 1 до 10 раз.

2. Взгляд смещать по диагонали, постепенно увеличивать задержки в отведенном положении, дыхание произвольное, но следить, чтобы не было его задержки. Повторять 1-10 раз.

3. Круговое движение глаз: от 1 до 10 кругов влево и вправо.

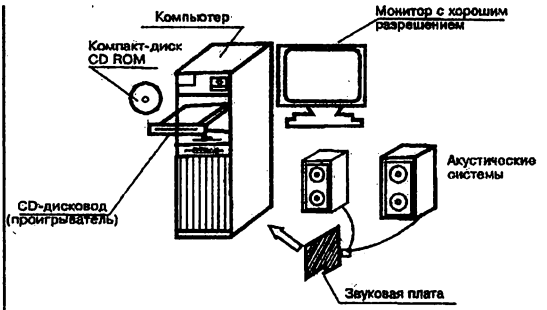


Рис. 51. Минимальный комплект мультимедиа

Рассмотрим более подробно каждый из компонентов. В качестве персонального компьютера лучше использовать широко распространенные модели, совместимые с IBM PC AT (или APPLE-II) и выше. Желательно, чтобы процессор был класса INTEL 80386 и выше, оперативная память не менее 2 Мбайт. Для вывода визуальной информации требуются мониторы с хорошим разрешением, как правило, не ниже 640*480 точек экран, 256 цветов. Такой режим обеспечивается видеоадаптером SVGA (и выше).

CD ROM (COMPACT DISK ONLY MEMERY – компакт-диск только для чтения) имеет 12 см в диаметре и позволяет хранить до 640 Мбайт цифровой информации. Как правило, сама информация заносится на диск на заводе и далее не может быть изменена.

Отметим ряд существенных достоинств компакт-дисков.

1. При малых физических размерах CD ROM обладают высокой информационной емкостью, что позволяет использовать их в системах мультимедиа,

справочных системах со сложной разветвленной структурой данных, учебных комплексах с богатым иллюстративным материалом, в том числе с графическими базами данных. Для наглядности можно отметить, что один компакт-диск CD ROM, имея размеры примерно дискеты, по информационному объему эквивалентен почти пятистам таким дискетам!

2. Считывание информации с компакт-дисков происходит с высокой скоростью, сравнимой со скоростью работы винчестера.

3. Компакт-диски просты и удобны в работе: для смены программного обеспечения достаточно сменить CD ROM (как пластинку в проигрывателе). Перезапись винчестера в таких объемах – достаточно длительная операция.

4. Компакт-диски практически не изнашиваются.

5. На компакт-диске невозможно случайно стереть информацию.

6. Компакт-диски не могут быть поражены вирусами.

В отличие от, например, винчестеров, дорожки которых представляют концентрические окружности, компакт-диск имеет всего одну физическую дорожку в форме непрерывной спирали, идущей от наружного диаметра диска к внутреннему. Тем не менее одна физическая дорожка может быть разбита на несколько логических.

При записи компакт-диска (обычно эта операция производится на заводе) со специальной матрицы (мастер диска) для формирования микрорельефа поверхности применяют лазер высокой

Вначале быстрее, потом как можно медленнее.

4. Изменение фокусного расстояния: смотреть на кончик носа, затем вдаль. Повторить несколько раз.

5. Смотреть на кончик пальца или карандаша, удерживаемого на расстоянии 30 см от глаз, а затем вдаль. Повторить несколько раз.

Смотреть прямо перед собой пристально и неподвижно, стараясь видеть более ясно, затем моргнуть несколько раз. Сжать веки, затем моргнуть несколько раз.

6. Потереть ладони друг о друга и легко, без усилий, прикрыть ими предварительно закрытые глаза, чтобы полностью загородить их от света (на 1 мин.). Представить погружение в полную темноту. Открыть глаза.

мощности. Узкий пучок излучения, попадая на гладкую поверхность пустого диска, оставляет на ней микроскопические углубления или пузырьки (в зависимости от особенностей конструкции). При считывании эти углубления регистрируются при помощи менее мощного лазера, сканирующего поверхность.

Преимущества оптического диска перед магнитными носителями связаны с микроскопическим размером области, необходимой для хранения одного бита данных. Обычный диск (флоппи) содержит 38 концентрических дорожек на каждый сантиметр, плотность записи на жестком диске (винчестерском) около 300 дорожек на сантиметр. Для оптического эта величина достигает 16000 дорожек на сантиметр.

Считывание с компакт-дисков происходит на так называемом проигрывателе (CD ROM DRIVER) (рис. 52). Работа этого устройства основана на следующем принципе: участки диска, на которых записаны символы двоичного кода "0" или "1", отличаются коэффициентом отражения лазерного луча, посылаемого проигрывателем CD ROM. Эти отличия улавливаются фотоэлементом, и общий сигнал преобразуется в соответствующую последовательность нулей и единиц. Проигрыватель CD ROM проводит первичную обработку считанных сигналов, анализируя и удаляя служебную информацию. Считанные и преобразованные данные проигрыватель CD ROM передает в компьютер через специальный адаптер. Конструктивно проигрыватель CD ROM может быть выполнен в виде отдельного устройства (EXTERNAL CD ROM DRIVE). Его также можно встраивать внутрь корпуса компьютера (INTERNAL CD ROM DRIVE). В последнем случае установочные размеры устройства часто совпадают с дисководом 5,25 дюйма. Обращение компьютера к CD ROM происходит аналогично обращению к другим дискам. Кстати, проигрыватели CD ROM позволяют прослушивать обычные аудиодиски (CD A)

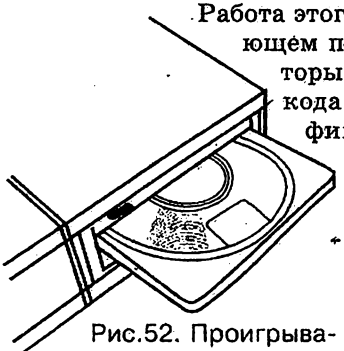


Рис.52. Проигрыватель компакт-дисков (CD ROM DRIVER)

и видеодиски (CD V), что создает дополнительные удобства для пользователя.

И, наконец, звуковая плата (Sound Blacter) представляет собой адаптер, имеющий выходной разъем для подключения наушников или динамиков. Функционально это устройство, которое преобразует цифровую информацию, поступающую от компьютера в аналоговый сигнал, достаточный по мощности для работы наушников или динамиков. Качество воспроизведения звука в значительной мере определяется характеристиками наушников или динамиков, рекомендации по выбору которых такие же, как для бытовой звуковоспроизводящей аппаратуры. Некоторые звуковые платы имеют также линейный выход для подключения усилителя и микрофонный вход. С микрофона звуковой сигнал может быть введен в компьютер, что используется, например, для самоконтроля произношения при изучении иностранных языков. Звуковые файлы можно также вводить в свои компьютерные программы.

В случае необходимости подобные операции можно произвести и с видеоизображениями (снятые видеокамерой), но только теперь – через видеоплату.

В последнее время появились так называемые перезаписываемые компакт-диски CD-R (CD-RECORDABLE – компакт-диск для перезаписи). Носители типа CD-R могут быть записаны на специальном CD-R-приводе. В основном здесь применяются технологии, основанные на изменении отражающих свойств вещества подложки диска под действием луча лазера.

Сравнительно невысокая стоимость компакт-дисков и проигрывателей CD ROM, а также простота в эксплуатации обусловили их широкое распространение в мире. В перерасчете на один бит информации программное обеспечение, поставляемое на CD ROM, в сотни раз дешевле, чем на дискетах. Компании США и Европы (в основном Великобритании), Азии (в основном Китая) выпустили уже тысячи наименований компакт-дисков – от справочников телефонов и телефаксов до всевозможных иллюстрированных энциклопедий, от курсов иностранных языков до развлекательных и познавательных игр. Персональный компьютер, оснащенный CD ROM-проигрывателем, открывает окно в новый информационный мир – мир **мультимедиа**.

§ 55. Вверх по спирали компьютерного совершенствования

Общепринято считать, что создателями первой персональной ЭВМ являются два двадцатилетних студента колледжей США Стив Джобс и Стив Возняк (кстати, предки последнего – выходцы из Украины).

Это происходило так. До 1976 года самой доступной ЭВМ являлся микрокомпьютер ALTAIR, однако, чтобы сделать на этой машине что-либо полезное, требовалось приобрести дополнительные устройства. В итоге цена комплекта возрастала существенно. Три тысячи долларов студентам колледжа было "не поднять", поэтому Стив Возняк, несомненно гениальный изобретатель и программист, решил собрать машину самостоятельно, используя наборы радиолюбителя "сделай сам". Следует сказать, что студенты имели неплохую подготовку – Джобс работал в фирме "АТАРИ" (ATARI), Возняк – в "ХЬЮЛИТ ПАККАРД" (HEWLETT PACKARD).

Созданная в домашней мастерской (в гараже) машина включала в себя клавиатуру, близкую к клавиатуре печатной машинки, и системный блок, реализованный на одной плате. Все это легко размещалось в "дипломате". Первоначально информация отображалась на подключаемом бытовом телевизоре, а затем были разработаны программно-аппаратные средства и к компьютеру был подключен графический дисплей, сначала монохромный, а затем и цветной.

Машина, так же как и фирма, созданная 1 апреля 1976 года, получила название APPLE (от англ. – яблоко). Причин именно так назвать детище было две – Джобс был вегетарианцем и оба увлекались ансамблем BEATLS. Было выпущено три модели APPLE: первая тиражом 200 экз. (сейчас предмет коллекционирования), вторая модель (в производстве с 20 апреля 1977 г. по 1 августа 1983 г.) – 3000 000 экз., третью модель постигла относительная неудача.

А дело в том, что главным конкурентом для вновь созданных ПК стала фирма IBM. В августе 1981 года эта

фирма сообщила о выпуске своего первого персонального компьютера. IBM в ту пору была крупнейшая компьютерная компания, столь огромная, что ее в США уважительно называли "голубой гигант" (из-за использования голубого цвета в торговом знаке фирмы). Она была знаменита своей большой и мощной ЭВМ – SYSTEM 370.

История создания этой фирмы уходит в 1890 год, когда в США проводилась перепись населения. Работу по переписи возглавил Холлерит, который использовал для этой цели перфокарты (см. с.155). Позднее он и создал компанию IBM (International Business Machines).

Президентом подразделения корпорации IBM, разработавшим концепцию персонального компьютера, был Дон Истридж (к сожалению, погибший в авиационной катастрофе в августе 1985 г.).

Если мир IBM-совместимых компьютеров можно назвать "дикорастущим" – он давно уже, вырвавшись из рук своих основателей, развивается как бы сам по себе, – то APPLE продолжает удерживать разработку и производство компьютеров в своих руках. Так что, в отличие от IBM PC, этот мир упорядоченный, спланированный по единому проекту.

Архитектура компьютеров, ОС, даже правила построения, структура пользовательских интерфейсов – все выходит из "мозгового центра" APPLE и передается третьим фирмам – разработчикам, выпускающим аппаратные расширения и продукты для MACINTOSH, причем осуществляется постоянный контроль за соблюдением "правил игры", за соответствием стандарта.

В истории конкуренции этих двух "китов" компьютерного дела много поучительного. Например, модель LIZA фирмы APPLE была объективно лучше машин фирмы IBM. Однако к тому времени могучая корпорация IBM уже контролировала рынок и сумела в короткий срок буквально "завалить" потребителя программными средствами для своих машин – IBM PC. APPLE не имела ресурсов для достойного ответа и очень быстро покинула число ведущих фирм-разработчиков ПЭВМ. Таким образом, судьбу персональных ЭВМ решают не столько удачные аппаратные решения, сколько богатство программных средств.

ЧТО может КОМПЬЮТЕР

Не переводя взгляда

Природа дала человеку два глаза. Но, как правило, они оба "расточительно" смотрят на один объект. А ведь есть немало профессий, где неплохо бы следить хотя бы за двумя объектами одновременно. Одна из них – машинопись. Другая, не менее популярная, – программирование.

Помочь и тем, и другим взялась американская фирма Reflection Technology. Новый монитор ее производства настолько мал и легок, что без неудобств крепится на голове на небольшом расстоянии от одного

Разумеется, не только фирмы IBM и APPLE MACINTOSH сегодня определяют направление стратегического развития, ибо в этом неразрывном процессе кооперирует между собой огромное множество фирм во всех странах мира. Мы все же во главу угла рассмотрения глобального компьютерного направления поставим ПЭВМ фирмы IBM. На примере IBM этот процесс лишь более нагляден, ибо здесь фокусируются, концентрируются и суммируются разрозненные противоречивые тенденции, обретая в конце концов форму **общепризнанного стандарта**. Хотя IBM сегодня не диктует новейшие направления компьютерной моды, но все же без согласия и признания IBM эти оригинальные изыски обычно не приживаются.

Посмотрим на хронику развития семейства персональных компьютеров IBM:

1981 – PC

1982 – PC 02

1983 – PC 03, PC JUNIOR, PC XT

1984 – P1, P2, PC AT 01

1985 – PC XT 286DD, PC AT 02 (512 КБ)

1986 – PC AT, PC XT SDD и PC XT SFD (640 КБ)

1987 – семейство PS/2 модели 30, 50, 60 и 80

1988 – PS/2 модель 70

1989 – PS модель 55 SX, портативный P70, плата 486

1990 – PS/1, PS/2 модели 286 30, 65SX, 75SX, 9XP, 95XP

1991 – 486 SX, LAPTOP L40SX

1993 – PENTIUM...

Не стоит подробно рассматривать особенности каждой из перечисленных ма-

шин. Среди них, безусловно, есть очень удачные модели, хотя были и тупиковые направления. От полупрофессиональных до самых совершенных моделей типа PENTIUM – такова гамма предлагаемых моделей.

Одновременно с развитием и становлением персонального компьютера развивались и совершенствовались его средства управления. В большой степени успеху IBM PC способствовала простая и достаточно удобная дисковая операционная система MS-DOS, которая в компьютерах самой IBM всегда имела "фирменное" наименование PC-DOS. Что бы ни говорили многочисленные критики этой ОС, но без ее массового распространения в качестве безусловного стандарта ни о каком единственном стандарте ПК не могло быть и речи. Именно MS-DOS была тем средством, которое обеспечило полную программную и аппаратную совместимость. Эта система и сейчас не собирается сдавать своих позиций.

Вот некоторые "этапы большого пути" этой ОС:

август 1981 – первая MS-DOS 1.0 установлена на первом IBM PC;

март 1983 – MS-DOS 2.0 (для жесткого диска);

март 1985 – MS-DOS 3.1 (для работы в сетях);

март 1990 – Билл Гейтс представил в Москве русскую версию MS-DOS 4.01;

октябрь 1991 – MS-DOS 5.0.

Проанализировав вышесказанное, можно попробовать вообразить, **что же будет завтра?**

Возможно, это будут машины на процессоре i786 или каком-то ином мощном

из глаз. Теперь вы можете, не поворачивая шеи, рассматривать одновременно изображение, формируемое на дюймовом экране светодиодной матрицей, и текст, лежащий перед вами на столе.

В отличие от подобных уже существующих мини-мониторов, новый, получивший название Private Eye, с помощью настраиваемой системы линз создает иллюзию работы с нормальным 12-дюймовым монитором настольных размеров. Это дает возможность не напрягать зрение, всматриваясь в миниатюрный экран, и даже работать без очков тем, кто пользуется ими.

Что касается удобства одновременного рассматривания двух объектов, фирма обещает быстрое привыкание к необычному стилю работы и даже комфортабельность.

процессоре, возможности которого нельзя вообразить, работающем с молниеносной скоростью в сотни мегагерц. Очень вероятно, что процессор и многие другие компоненты будут сменными, чтобы пользователь не только складывал свою систему из готовых "кирпичиков", но и не страдал от стремительного прогресса в электронике. Объем памяти машин будет очень разным – от 4 Мбайт до сотни Гбайт. На смену нынешней таблице ASCII придет новый стандарт, позволяющий одновременно использовать тысячи символов и алфавитов всех языков народов мира.

Графические возможности возрастут весьма существенно. Графический интерфейс пользователя также сможет радикально преобразиться, интегрируя в себе необыкновенную гибкость в интуитивном приспособлении к нуждам каждого конкретного пользователя. Пользователь сможет легко участвовать в управлении ресурсами машины и данными, невольно превращаясь в программиста, хотя в традиционном понимании для этого не нужно будет изучать каких-либо формальных языков программирования – все формальности кодирования программ сможет осуществлять сама машина своими встроенными аппаратными средствами. Поэтому программирование из науки превратится в искусство, доступное любому пользователю со свободной фантазией и богатым воображением. В интерфейсе пользователя смогут свободно уживаться любые тексты, данные, электронная почта, графика, четкие живые телевизионные изображения, стереозвук. Однако машина сможет выполнять одновременно множество задач, создавая совершенно непривычную сегодня среду пользователя.

Дисплеи большинства машин станут совершенно другими: электроннолучевые трубки, облучающие пользователей вредными лучами и магнитными полями, уступят место более гигиеничным новым плоским дисплеям, размеры которых могут быть любыми в зависимости от необходимости и степени разрешения и практических потребностей пользователя.

Накопители на гибких дисках, видимо, смогут сохраниться, значительно увеличив емкость, а вот жесткие диски скорее всего отправятся в технические музеи, освободив место различным модификациям оптических дисков.

Самым недорогим и распространенным источником информации станут тиражируемые компактные диски CD ROM. Пакетные дисководы для CD ROM могут стать принадлежностью почти каждой машины, вытеснив дорогие и ненадежные жесткие диски.

❐

Разговор был бы неполным, если бы мы не затронули тему биокомпьютера.

Биокомпьютер (или молекулярный компьютер) – это пока еще не существующее логико-вычислительное устройство, которое будет использовать принципы обработки информации, присущие живым организмам, и, в качестве элементной базы, сложные сочетания молекул биологических веществ.

Для начала приведем перечень удивительных фактов из мира живых существ, которые вдохновляют ученых.

Крошечная птичка славка на своем пути в Южную Америку летит сначала в сторону Африки, высоко пересекая Атлантический океан. На высоте 6000 метров она ловит поток воздуха, который несет ее к Южной Америке. Руководимая миграционным инстинктом, она держит свой курс в течение нескольких дней на протяжении 4000 километров – окутанные в перышки 20 граммов храбрости! Нас охватывает восторг и удивление.

Гремучая змея улавливает разницу в температуре, равную тысячной доле градуса, а некоторые рыбы ощущают стомиллиардную долю пахучего вещества в одном литре воды. Это все равно, что уловить присутствие 30 г такого вещества в целом Азовском море!

Крысы ощущают радиацию, обыкновенный черный таракан радиацию видит, а отдельные виды микробов реагируют даже на слабое изменение радиации.

Глубоководные рыбы улавливают изменения плотности тока менее чем на одну стомиллиардную часть ампера, а нильская рыба мормирус с помощью электромагнитных колебаний "прощупывает" свой путь в воде.

Летучие мыши пользуются системой эхолокации, угри производят электричество, чайки опресняют морскую воду; осы изготавливают бумагу; термиты строят воздушные

кондиционеры; осьминоги путешествуют с помощью реактивного двигателя; птицы плетут или строят многоквартирные дома; муравьи занимаются садоводством, шитьем или животноводством; светлячки имеют встроенные в тело фонари. Такая гениальность вызывает в нас восхищение.

Именно с целеустремленного "подглядывания" за природой родилась идея создать биологический компьютер. При этом нужно стремиться не просто к слепому подражанию, к заимствованию всех характеристик биологических объектов, а к критическому, строгому отбору только полезных для техники свойств.

Из всех чудесных вещей на земле ничто не является более удивительным, чем человеческий мозг. Каким образом мозг справляется со 100 миллионами сообщений, которые поступают в него каждую секунду? Как он не перегружается такой лавиной? Если мы в один прием охватываем только одну мысль, то как разум справляется с миллионами одновременных сообщений? Очевидно, разум не только выдерживает этот поток, но и с легкостью управляет им. Такое изучение и лежит в основе создания биокомпьютера.

За последние годы ученые сделали огромные успехи в исследовании мозга. Тем не менее, то, что они узнали – ничто по сравнению с тем, что остается неизвестным. Человеческий мозг является, несомненно, самой таинственной частью чуда – "чуда" в смысле чего-то, что вызывает удивление.

Чудо берет свое начало в матке матери. Три недели спустя после зачатия начинают формироваться клетки мозга. Они растут рывками, иногда до 250000 клеток в минуту. После рождения мозг продолжает расти и формировать свою систему связей. Пропасть, отделяющая человеческий мозг от мозга любого животного, вскоре становится очевидной: в отличие от мозга любого животного, мозг ребенка в продолжение первого года жизни увеличивается втрое. Со временем человеческий мозг, помимо клеток другого типа, вмещает около 100 миллиардов нервных клеток, так называемых нейронов, несмотря на то, что он составляет всего лишь 2 процента веса тела.

Биологи, химики, физики и специалисты в области вычислительной техники надеются, что подобную картину, которая происходит в тканях животных и в коре головного мозга человека, можно будет получить в молекулярном компьютере, состоящем не из интегральных, а из взаимодействующих между собой белков и других сложных молекул. В таком биокомпьютере сигналы обрабатываются не последовательно, бит за битом, а как динамические структуры: молекулы белка узнают другие окружающие их молекулы по пространственной структуре их поверхности.

В молекулярном компьютере вычисления – это взаимодействие белковых молекул с окружающей их физико-химической средой.

Переключателями служат ферменты, а программа, скорее, неявно, чем явно, выражена в структуре белков и самой системы, в которой они интегрированы. Поэтому в биокомпьютере программирование представляет собой эволюционный процесс, осуществляемый с помощью изменений и отбора.

Своей высокой эффективностью биопроцессор обязан широкому параллелизму на молекулярном уровне. Хотя скорость работы считывающего фермента на пять порядков меньше быстродействия существующих обычных переключателей, число цифровых операций обычной ЭВМ, необходимое для воспроизведения процесса распознавания молекул определенного типа, слишком велико, и преимущество в быстродействии остается за биокомпьютером.

Это одно из направлений в создании биокомпьютеров. Существуют и другие направления. Например, создание приборов молекулярной электроники, которые могли бы выполнять функции логических элементов и памяти цифровых ЭВМ.

Итак, компьютер будущего – это компьютер, работающий на принципах биосистем.

Завершим книгу словами поэта: "Читатель избавит меня от излишней обязанности описывать развязку". Так А.С.Пушкин закончил свою "Барышню-крестьянку". Мы же надеемся, что читатель, наоборот, продолжит читать книгу, но теперь в конкретной реальной жизни. Надеемся, что теперь "книгой мудрости" окажется настоящий компьютер, который станет и помощником, и добрым товарищем, и настоящим другом в чудесном и волшебном "городе Солнца"!

Известно ли вам, что...

такое виртуальная реальность?

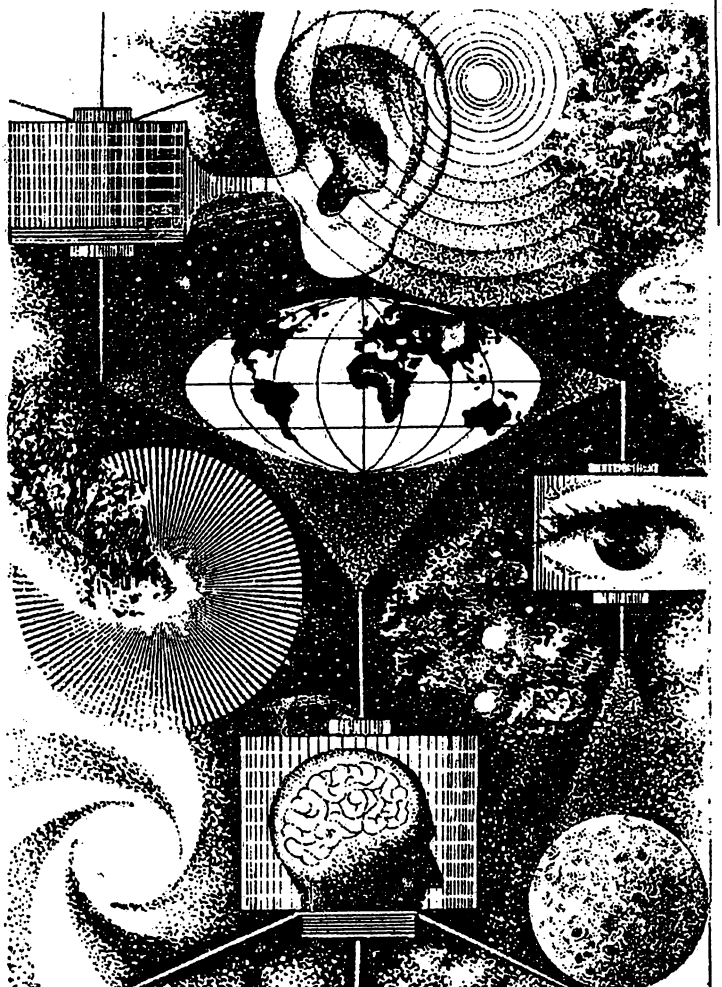
Виртуальная реальность – это новая технология неконтактного информационного взаимодействия, реализующая с помощью комплексных мультмедиа-операционных сред иллюзию непосредственного вхождения и присутствия в реальном времени в стереоскопически представленном "экранном мире".

Пользователь, надев на себя "информационные перчатки" и "информационный костюм", попадает в мир "Виртуальной реальности". Рука пользователя, одетая в такую "интерфейс-перчатку", может быть спроецирована в виртуальной форме в трехмерной компьютерно-генерированной среде. Манипулируя "информационной перчаткой", пользователь может взаимодействовать с виртуальным миром, передвигая и управляя объектами, либо использовать набор жестов в качестве команд. При наличии "информационного костюма", "информационной перчатки", "информационных очков" со встроенными стереоскопическими экранами (очки-телемониторы) пользователь может, образно выражаясь, "шагнуть" прямо в виртуальный мир.

Уже в настоящее время возможности системы "Виртуальная реальность" используются при тренаже спортсменов, профессиональной подготовке специалистов в области астронавтики, архитектуры, медицинской диагностики, организации развлечений и досуга, а также в областях, использующих научную визуализацию. Например, если возможности трехмерной компьютерной графики позволяют осуществлять прогноз результатов хирургической операции, представлять трехмерное изображение на экране компьютера, то использование системы "Виртуальная реальность" позволяет создать иллюзию реально проводимой хирургической операции.

Контакт пользователя с системой "Виртуальная реальность" может осуществляться голосом или с помощью специального устройства – джойстринга, обеспечивающего эффект обратной силовой связи, а также с помощью очков телемониторов.

ОТВЕТЫ И РЕШЕНИЯ



РЕШЕНИЯ К § 7, 8

I. а) 16*8= 128 бит; б) 25*8=200 бит; в) 35*8=280 бит

II. 1 байт = 8 бит; 1 кбайт = 1024 байта = 1024*8 = 8192 бит; 1 мбайт = 1048576*8 = 8388608 бит; 1 гбайт = приближенно 8 млрд. байт

III. *1- й вариант решения*: За один час можно переработать 160*60 = 9600 слов. За 7 час. - 9600*7 = 67200 слов,т.е. приблизительно 67.2 кбайт:2.5 кбайт = 27 страниц

2- й вариант решения: 160 слов * 60 мин. = 9600 (в час); 9600 слов * 7 час = 67200 слов (за семь часов). На 1 стр. = 2500 слов; 67200:2500 = 27 страниц * 2.5 = 67.5 кбайт.

IV. Из расчета 1 страница - это 2500 символов, а на то, чтобы ее прочитать, идет 2500:180 приближенно 14 мин, следовательно 32*1024 = 32768 байт. 32768:180 = 182 мин., т.е. приближенно 3 часа.

V. 24*80 = 1920 байт. Объем памяти программы 19*1024 байт = 19456 байт. Итого информация занимает 1920:19456 = приближенно 0. 1, т.е. прмерно 10%.

РЕШЕНИЯ К § 9

I. а) $567 = 5*10^2 + 6*10^1 + 7*10^0$; б) $842,3 = 8*10^2 + 4*10^1 + 2*10^0 + 3*10^{-1}$; в) $1924,803 = 1*10^3 + 9*10^2 + 2*10^1 + 4*10^0 + 8*10^{-1} + 0*10^{-2} + 3*10^{-3}$; г) $10000 = 1*10^4 = 10^4$; д) $0100,00 = 0*10^3 + 1*10^2 + 0*10^{-2} = 10^2$; е) $0,002 = 2*10^{-3}$.

II. а) $8*10^2 + 5*10^1 + 3*10^0 + 7*10^{-1} + 6*10^{-2} = 853,76$;

б) $0*10^4 + 1*10^3 + 8*10^2 + 4*10^1 + 0*10^0 + 0*10^{-1} + 9*10^{-2} = 01840,09 = 1840,09$

в) $9*10^5 + 4*10^3 + 3*10^0 + 4*10^{-2} + 4*10^{-3} = 904003,044$

III. а) LX = 50 + 10 = 60; б) XL = 50 - 10 = 40; в) CXI = 100 + 10 + 1 = 111; г) IXC = 100 - 10 - 1 = 89; д) MDCCCXII = 1000 + 500 + 100 + 100 + 100 + 10 + 1 + 1 = 1812; е) MCMLXI = 1000 + 1000 - 100 + 50 + 10 + 1 = 1961

IV. а) 45 = 50 − 5 = VL; б) 55 = 50 + 5 = LV; в) 900 = 1000 − 100 = CM; г) 1500 = 1000 + 500 = MD; д) 1554 = 1000 + 500 + 50 + 5 − 1 = MDLIV; е) 1917 = 1000 + 1000 − 100 + 10 + 5 + 1 + 1 = 1917

РЕШЕНИЯ К § 10

I. а) 7D = 111B; б) 17D = 10001B; в) 37D = 100101B; г) 48D = 110000B; д) 98D = 1100010B; е) 102D = 1100100B; ж) 193D = 11000001B; з) 254D = 11111101B; и) 513D = 1000000001B; к) 999D = 1111100111B.

II. а) 101B = 5D; б) 1001B = 9D; в) 1100B = 12D; г) 10111B = 23D; д) 11011B = 27D; е) 1011000B = 88D; ж) 10111011B = 187D; з) 100010011B = 275D; и) 1000000011B = 515D; к) 010101010101B = 1365D.

III. а) да; б) нет; в) да; г) да; д) да.

IV. а) $1*2^4 + 0*2^3 + 1*2^2 + 0*2^1 + 1*2^0 = 10101B = 21D$; б) $1*2^6 + 1*2^3 + 1*2^0 + 1*2^{-2} + 1*2^{-4} = 1001001.0101B = (64 + 8 + 1 + 0.25 + 0.0625)D = 73.3125D$; в) $1*2^7 + 1*2^4 + 1*2^{-3} + 1*2^{-5} = 10010000.00101B = (128 + 16 + 0.125 + 0.3125)D = 142.15625D$

V. 010; 100; 111; 1001; 1101; 10001; 100001; 110000
2 4 7 9 13 17 33 48

VI. Переводят в соответствующий код либо левую часть тождества, либо правую, если они совпадают, тогда числа эквивалентны.

VII. а) слово "ВЕК" − 11110111 11100101 11101011
В Е К

б) "2*X+1>X−1" − 00110010 00101010 01011000 00101011 00110001
2 * X + 1

00111110 01011000 00101101 00110001
> X − 1

в) HOW DO YOU DO − 01001000 01001111 01010111 00100000 01000100
H O W ПРОБЕЛ D

01001111 01000000 01011001 01001111 01010101 00100000
O ПРОБЕЛ Y O U ПРОБЕЛ

01000100 01001111
D O

г) NOSCETE IRSUM − 01001110 01001111 01010011 01000011
N O S C

01000101 01010100 01000101 00100000 01001001 01010000
E T E ПРОБЕЛ I P

01010011 01010101 01001101
S U M

VIII. а) 88; б) VIVAT; в) X*Y=0; г) THANK YOU.

РЕШЕНИЯ К § 11

I. а) 111 + 101 = 1100; 7 + 5 = 12

б) 11011 + 01110 = 101001; 27 + 14 = 41

в) 0010001 + 1011101 = 1101110; 17 + 93 = 110

г) 11111111 + 11111111 = 111111110; 255 + 255 = 510

II. а) 111 − 101 = 010; 7 − 5 = 2

б) 11011 − 01110 = 01101; 27 − 14 = 13

в) 10011010 − 01100101 = 00110101; 154 − 101 = 53

г) 10101010 − 01010101 = 1010101; 170 − 85 = 85

III. а) 11011101 + 10101110 = 110001011; 221 + 174 = 395

11011101 − 001011111 = 100101100; 221 − 95 = 300

б) 10001000 − 00011001 = 01101111; 136 − 25 = 111

1100011 + 11001000 = 11001000; 99 + 200 = 200

IV. а)

```
  111      7
   x       x
  101      5
  111
+000
 111
100011    35
```

б)

```
   11011      27
    x          x
    1110       14
   00000
 + 11011
   11011
  11011
101111010    378
```

в)

```
  100111      39
    x          x
    1001       9
   100111
+
  100111
101011111    351
```

г)

```
   10101010      170
    x            x
    1010101       85
   10101010
+  10101010
  10101010
11100001110010  14450
```

РЕШЕНИЯ К § 14, 15

II. 3. Засекаю уровень воды в мензурке.

4. Опускаю на нити вилку в мензурку до полного погружения.

III. 1. Необходимо найти полупериметр треугольника:

$$P=\frac{A+B+C}{2}$$

2. Найти площадь треугольника, используя формулу Герона:

$$S=\sqrt{P(P-A)(P-B)(P-C)}$$

IV. 1. Привязываем нить к гирьке – получаем маятник.
2. Заставляем маятник колебаться.
3. Засекаем время колебаний.
4. Считаем число колебаний за какой-то интервал времени (например, 3 мин.)
5. Зная, что $T=2\pi\sqrt{\frac{L}{G}}$ и $T=\frac{t}{N}$, где T – период колебаний, t – время колебаний; N – число колебаний, L – длина маятника, G – ускорение свободного падения (G=9,8 м/с^2); рассчитываем длину нити: $L=\frac{Gt^2}{4\pi^2N^2}$.
6. Зная длину нити, определяем длину и ширину стола.
7. Перемножая эти величины, определяем площадь стола.

V. Потому что 10A+B+35 задуманное число.
Получится: (2A+5)*5+10+B=10A+25+10+B=10A+B+35

VI. Для трех колец оптимальный алгоритм:
1. 3 → III 2. 2 → II 3. 3 → II 4. I → III 5. 3 → I
6. 2 → III 7. 3 → III

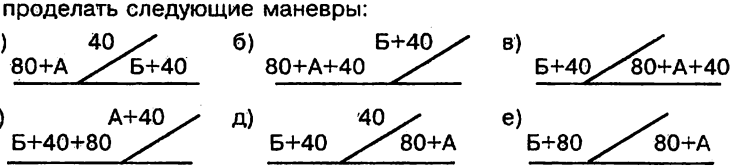
Обозначим через F(K) число перекладываний для переноса K колец с одного стержня на другой. Очевидно: F(1)=1; F(2)=3; F(3)=7; F(4)=15; F(5)=31 и т.д. В общем случае F(N)=2F(N– 1)+1. Можно заменить закономерность F(3)=2^3– 1; F(4)=2^4–1; F(5)=2^5–1

$$F(N)=2^N-1$$

VII. Задача решается в 24 хода следующими перестановками:

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

VIII. Для того чтобы разойтись, поезда с локомотивами А и Б должны проделать следующие маневры:



РЕШЕНИЯ К §16

I. 1. Ввод X, Y, Z;
2. S присвоить X,
3. Если S>Y, то идти к 5.
4. S присвоить Y.
5. Если S>Z, то идти к 7.
6. S присвоить Z.
7. Вывести значение максимального числа: S.
8. Закончить.

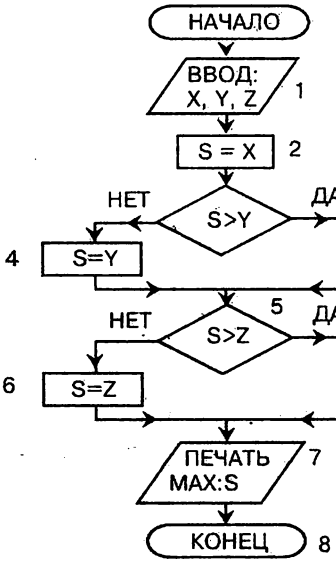


Рис. 53

II. 1.Ввод X.
2. Если X<0, то идти к 8.
3. Если X=0, то идти к 6.
4. Y присвоить 1.
5. Идти к 9.
6. Y присвоить 0.
7. Идти к 9.
8. Y присвоить – 1.
9. Вывести: Y.
10. Закончить.

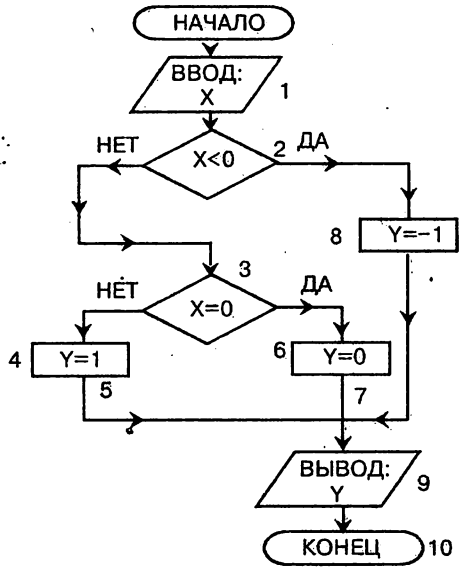


Рис. 54

III. I. Начало.
2. Ввести действительные числа A и B.
3. Если A=0, то идти к 10.
4. C присвоить значение B/A.
5. Если A>0, то идти к 8.
6. Вывести решение в виде:"X<C".
7. Идти к 14.
8. Вывести решение в виде: "X>C".
9. Идти к 14.
10 Если B<0, то идти к 13.
11. Вывести на печать "решений нет".
12. Идти к 14.
13. Вывести на печать "X- любое число".
14. Закончить.

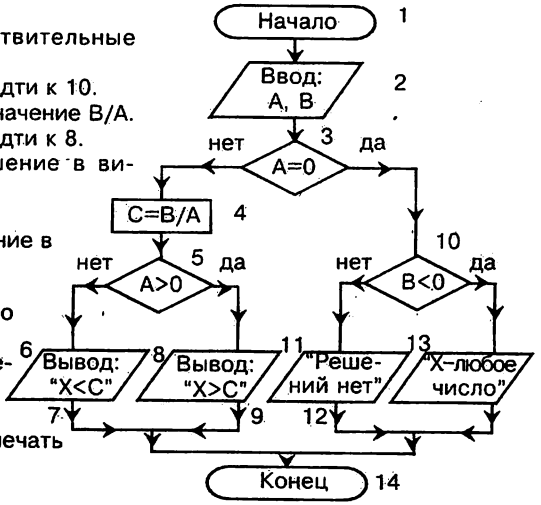


Рис. 55

IV.

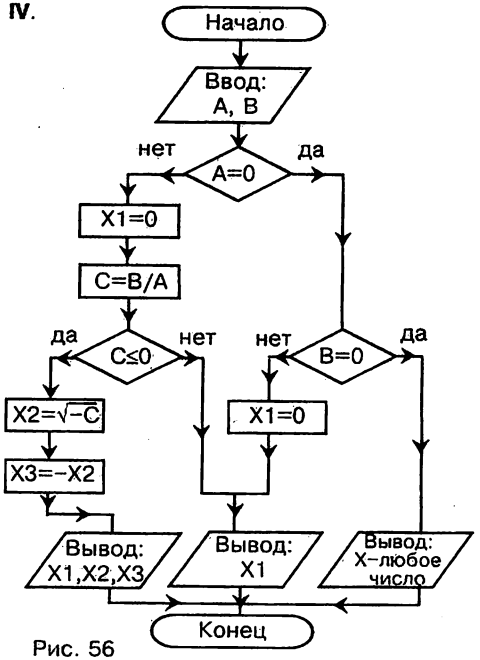


Рис. 56

V.

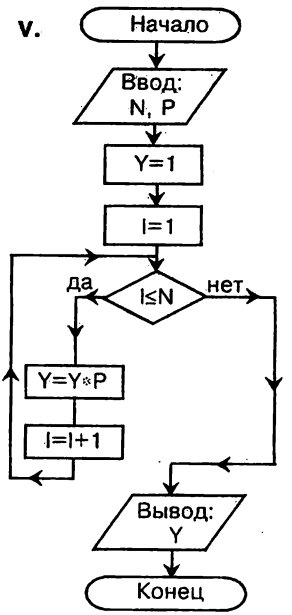


Рис. 57

РЕШЕНИЯ К § 25

I. а) – целая; б) – действительная; в) – символьная; г) – символьная; д) – действительная; е) – символьная.

II. а) вместо "," нужно ставить "."; б) нет кавычек; в) вместо буквы "O" должна быть цифра "0"; г) степень не должна превосходить 38; д) внутри константы стоят кавычки.

III. а) вторая русская буква; б) первая цифра; в) три символа; г) знак символьной переменной должен стоять после буквы "D"; д) первая русская буква; е) первые цифры.

IV. а) 0.38E–2; б) –.173056E+3; в) E–5; г) –.1274E6; д) –.489E–6.

V. В этой цепочке 4-я запись неверна, должно быть: 735.14E–4

VI. а) X^2+Y^2–Z^2; б) B+3.14*C; в) A^2+B^2–2*A*B*COS(BE); г)16/(C–D)*K; д) A0+Y*(A1+Y*(A2–A3*Y)); е) LOG(ABS(X–1))–S*(1/(2+X)+SQR(B)/TAN(X)); ж) (0.237–B*SIN(SQR(X)))^3–4^COS(Y).

VII. а) $\frac{x^2}{a/bcosx} - \frac{\sqrt{y^2-1}}{ck-\ln|x+3|} + 3bk^3\sqrt{tgx^3-2}$;

б) $5c(i,j)-3b/n+e^{x-1}\sqrt{\sin^2(x-y)-1} + \frac{1-|x^2-y^2|}{e^{\cos i}}$

в) $x^{y^z} + 0{,}017bc - arctgx + \frac{\sin^2 x - \cos^2 x}{tgx-1}$

VIII. а) X*Y/Z; б) (A+B)*C/X; в) K+Y^(1+N)*(K+N); г) X*Y/A*B/K*N.

РЕШЕНИЯ К § 26

I. а) X=(A+Z)*(A–Z); б) X=Y^2–3*K*N; в) X=X–10*SIN(X).

II. а) 3X=X^3 – в левой части использовано недопустимое имя переменной; б) K+L=Y – в левой части должно стоять имя переменной, а не выражение; в) A1:=A1/BK – в левой части равенства неправильно написано имя переменной (стоит двоеточие); г) Q=Q! – символ "!" не применяется; д) R(2.8)=2*PI*R – неправильное имя переменной; е) D¤=L¤;H¤;2.8 – в правой части недопустимый символ ";", кроме того, D¤ определяет символьное выражение, следовательно, константа 2.8 недопустима; ж) Л3=45 неправильное имя переменной (русская буква); з) I%="ИМЯ" – несоответствие типа переменных (переменная целая, а константа символьная); и) S¤=S%+1 – несоответствие типов (переменная символьная, а правая часть числовая, причем S% – целая); к) X=5Y–COSX – в правой части пропущен знак умножения и в функции косинуса аргумент должен быть в скобках.

III. а)

	C	D
исходное состояние (команда NEW) ⟶	0	0
выполнение 10 строки ⟶	2	0
20 ⟶	2	4

		30	6	4
		40	6	1
		50	7	1
б)			C	D
	исходное состояние (NEW)		0	0
	строка	10	0	5
		20	0	6
		30	1	12

IV. а) Y=TAN(Y); б) LS(5,8)=(A+B)/2*COS(X); в) A$(3)="АЭРО" +"ДИНАМИКА"

V. а) X=10; б) X$="ТЕРМОГИДРОСТАТ"

РЕШЕНИЯ К § 27

I. По строкам:

```
10 - очистит экран;
20 - ABCDE - слитное написание;
30 - A          B          C          D          E
40 - ABCDE - слитное написание;
50 - 12345
60 - 1 2 3 4 5
70 -  1          2          3          4          5
80 - A
90 - B
100 - C
110 - D
120 - E
130-170 - ABCDE
180-220 -  A          B          C          D          E
230 -  A
       B
       C
       D
       E
240 - ABCDE - слитное написание
250 -      A          B          C          D          E
```

II. Точка с запятой в списке переменных оператора PRINT для символьных и числовых данных "работает" по-разному: для символьных – она сливает воедино значения; для числовых – соединяет их, но между ними оставляет пробел (точнее, 2 пробела, с учетом знака).

III.

```
7.2                          33          22
1 зона     2 зона     3 зона     4 зона     5 зона
```

В первой зоне печатает значение переменной X, вторую и третью зону пропускает, в четвертой зоне печатает результат произведения значения переменной Y на 11, в пятой зоне печатает константу 22.

IV.
```
10 LET K$="10A"
20 LET N=25
30 LET B=4.03
40 LET P=0.87
50 PRINT"КЛАСС",K$
60 PRINT"КОЛ-ВО УЧ-СЯ",N
70 PRINT"СРЕДНИЙ БАЛЛ",B
80 PRINT"ПОСЕЩАЕМОСТЬ",P
```

V.
```
PRINT,,,A,,B,C$
```

VI. 1) ОТВЕТ: X= 8.000001

2) При A= 0 X= −2, Y= 3

−2 в квадрате равно 4,3, в кубе равно 27.

VII.
```
10 LET A=2
20 LET B=9
30 LET X=.5
40 LET Y=A^X-B*COS(X)
50 LET Z=SQR(B)+A*X
60 PRINT"ПРИ ЗНАЧЕНИИ A=";A;", B=";B", X=";X
70 PRINT"ФУНКЦИЯ Y РАВНА";Y;", ЗНАЧЕНИЕ ФУНКЦИИ Z=";Z
80 END
RUN (ВК)
```

ПРИ ЗНАЧЕНИИ A= 2, B= 9, X=5
ФУНКЦИЯ Y РАВНА−6.48403, ЗНАЧЕНИЕ ФУНКЦИИ Z=4

VIII. 1)

ТАБЕЛЬ
УЧЕТА УСПЕВАЕМОСТИ УЧАЩИХСЯ 10А КЛАССА

ЧЕТВЕРТЬ	I	II	III	IV
МОИСЕЕНКО	4	3	3	4
ХМАРА	5	4	4	5

2) ГРАФИК ДЕЖУРСТВА

ФАМИЛИЯ	ПН.	ВТ.	СР.	ЧТ.	ПТ.	СБ.
ГОЛДА	ДА				ДА	
КАМЕНСКИЙ		ДА				ДА
СЕРЕДА			ДА			
ШЕВЧЕНКО				ДА		

IX.
```
20 PRINT TAB(1);T$;TAB(10);H$;TAB(19);D$
20 PRINT SPC(1);T$;SPC(3);H$;SPC(3);D$
```

X.
```
5 CLS
10 PRINT AT(15,1);"КОМПЬЮТЕР РАД ОБЩАТЬСЯ С ВАМИ!"
20 PRINT AT(10,22);"ДЛЯ ПРОДОЛЖЕНИЯ НАЖМИТЕ ЛЮБУЮ
КЛАВИШУ..."
```

РЕШЕНИЯ К § 28

I. В строке: 10 – в десятичном числе вместо запятой необходимо поставить точку; 20 – нет закрывающихся кавычек; 30 – умножение выполнено знаком ".", а необходимо "*"; 40 – желательно заменить запятую, на точку с запятой; 50 – отсутствует список переменных.

II.
```
10 CLS
20 INPUT"КАК ВАШЕ ИМЯ";B$
30 PRINT"ПРИВЕТ ";B$;"!"
40 PRINT"Я РАД С ВАМИ ОБЩАТЬСЯ!"
50 END
```

III.
```
10 CLS
20 INPUT"ЗАМЕНИТЕ ОДНИМ СЛОВОМ: БОЛЬШОЙ ТАНЦЕВАЛЬНЫЙ
ВЕЧЕР";B$
30 PRINT"НЕ ПРИДЕТ НА ШКОЛЬНЫЙ ";B$
40 PRINT"КТО ПОЛУЧИТ НИЗКИЙ ";B$;"!"
50 INPUT"А ТЕПЕРЬ ПОДБЕРИТЕ В НУЖНОМ ПАДЕЖЕ СИНОНИМ К
СЛОВУ 'ОТБРОСЫ'";S$
60 PRINT"ЕСЛИ В ДОМЕ МНОГО";S$;"А"
70 PRINT "В ДОМЕ МОЖЕТ ВСПЫХНУТЬ С";S$;"А"
80 PRINT"ВЕРНО! СЛОВА 'БАЛЛ' И 'ССОРА' - ПИШУТСЯ С
УДВОЕННЫМИ СОГЛАСНЫМИ!"
90 END
```

IV. Например:
```
6 INPUT"ВАША ФАМИЛИЯ";F$
8 PRINT "АВТОР ЭТОГО СТИХОТВОРЕНИЯ: ДЖОРДЖ ГОРДОН ";F$
```

РЕШЕНИЯ К § 29

I.
```
10 PRINT "И ДЕНЬ ИДЕТ, И НОЧЬ ИДЕТ."
20 GOTO 10
```

II.
```
5'          АННА АХМАТОВА
10 PRINT"И ВОВСЕ Я НЕ ПРОРОЧИЦА,"
20 GOTO 50
30 PRINT"А ПРОСТО МНЕ ПЕТЬ НЕ ХОЧЕТСЯ"
40 GOTO 70
50 PRINT"ЖИЗНЬ МОЯ СВЕТЛА, КАК РУЧЕЙ,"
60 GOTO 30
70 PRINT"ПОД ЗВОН ТЮРЕМНЫХ КЛЮЧЕЙ."
80 END

RUN (ВК)
И ВОВСЕ Я НЕ ПРОРОЧИЦА,
ЖИЗНЬ МОЯ СВЕТЛА, КАК РУЧЕЙ,
А ПРОСТО МНЕ ПЕТЬ НЕ ХОЧЕТСЯ
ПОД ЗВОН ТЮРЕМНЫХ КЛЮЧЕЙ.
```

III. а) 3; б) 9

IV.
```
RUN (ВК)
Я
И
ГРАФ
БИО
```

АВТО
ГРАФИЯ
БИОГРАФИЯ
АВТОБИОГРАФИЯ

РЕШЕНИЯ К § 30

I. Строка 20 в этих программах записана по-разному: K=8 и 8=K – это одно и то же, ведь в условии оператора IF...THEN знак "=" имеет смысл "равно", а не "присвоить".

После выполнения первой программы в строке 20, в случае, если условие K=8 не выполняется, компьютер перейдет на выполнение следующей за этим оператором строки, т.е. 30-й. На экране компьютер напишет: "НЕПРАВИЛЬНО !".

Во второй программе строка 20 будет выполнена точно так же. В случае невыполнения условия (8=K) компьютер перескочит на следующую строку после IF...THEN и будет ее выполнять. А 30-я строка – новое условие, причем заведомо выполнимое (т.е. действительно K не равно 8), следовательно, на экране компьютер тоже напишет: "НЕПРАВИЛЬНО !", т.е. обе эти программы будут работать совершенно одинаково.

II.
```
10 INPUT"КАКОВ ЦВЕТ ЛАКМУСОВОЙ БУМАЖКИ";C$
20 IF C$="КРАСНЫЙ" THEN 60
30 IF C$="СИНИЙ" THEN 80
35 IF C$="БЕСЦВЕТНЫЙ" THEN 40
37 IF C$<>"КРАСНЫЙ" AND C$<>"СИНИЙ" AND C$<>"БЕСЦВЕТНЫЙ"
THEN PRINT "НЕПРАВИЛЬНЫЙ ВВОД"
38 GOTO 10
40 PRINT"РАСТВОР НЕЙТРАЛЬНЫЙ"
50 GOTO 90
60 PRINT"РАСТВОР КИСЛОТНЫЙ"
70 GOTO 90
80 PRINT"РАСТВОР ЩЕЛОЧЕЙ"
90 STOP
```

III.
```
10 INPUT"ВВЕДИТЕ ДВА ЧИСЛА A,B";A,B
20 IF A>B THEN X=A ELSE X=B
30 PRINT"МАКСИМАЛЬНОЕ ЧИСЛО";X
40 END
```

```
10 INPUT"ВВЕДИТЕ ДВА ЧИСЛА";A,B
20 IF A>B THEN 50
30 X=B
40 GOTO 60
50 X=A
60 PRINT"МАКСИМАЛЬНОЕ ЧИСЛО";X
70 END
```

IV. `10 IF X>=0 THEN IF X<=10 THEN PRINT"ТОЧКА ПРИНАДЛЕЖИТ ЭТОМУ ИНТЕРВАЛУ"`

V. Одна из программ проверки, делится ли одно из этих чисел на другое нацело:

```
10 INPUT"ВВЕДИТЕ ДВА ЧИСЛА"; X,Y
20 IF X*Y=0 THEN 10
30 N%=X/Y
40 IF N%*Y=X THEN PRINT X;" ДЕЛИТСЯ НАЦЕЛО НА";Y ELSE 50
45 GOTO 80
50 N%=Y/X
60 IF N%*X=Y THEN PRINT Y;"ДЕЛИТСЯ НАЦЕЛО НА";X ELSE 70
65 GOTO 80
70 PRINT "ЧИСЛА НЕ ДЕЛЯТСЯ ДРУГ НА ДРУГА НАЦЕЛО"
80 END
```

Эта же программа несколько в другом виде:

```
10 INPUT"ВВЕДИТЕ ДВА ЧИСЛА M,N";M,N
20 IF X*Y=0 THEN 10
30 N%=X/Y
40 IF N%*Y=X THEN PRINT X;"ДЕЛИТСЯ НАЦЕЛО НА";Y
50 IF N%*Y<>X THEN PRINT"ЧИСЛА НЕ ДЕЛЯТСЯ ДРУГ НА ДРУГА
НАЦЕЛО"
60 END
```

VI.

```
5 CLS
10 INPUT"ЗАДУМАЙТЕ ЦЕЛОЕ
ЧИСЛО НЕ БОЛЬШЕ СТА";N
50 CLS
60 PRINT"УГАДАЙТЕ ЧИСЛО ОТ
1 ДО 100"
70 INPUT "ЧИСЛО ="P
110 IF P=N THEN GOTO 155
130 GOTO 70
155 PRINT"ВЫ УГАДАЛИ!"
```

Добавим проверку на ввод числа не больше 100

```
20 IF N>100 THEN 10
```

А теперь добавим строки в программу, которые подсказывали бы отгадчику (больше или меньше задуманного числа он сделал ответ).

```
90 IF P>N THEN PRINT"МНОГО"
100IF P<N THEN PRINT"МАЛО"
```

Заведем счетчик сделанных попыток. Для этого сначала обнулим переменную S:

```
40 S=0 ,
```

а затем поставим сам счетчик:

```
80 S=S+1 ,
```

теперь зададим нужное число попыток (например, 5):

```
120 IF S=5 THEN 135
130 GOTO 70.
135 CLS
140 PRINT"КОЛИЧЕСТВО ПОПЫТОК ИСЧЕРПАНО! ВЫ НЕ УГАДАЛИ!"
150 PRINT"ЗАДУМАННОЕ ЧИСЛО = ";N
153 IF S=5 THEN 160
```

А теперь предоставим возможность после прогона программы

выбрать: будет ли продолжена игра или закончена:

```
160 PRINT"ХОТИТЕ ЛИ ЕЩЕ
СЫГРАТЬ (ОТВЕТЬТЕ 'ДА' ИЛИ 'НЕТ')";
170 INPUT H$
30 D$="ДА": N$="НЕТ"
180 IF H$=D$ THEN 5
190 IF H$=N$ THEN 210
200 IF H$<>D$ THEN IF H$
<>N$ THEN 160
210 END
```

Окончательно программа будет выглядеть так:

```
5 CLS
10 INPUT"ЗАДУМАЙТЕ ЦЕЛОЕ
ЧИСЛО НЕ БОЛЬШЕ СТА";N
20 IF N>100 THEN 10
30 D$="ДА": N$="НЕТ"
40 S=0
50 CLS
60 PRINT"УГАДАЙТЕ ЧИСЛО
ОТ 1 ДО 100"
70 INPUT"ЧИСЛО=";P
80 S=S+1
90 IF P>N THEN PRINT"МНОГО"
100 IF P<N THEN PRINT"МАЛО"
110 IF P=N THEN GOTO 155
120 IF S=5 THEN 135
130 GOTO 70
135 CLS
140 PRINT"КОЛИЧЕСТВО ПОПЫТОК
ИСЧЕРПАНО! ВЫ НЕ УГАДАЛИ!"
150 PRINT"ЗАДУМАННОЕ ЧИСЛО =";N
153 IF S=5 THEN 160
155 PRINT"ВЫ УГАДАЛИ!"
160 PRINT"ХОТЕЛИ ЕЩЕ СЫГРАТЬ
(ОТВЕТЬТЕ 'ДА' ИЛИ 'НЕТ')"
170 INPUT H$
180 IF H$=D$ THEN 5
190 IF H$=N$ THEN 210
200 IF H$<>D$ THEN IF H$<>N$
THEN 160
210 END
```

VII. Тренировка 7-1

```
10 REM ПОИСК МАКСИМАЛЬНОГО
ЧИСЛА ИЗ ТРЕХ ЗАДАННЫХ
20 INPUT"ВВЕДИТЕ ТРИ ЧИСЛА:
X,Y,Z";X,Y,Z
30 S=X
40 IF S>Y THEN 60
50 S=Y
60 IF S>Z THEN 80
70 S=Z
80 PRINT"МАКСИМАЛЬНЫМ ИЗ
ЭТИХ ЧИСЕЛ ЯВЛЯЕТСЯ ЧИСЛО:";S
90 END
```

Тренировка 7-2

```
10 REM ФУНКЦИЯ СИГНИУМ
(SIGN X)
20 INPUT"ВВЕДИТЕ ЧИСЛО X";X
30 IF X<0 THEN 80
40 IF X=0 THEN 60
50 Y=1
55 GOTO 90
60 Y=0
70 GOTO 90
80 Y=-1
90 PRINT"ФУНКЦИЯ Y = ";Y
100 END
```

Тренировка 7-3

```
10 REM РЕШЕНИЕ НЕРАВЕНСТВА
ВИДА AX >B
20 INPUT"ВВЕДИТЕ
КОЭФФИЦИЕНТЫ A И B:";A,B
30 IF A=0 THEN 100
40 C=B/A
50 IF A>0 THEN 80
60 PRINT"НЕИЗВЕСТНОЕ B
НЕРАВЕНСТВЕ: X<";C
70 GOTO 120
80 PRINT"НЕИЗВЕСТНОЕ
B НЕРАВЕНСТВЕ: X>";C
90 GOTO 120
100 IF B<0 THEN PRINT "X -
ЛЮБОЕ ЧИСЛО" ELSE PRINT"
НЕРАВЕНСТВО РЕШЕНИЙ
НЕ ИМЕЕТ"
120 END
```

Тренировка 7-4

```
10 REM РЕШЕНИЕ УРАВНЕНИЯ
ВИДА AX^3+BX=0
20 INPUT"ВВЕДИТЕ
КОЭФФИЦИЕНТЫ A И B";A,B
40 X1=0
50 C=B/A
60 IF C<=0 THEN 62 ELSE
GOTO 120
62 X2=SQR(-C)
64 X3=-X2
80 PRINT"УРАВНЕНИЕ ИМЕЕТ ТРИ
КОРНЯ:X1=";X1;"X2=";X2;"X3=";X3
90 GOTO 130
100 IF B=0 THEN 110 ELSE X1=0
105 GOTO 120
110 PRINT "НЕИЗВЕСТНОЕ X -
ЛЮБОЕ ЧИСЛО"
120 PRINT"УРАВНЕНИЕ ИМЕЕТ
ОДИН КОРЕНЬ X1=";X1
130 END
```

Тренировка 7-5

```
10 REM РЕШЕНИЕ УРАВНЕНИЯ
ВИДА Y=P^N, ГДЕ P-ЛЮБОЕ
ДЕЙСТВИТЕЛЬНОЕ ЧИСЛО,
N-НАТУРАЛЬНОЕ ЧИСЛО
20 INPUT"ВВЕДИТЕ N,P";N,P
30 Y=1
40 I=1
50 IF I<N THEN 60 ELSE 90
60 Y=Y*P
70 I=I+1
80 GOTO 50
90 PRINT"ЗНАЧЕНИЕ Y=";Y
100 END
```

РЕШЕНИЯ К § 31

I.
```
10 FOR X=-5 TO 5 STEP 2
20 Y=5/Y
30 PRINT X,Y
40 NEXT X
50 END
```

II. а) STEP 4; б) STEP 0.5;

III.
```
10 CLS
20 K$="КЛАРА У КАРЛА
УКРАЛА КОРАЛЛЫ, А КАРЛ
У КЛАРЫ УКРАЛ КЛАРНЕТ"
25 PRINT K$
30 FOR I=1 TO 2000
40 NEXT I
50 CLS
60 INPUT "НАПИШИТЕ
УВИДЕННУЮ СЕЙЧАС
СКОРОГОВОРКУ (ВНИМАТЕЛЬНО
СЛЕДИТЕ ЗА ПРОБЕЛАМИ И
ЗНАКАМИ ПРЕПИНАНИЯ)";S$
70 IF K$=S$ THEN PRINT
"ПРАВИЛЬНО"
80 IF K$<>S$ THEN PRINT
"ЧТО-ТО НЕ ТО, ПОПРОБУЙТЕ ЕЩЕ"
82 FOR I=1 TO 1000
84 NEXT I
90 GOTO 60
```

IV.
```
1.5
1
.5
0
```

V.
```
10 LET S=0
30 FOR I=1 TO 10
50 LET S=S+I
60 NEXT I
70 PRINT S
80 END
```

VI. Цикл может быть таким:

```
10 S=1:A=1
15 D=-3
```

```
20 FOR I=1 TO 11                40 NEXT I
30 A=A+D:S=S+A                  50 PRINT S
                                60 END
```

VII.

а)
```
10 U1=2:Q=3:S=0                 30 PRINT S
20 S=U1*(1-Q^10)/(1-Q)          40 END
```

б)
```
10 S=2:U=2:Q=3                  40 NEXT I
20 FOR I=1 TO 9                 50 PRINT S
30 U=U*Q:S=S+U                  60 END
```

VIII.
```
10' REM ВЫЧИСЛЕНИЕ N!           50 NN=NN*K
20 INPUT "ВВЕДИТЕ ЦЕЛОЕ ЧИСЛО";N 60 NEXT K
30 NN=1                         70 PRINT N;"! РАВНО:";NN
40 FOR K=1 TO N                 80 END
```

IX.
```
10 ' СБЕРКНИЖКА
20 INPUT"СУММА ВКЛАДА (РУБ.)";S
30 INPUT"КОЛИЧЕСТВО ЛЕТ ХРАНЕНИЯ ЭТОЙ СУММЫ";L
40 INPUT"ПОД КАКИМИ ПРОЦЕНТАМИ";P
50 FOR I=1 TO L
60 S=S+S*P/100
70 NEXT I
80 PRINT"ЧЕРЕЗ";L;"ЛЕТ СУММА ВКЛАДА БУДЕТ";S;"РУБ."
90 END
```

РЕШЕНИЯ К § 32

I.

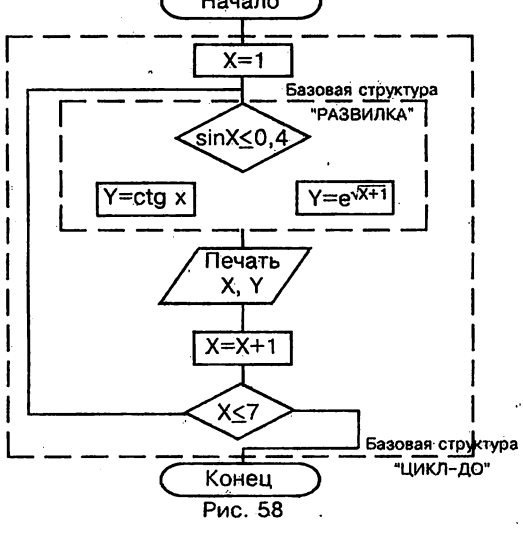


Рис. 58

II. Решения заданий из тренировки к § 16:

Номер 1. Следующие друг за другом базовые структуры ЦИКЛ--ПОКА.

Номер 2.

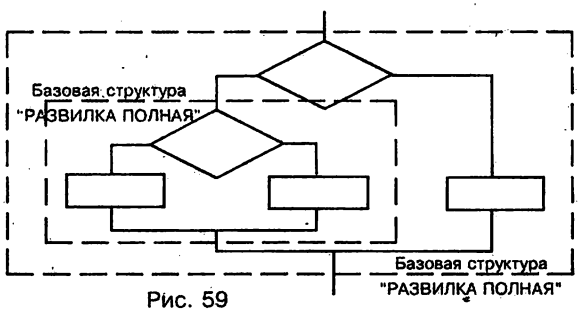


Рис. 59

Номер 3.

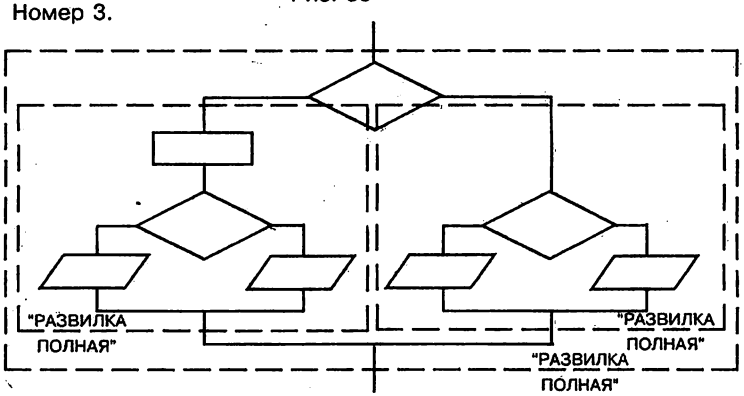


Рис. 60

III

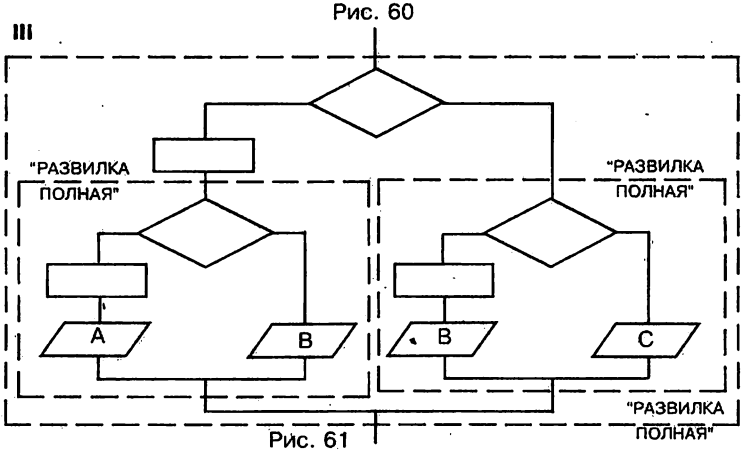


Рис. 61

РЕШЕНИЕ К § 33

Программа может быть такой:

```
10 FOR K=1 TO 10
20 CLS
30 FOR L=1 TO 1000   Пауза
40 NEXT L
50 PRINT TAB(10);"ОПАСНОСТЬ!"
60 FOR L=1 TO 1000   Пауза
70 NEXT L
80 NEXT K
```

РЕШЕНИЯ К § 34

I. 1) Найдем определитель системы Δ: $\Delta=a_{11}\cdot a_{22}-a_{21}\cdot a_{12}$

2) Найдем определитель системы Δx и Δy: $\Delta x= a_{13}\cdot a_{22}-a_{23}\cdot a_{12}$; $\Delta y = a_{11}\cdot a_{23}-a_{21}\cdot a_{13}$

3) Если Δ≠0, то найдем неизвестные:

$x=\Delta x/\Delta$; $y=\Delta y/\Delta$

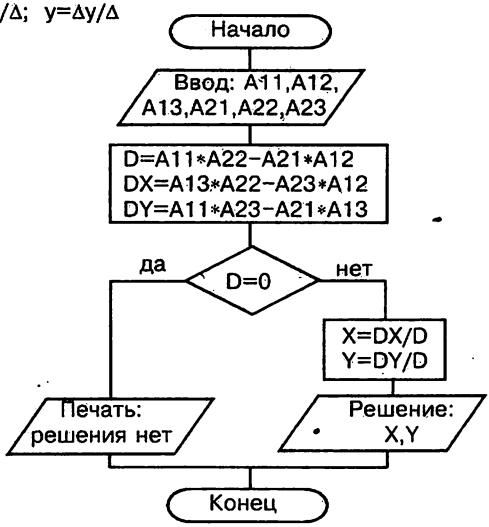


Рис. 62

II.

```
10 PRINT"РЕШЕНИЕ ЛИНЕЙНЫХ
   СИСТЕМ С ПОМОЩЬЮ
   ОПРЕДЕЛИТЕЛЕЙ"
20 PRINT"УРАВНЕНИЕ ВИДА:"
30 PRINT"A11X+A12Y=A13"
40 PRINT"A21X+A22Y=A23"
50 INPUT"ВВЕДИТЕ
   КОЭФФИЦИЕНТЫ УРАВНЕНИЙ
   В ТАКОЙ ПОСЛЕДОВАТЕЛЬНОСТИ:
   A11,A12,A13,A21,A22,A23";
   A1,A2,A3,B1,B2,B3
60 LET D=A1*B2-B1*A2
70 LET DX=A3*B2-B3*A2
80 LET DY=A1*B3-B1*A3
90 IF D=0 THEN 100 ELSE 120
100 PRINT"РЕШЕНИЯ НЕТ"
110 GOTO 150
120 LET X=DX/D
130 LET Y=DY/D
140 PRINT"РЕШЕНИЕ СИСТЕМЫ:
   X=";X;" Y=";Y
150 END
```

III. Для вычисления суммы элементов массива может быть применена обычная схема: сумма=сумма+слагаемое (S=S+Xi), где i меняется от 1 до N. Если учесть, что вначале S=0 и что среднее арифметическое равно конечной сумме, деленной на число слагаемых N, то алгоритм будет таким:

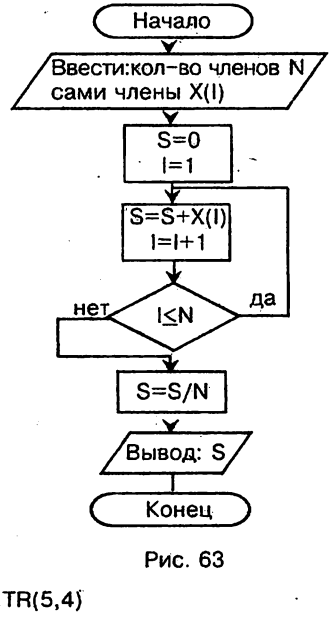


Рис. 63

IV. 5 DIM N¤(7),M¤(12),D%(12),TR(5,4)

РЕШЕНИЯ К § 35

I. а) В строке 10 неверно указана размерность, должно быть B¤(4, 4); перепутаны операторы NEXT местами в строках 50 и 60; нет блока данных, т.е. оператора DATA; б) В операторе DATA нехватка данных: там лишь три числа; в) в операторе DIM описан трехмерный массив, видимо, должно быть DIM C(3,9); в строке 60 оператор DATA имеет несоответствие с массивом C% в операторе READ (строка 40; значения "школа", "дверь", "stop"; 3.2 необходимо заменить на целые числа.

II. а) X=1.5; Y=3; Z=−9
K=1.5; L=3

б) A=17; B=19
C=11; D=13; E=15

III. Программа может быть такой:

```
10 READ X
20 IF X<0 THEN 70
30 READ Y
40 IF Y<0 THEN 70
50 Q=SQR(X*Y)
60 PRINT Q
65 GOTO 75
70 PRINT"ЗАДАЧУ РЕШАТЬ
НЕ БУДЕМ"
75 END
80 DATA 3,4
```

числа можно задавать любые

IV. 10' ПЛАТА ЗА ЭЛЕКТРОЭНЕРГИЮ
20 LET P=3

```
30 DIM N¤(P),M(P),U(P)
40 FOR K=1 TO P
50 READ N¤(K),M(K)
60 PRINT"ВРЕМЯ РАБОТЫ В ДЕНЬ (В ЧАС): ПРИБОР-";N¤(K);
70 INPUT U(K)
80 NEXT K
82 INPUT "КОЛИЧЕСТВО ДНЕЙ ПОТРЕБЛЕНИЯ";D
85 INPUT "ВВЕДИТЕ СТОИМОСТЬ 1 КВТ.ЧАС";T
87 PRINT "_____"
90 PRINT "ПРИБОР     МОЩНОСТЬ    ВРЕМЯ     ПОТРЕБЛ.  СУММА"
                                 РАБОТЫ    ЭНЕРГИЯ"
110 PRINT "_____"
120 LET SE=0
130 LET R=0
140 FOR K=1 TO P
150 LET E=D*M(K)*U(K)
160 LET S=E*T
170 PRINT N¤(K);TAB(6);M(K);TAB(20);U(K);TAB(27);E;TAB(36);S
180 SE=SE+E
190 R=R+S
200 NEXT K
205 PRINT
210 PRINT"ОБЩАЯ ПОТРЕБЛЯЕМАЯ ЭНЕРГИЯ-";SE;", ОБЩАЯ СУММА-";R
220 END
230 DATA ЭЛЕКТРОЛАМПОЧКИ,0.8, ТЕЛЕВИЗОР,0.12,ХОЛОДИЛЬНИК,0.14
```

Переменные: N¤ – название электроприборов; M – их мощность; V – время работы; T – стоимость 1 квт.ч электроэнергии; E – потребляемая электроэнергия; SE – общая потребляемая электроэнергия; R – общая сумма.

РЕШЕНИЯ К § 37

I. Программа может быть такой:

```
10 GOSUB 95
30 LET Y1=SQR(X)
40 GOSUB 95
60 LET Y2=SIN(X)
70 GOSUB 95
90 LET Y3=COS(X)
92 GOTO 100
95 INPUT "ВВЕДИТЕ
АРГУМЕНТ ФУНКЦИИ";X
97 IFX<=0 THEN PRINT
"ПОВТОРИТЕ ВВОД" ELSE 99
99 RETURN
100 PRINT Y1,Y2,Y3
110 END
```

II. а) После того как подпрограмма будет вызвана 5 раз из тела цикла, управление будет передано на строку 40, а затем на саму подпрограмму (строка 50). Таким образом подпрограмма начнет выполняться без помощи оператора GOSUB. Когда выполнение дойдет до оператора RETURN (строка 70), то интерпретатор обнаружит, что в стеке ничего нет, и программа закончится ошибкой.

Необходимо между строками 40 и 50 поставить оператор STOP или END.

б) Оператор GOSUB нельзя применять для входа внутрь цикла. Необходимо в операторе GOSUB (строка 20) указать другой номер метки (например, 50 или 60).

III. При первом проходе компьютер доходит до строки 40, вызывает подпрограмму, начиная со строки 100. В самой же подпрограмме вызывается основная программа, а та в свою очередь снова вызывает ту же подпрограмму, а подпрограмма снова основную программу (вот это и есть рекурсия). Но выполняться такая программа будет лишь три раза.

При обращении первый раз к подпрограмме идет наращивание переменной счетчика (строка 120), а по вызову снова основной программы будет идти сравнение этой величины счетчика. И как только оно достигнет величины, равной 3, пойдет работа с возвратами (оператор RETURN в строке 60).

Первый раз он возвращает управление в программу на строку, следующую после оператора GOSUB 100 (т.е. на саму себя, т.е. снова на строку 60). А вторично этот оператор RETURN (в строке 60) отсылает на строку 140, где снова стоит оператор RETURN, который в свою очередь отсылает на такой же оператор и т.д., пока не заполнится стек 3 раза.

Затем произойдет выход на оператор END (в строке 70).

IV.

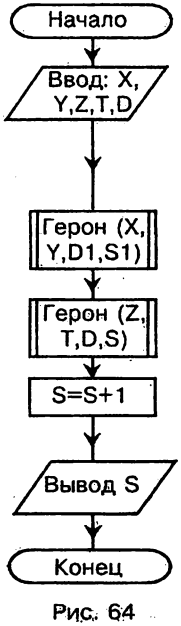


Рис. 64

```
10 REM ПЛОЩАДЬ ЧЕТЫРЕХУГОЛЬНИКА
20 INPUT "ВВЕДИТЕ СТОРОНЫ
ЧЕТЫРЕХУГОЛЬНИКА AB, BC, CD, AD";X,Y,Z,T
25 INPUT "ВВЕДИТЕ ДИАГОНАЛЬ
ЧЕТЫРЕХУГОЛЬНИКА";D
30 LET A=X
40 LET B=Y
50 LET C=D
60 GOSUB 200
70 LET S1=S
80 LET A=Z
90 LET B=T
100 GOSUB 200
110 LET S=S+S1
120 PRINT "ПЛОЩАДЬ ЧЕТЫРЕХУГОЛЬНИКА
S=";S
130 END
200 LET P=(A+B+C)/2
210 LET S=SQR(P*(P-A)*(P-B)*(P-C))
220 RETURN
```

V.

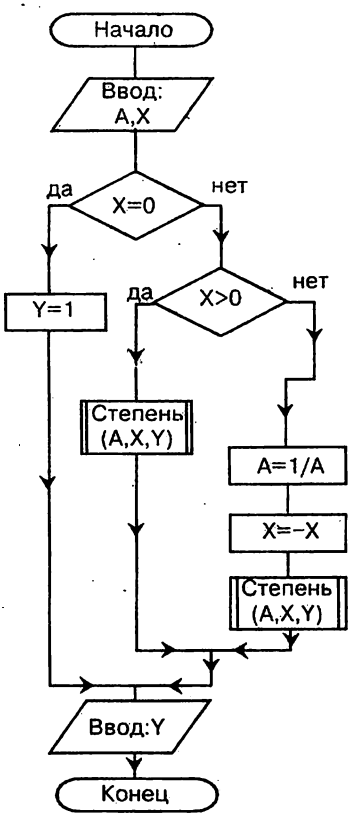
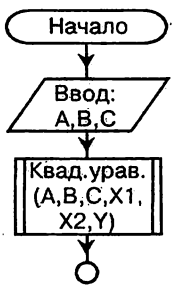


Рис. 65

```
10 REM РЕШЕНИЕ УРАВНЕНИЯ
Y=A^X, ГДЕ X - ЦЕЛОЕ ЧИСЛО
20 INPUT"ВВЕДИТЕ ОСНОВАНИЕ И
ПОКАЗАТЕЛЬ СТЕПЕНИ: A,X";A,X
30 IF X<>0 THEN 60
40 LET Y=1
50 GOTO 140
60 IF X<0 THEN 110
70 LET P=A
80 LET N=X
90 GOSUB 230
100 GOTO 140
110 LET P=1/A
120 LET N=-X
130 GOSUB 230
140 PRINT A;"В СТЕПЕНИ";X;"РАВНО";Y
150 END

230 REM ПОДПРОГРАММА РЕШЕНИЯ
УРАВНЕНИЯ ВИДА Y=P^N, ГДЕ N -
НАТУРАЛЬНОЕ ЧИСЛО
240 LET Y=1
250 LET I=1
260 IF I>N THEN 300
270 LET Y=Y*P
280 LET I=I+1
290 GOTO 260
300 RETURN
```

VI.




```
10 REM РЕШЕНИЕ БИКВАДРАТНОГО
УРАВНЕНИЯ
20 INPUT"ВВЕДИТЕ
КОЭФФИЦИЕНТЫ A,B,C";A,B,C
30 GOSUB 330
40 IF Y<>2 THEN 220
50 IF A<0 THEN 90
60 LET P1=X1
70 LET P2=X2
80 GOTO 110
```

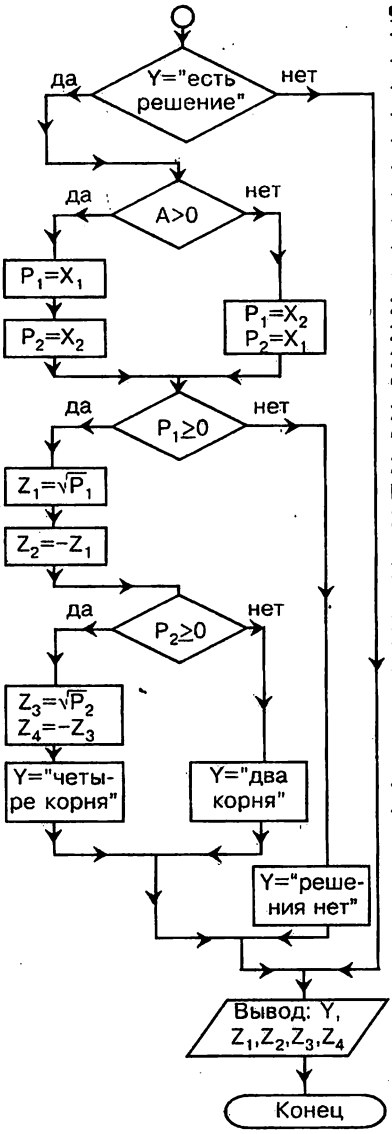


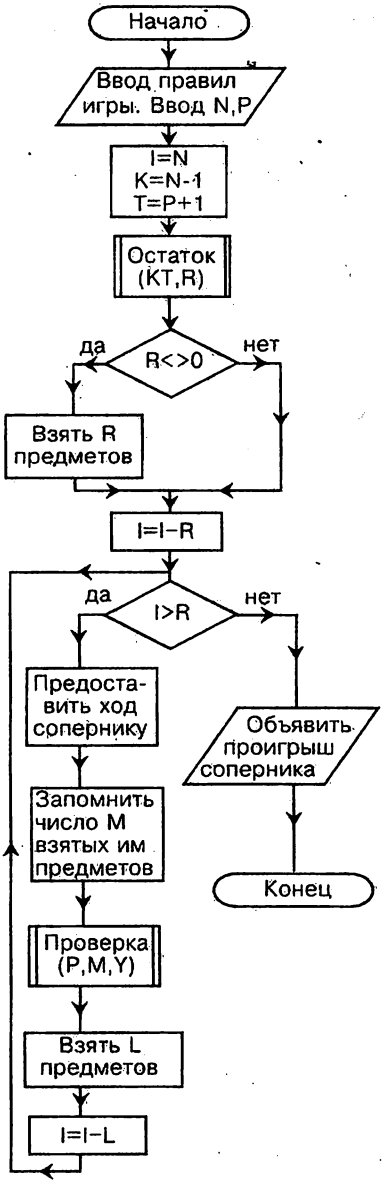
Рис. 66

```
90 LET P1=X2
100 LET P2=X1
110 IF P1<0 THEN 210
120 LET Z1=SQR(P1)
130 LET Z2=-Z1
140 IF P2<0 THEN 190
150 LET Z3=SQR(P2)
160 LET Z4=-Z3
170 LET Y=4
180 GOTO 200
190 LET Y=2
200 GOTO 220
210 LET Y=0
220 IF Y=4 THEN PRINT
"4 КОРНЯ:";Z1,Z2,Z3,Z4
230 IF Y=2 THEN PRINT
"2 КОРНЯ:";Z1,Z2
240 IF Y=0 THEN PRINT"НЕТ
РЕШЕНИЯ"
250 END

330 REM ПОДПРОГРАММА РЕШЕНИЯ
КВАДРАТНОГО УРАВНЕНИЯ
340 LET D=B^2-4*A*C
350 IF D<0 THEN 400
360 LET X1=(-B+SQR(D))/(2*A)
370 LET X2=-(X1+B/A)
380 LET Y=2
390 GOTO 410
400 LET Y=0
410 RETURN
```

VII. Разделим с остатком число (N−1) на число (P+1); R – остаток от деления. На первом шаге возьмем R предметов (если R=0) или передадим ход сопернику (если R=0).

Каждый своим ходом будет дополнять количество взятых соперником предметов до P+1 до тех пор, пока не останется один предмет. При составлении программы предусмотреть контроль правильности хода соперника. Остаток вычислим посредством подпрограммы (см. § 30).




```
10 REM ИГРА БАШЕ
20 PRINT"ИМЕЕТСЯ N ПРЕДМЕТОВ.
ИГРОКИ ХОДЯТ ПО ОЧЕРЕДИ.";
22 PRINT"ЗА ОДИН ХОД МОЖНО
ВЗЯТЬ ОТ 1 ДО P ПРЕДМЕТОВ"
24 PRINT"ПРОИГРЫВАЕТ ТОТ, КТО
ВЗЯЛ ПОСЛЕДНИЙ ПРЕДМЕТ":PRINT
30 PRINT"НАЧИНАЕМ ИГРУ"
40 INPUT"ВВЕДИТЕ ОБЩЕЕ
КОЛИЧЕСТВО ПРЕДМЕТОВ (N) И
ПРЕДЕЛЬНОЕ ЧИСЛО БЕРУЩИХСЯ
ПРЕДМЕТОВ (P)"; N,P
50 LET I=N
60 LET K=N-1
70 LET L=P+1
80 GOSUB 300
90 IF R=0 THEN 110
100 PRINT"МОЙ ХОД";R
110 LET I=I-R
120 IF K<=L THEN 190
130 PRINT"ПРЕДМЕТОВ";I
140 INPUT "ВАШ ХОД";M
150 GOSUB 210
160 PRINT"МОЙ ХОД";L-M
170 LET I=I-L
180 GOTO 120
190 PRINT"ОСТАЛСЯ ОДИН
ПРЕДМЕТ,ВЫ ПРОИГРАЛИ"
200 END

210 REM ПРОВЕРКА
220 IF M<1 THEN 240
230 IF M<=P THEN 260
240 PRINT"ВЫ НАРУШИЛИ ПРАВИЛА
ИГРЫ. ИГРА ПРЕКРАЩЕНА."
250 STOP
260 RETURN
```

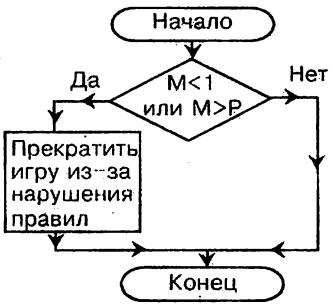


Рис. 67

```
300 REM ПРОГРАММА ДЕЛЕНИЯ С
ОСТАТКОМ
310 LET R=K
320 IF R<L THEN 350
330 LET R=R-L
340 GOTO 320
350 RETURN
```

К § 38

I. При A=1 программа прерывается, т.к. X=−3; при A=2 X=2; при A=3 X=13; при A=4 X=20.

III. Программа может быть такой:

```
10 REM ВЫЧИСЛЕНИЕ ГИПОТЕНУЗЫ
20 DEF FN G(A,B)=SQR(A*A+B*B)
30 INPUT"ВВЕДИТЕ ЗНАЧЕНИЕ КАТЕТОВ A,B";A,B
40 C=FN G(A,B)
50 PRINT"ГИПОТЕНУЗА РАВНА";C
```

IV. Программа может быть такой:

```
10 PRINT"КАКОЕ ТЕМПЕРАТУРНОЕ ПРЕОБРАЗОВАНИЕ ВАС
ИНТЕРЕСУЕТ:"
20 PRINT"1. ИЗ ШКАЛЫ ФАРЕНГЕЙТА В ЦЕЛЬСИЯ"
30 PRINT"2. ИЗ ШКАЛЫ КЕЛЬВИНА В ЦЕЛЬСИЯ"
35 PRINT"3. ИЗ ШКАЛЫ ЦЕЛЬСИЯ В ФАРЕНГЕЙТА"
40 PRINT"4. ИЗ ШКАЛЫ ЦЕЛЬСИЯ В КЕЛЬВИНА"
60 PRINT"5. ИЗ ШКАЛЫ КЕЛЬВИНА В ФАРЕНГЕЙТА"
70 PRINT"6. ИЗ ШКАЛЫ ФАРЕНГЕЙТА В КЕЛЬВИНА"
80 INPUT"НАБЕРИТЕ НОМЕР ИНТЕРЕСУЮЩЕГО ПЕРЕВОДА";N
90 ON N GOSUB 130,140,150,160,170,180
100 INPUT"НАБЕРИТЕ ТЕМПЕРАТУРУ В НАЧАЛЬНОЙ ШКАЛЕ";T
110 PRINT"ТЕМПЕРАТУРА В ДРУГОЙ ШКАЛЕ БУДЕТ";
FN TE(T);"ГРАДУСОВ"
120 END

130 DEF FN TE(T)=(T-32)*5/9
135 RETURN

140 DEF FN TE(T)=T-273
145 RETURN

150 DEF FN TE(T)=9/5*T+32
155 RETURN

160 DEF FN TE(T)=T+273
165 RETURN
```

```
170 DEF FN TE(T)=9/5*(T-273)+32
175 RETURN

180 DEF FN TE(T)=273+(T-32)*5/9
185 RETURN
```

V.
```
10' ПРИНАДЛЕЖНОСТЬ КРУГУ
20 DEF FN KR(X,Y)=(X-X0)^2+(Y-Y0)^2-R^2
30 INPUT "ЦЕНТР КРУГА X0,Y0";X0,Y0
40 INPUT"РАДИУС R";R
50 INPUT"ТОЧКА X,Y";X,Y
60 IF FN KR(X,Y)<=0 THEN PRINT"ТОЧКА ПРИНАДЛЕЖИТ КРУГУ"
ELSE PRINT"ТОЧКА НЕ ПРИНАДЛЕЖИТ КРУГУ"
70 END
```

К § 39

I. Эта программа аналогична программе нахождения самого длинного слова за исключением строк 10 и 50 (см. с.257). Строка 10 содержит достаточно длинную символьную строку, с которой компьютер сравнивает все вводимые слова, неважно, какие символы помещены в переменную A$

```
10 LET A$="...................................................."
30 IF LEN(B$)<LEN(A$) THEN LET A$=B$
70 PRINT "САМОЕ КОРОТКОЕ СЛОВО:"
```

II.
```
5 A$="КОРЗИНА"
10 MID$(A$,2,3)="АРТИСТ"
20 PRINT A$
```

III.
```
10 INPUT "ВВЕДИТЕ СЛОВО
   ИЛИ ФРАЗУ:";S$
20 LET L=LEN(S$)
30 FOR N=L TO 1 STEP-1
40 PRINT MID$(S$,N,1);
50 NEXT N
60 STOP
```

Испытайте на таких словах: ТУТ, ПОП, ТОПОТ, КАЗАК, КОМОК, ЗАКАЗ, ШАЛАШ или другой вариант:

```
10 INPUT"ВВЕДИТЕ СТРОКУ";S$
20 LET N=LEN(S$)
30 FOR I=0 TO N-1
40 X$=X$+MID$(S$,N-I,1)
50 NEXT I
60 PRINT X$
```

IV.
```
10 INPUT"ВВЕДИТЕ СЛОВО";S$
20 INPUT"ПОЗИЦИИ КАКОЙ
   БУКВЫ ИСКАТЬ";B$
30 PRINT"В СЛОВЕ";S$;"
   БУКВА ";B$;" ВСТРЕЧАЕТСЯ В
   ПОЗИЦИЯХ"
40 FOR K=1 TO LEN(S$)
50 IF MID$(S$,K,1)=B$
   THEN PRINT K;
60 NEXT K
70 END
```

V.
```
10 INPUT"ВВЕДИТЕ СТРОКУ";ST$
20 INPUT"ВВЕДИТЕ
   ФРАГМЕНТ";FR$
30 INPUT"С КАКОЙ
   ПОЗИЦИИ";M
40 FOR L=M TO LEN(ST$)-
   LEN(FR$)
50 FOR K=1 TO LEN(FR$)
60 IF MID$(FR$,K,1)<>MID$
   (ST$,L+K-1,1) THEN 100
```

```
70 NEXT K                          100 NEXT L
80 PRINT"С ПОЗИЦИИ";L              110 PRINT"ЗАДАННОГО ФРАГМЕНТА НЕТ"
90 GOTO 120                        120 END
```

VI.
```
10 C¤=INKEY¤
   20 IF C¤="" THEN PRINT" ";
   30 IF C¤<>"" THEN PRINT"ПРАЗДНОСТЬ РОЖДАЕТ ПОРОКИ"
   40 GOTO 10
```

VII.
```
10 REM СЧАСТЛИВЫЙ БИЛЕТ
   20 INPUT"ВВЕДИТЕ НОМЕР
   БИЛЕТА (ШЕСТИЗНАЧНЫЙ)";N¤
   25 S1=0
   27 S2=0
   30 FOR I=1 TO 3
   40 S1=S1+VAL(MID¤(N¤,I,1))
   50 S2=S2+VAL(MID¤(N¤,3+I,1))
   60 NEXT I
   70 IF S1=S2 THEN PRINT"БИЛЕТ
   СЧАСТЛИВЫЙ" ELSE PRINT
   "БИЛЕТ НЕСЧАСТЛИВЫЙ"
   80 END
```

К § 41

Программа может быть такой:

```
2 CLEAR 2000
3 REM БИБЛИОТЕКА
4 CLS
5 READ N
10 DIM L¤(N)
20 DIM M¤(N)
30 DIM S¤(N)
40 DIM A¤(N)
50 DIM A(N)
60 DIM T¤(N)
70 DIM T(N)
80 FOR I=TO N
90 READ M¤(I),S¤(I),A¤(I),T¤(I)
100 A(I)=LEN(A¤(I))
110 T(I) LEN(T¤(I))
120 FOR J=A(I)+1 TO 25
130 A¤(I)=A¤(I)+" "
140 NEXT J
150 FOR K=T(I)+1 TO 25
160 T¤(I)=T¤(I)+" "
170 NEXT K
180 L¤(I)=M¤(I)+S¤(I)+A¤(I)+T¤(I)
190 NEXT I
200 FOR I=1 TO N-1
210 FOR K=I+1 TO N
220 IF L¤(I)> L¤(K) THEN 230
ELSE 260
230 E¤=L¤(I)
240 L¤(I)=L¤(K)
250 L¤(K)=E¤
260 NEXT K
270 NEXT I
280 PRINT
290 PRINT
300 PRINT
310 PRINT"ДЛЯ ПОИСКА
ПО РАЗДЕЛУ НАЖМИТЕ 1"
320 PRINT"ДЛЯ ПОИСКА
ПО ИМЕНИ АВТОРА НАЖМИТЕ 2"
330 PRINT"ДЛЯ ПОИСКА
ПО НАЗВАНИЮ НАЖМИТЕ 3"
340 INPUT"ЧТО ВЫ ХОТИТЕ
СДЕЛАТЬ";O
350 IF O=1 THEN 400
360 IF O=2 THEN 490
370 IF O=3 THEN 620
380 IF O<>1 AND O<>2 AND O<>3
THEN 340
390 GOTO 310
400 CLS
410 INPUT"ВВЕДИТЕ ШИФР
РАЗДЕЛА, КОТОРЫЙ ХОТИТЕ
ПОСМОТРЕТЬ";X¤
420 FOR I=1 TO N
430 IF X¤=MID¤(L¤(1),1,2)
THEN 450
440 GOTO 460
450 PRINT L¤(I)
460 NEXT I
470 GOSUB 770
480 GOTO 280
```

```
490 CLS
500 INPUT"КНИГИ КАКОГО АВТОРА ХОТИТЕ ПОСМОТРЕТЬ";N$
510 Q=LEN(N$)
520 FOR W=Q+1 TO 25
530 N$=N$+1 TO 25
530 N$=N$+" "
540 NEXT W
550 FOR I=1 TO N
560 IF MID$(L$(I),4,25)=N$ THEN 580
570 GOTO 590
580 PRINT L$(I)
590 NEXT I
600 GOSUB 770
610 GOTO 280
620 CLS
630 INPUT"ВВЕДИТЕ НАИМЕНОВАНИЕ КНИГИ";Z$
640 P=LEN(Z$)
650 FOR R=P+1 TO 25
660 Z$=Z$+" "
670 NEXT R
680 FOR I=1 TO N
690 IF MID(L$(I),LEN(L$(I))-25+1,25))=Z$ THEN 710
700 GOTO 720
710 PRINT L$(I)
720 NEXT I
730 GOSUB 770
750 GOTO 280
760 END
770 INPUT"ХОТИТЕ ВЕРНУТЬСЯ В МЕНЮ (Д/Н)";Y$
780 IF Y$="Н" THEN 760
790 IF Y$<>"Д" AND Y$<>"Н" THEN 770
800 CLS
810 RETURN
815 DATA 5
820 DATA 08,5,КУРК,МАТЕМАТИКА
830 DATA 14,3,ЦУЗМЕР,ЧЕЛОВЕК И ЕГО ЗДОРОВЬЕ
840 DATA 21,2,ПУШКИН,ПОВЕСТИ БЕЛКИНА
850 DATA 21,7,ШЕВЧЕНКО, КОБЗАРЬ
```

В этой программе в строке 815 указывается количество переменных (N – число книг). Здесь переменная L$ – полная информация о книге, M$ – раздел, S$ – подраздел, A$ – автор, T$ – название. Эта программа резервирует 25 символов для фамилий авторов и 25 символов для названий произведений, поэтому при вводе данных необходимо не выйти за эти ограничения.

В строке 90 считывается информация из DATA; цикл 120–140 дополняет пробелами до 25 символов в имени автора. Цикл 150–170 делает то же самое с названием книги. Со строки 200 до строки 270 выполняется упорядочение книг по алфавиту.

Цикл 520–540 дополняет фамилию автора до 25 символов. Строка 580 выводит на печать все книги выбранного автора. Строки 650–670 дополняют соответственно название книги до 25 символов. Строка 710 выводит все данные, соответствующие выбранному названию.

Если вам необходимо дополнить список новых книг, добавьте новые строки с операторами DATA и не забудьте при этом соответственно изменить строку 815.

К § 42

Тест-программа может быть такой:

```
10 S=0
12 CLEAR 1000' ОПЕРАТОР РЕЗЕРВИРУЕТ ПОЛЬЗОВАТЕЛЬСКУЮ ПАМЯТЬ (МОЖНО ОБОЙТИСЬ И БЕЗ НЕГО)
15 CLS
```

```
20 FOR P=1 TO 19
25 PRINT
27 PRINT
30 READ U¤,O¤,K
40 PRINT U¤
45 PRINT
50 PRINT O¤
60 T=K
70 INPUT"НАБЕРИТЕ НОМЕР ПРАВИЛЬНОГО ОТВЕТА:";X
80 GOSUB 650
90 NEXT P
100 CLS
110 IF S=0 OR S=1 THEN PRINT"ВЫ ГЕНИЙ!"
120 IF S=2 OR S=3 OR S=4 THEN PRINT"ВЫ ИНТЕЛЛЕКТУАЛ!"
130 IF S=5 OR S=6 THEN PRINT"ВЫ НОРМАЛЬНЫЙ ЧЕЛОВЕК!"
140 IF S=7 OR S=8 THEN PRINT"ВЫ РЯДОВОЙ ИДИОТ!"
150 IF S=9 OR S=10 THEN PRINT"ВЫ РЯДОВОЙ ИДИОТ С
ПОБОЧНЫМИ ЯВЛЕНИЯМИ!"
160 IF S=11 OR S=12 THEN PRINT"ВЫ АБСОЛЮТНЫЙ ИДИОТ!"
170 IF S=13 OR S=14 OR S=15 THEN PRINT"ВЫ НЕ СПОСОБНЫ
МЫСЛИТЬ!"
180 IF S=16 OR S=17 OR S=18 OR S=19 THEN PRINT"ВАС
НЕОБХОДИМО ИЗОЛИРОВАТЬ ОТ ОБЩЕСТВА!"
200 END
210 DATA"ПРОФЕССОР ЛОЖИТСЯ СПАТЬ В 8 ЧАСОВ ВЕЧЕРА, А
БУДИЛЬНИК ЗАВОДИТ НА 9 ЧАСОВ УТРА. СКОЛЬКО БУДЕТ СПАТЬ
ПРОФЕССОР?"
220 DATA"1. ОДИН ЧАС; 2. 13 ЧАСОВ; 3. 12 ЧАСОВ",1
230 DATA"МОЖЕТ ЛИ МУЖЧИНА ЖЕНИТЬСЯ НА СЕСТРЕ СВОЕЙ
ВДОВЫ?"
240 DATA"1. ДА; 2. НЕТ",2
250 DATA"ЕСТЬ ЛИ 7 НОЯБРЯ В АВСТРАЛИИ?"
260 DATA"1. НЕТ; 2. ЕСТЬ",2
270 DATA"У МАМЕДА 10 ОВЕЦ. ВСЕ, КРОМЕ 9 СДОХЛИ. СКОЛЬКО
ОСТАЛОСЬ ОВЕЦ?"
280 DATA"1. 1.ОВЦА; 2. 9 ОВЕЦ; 3. 10 ОВЕЦ",2
290 DATA"ВЫ - ПИЛОТ САМОЛЕТА, ЛЕТЯЩЕГО ИЗ ГАВАНЫ В
МОСКВУ С ДВУМЯ ПЕРЕСАДКАМИ В АЛЖИРЕ. СКОЛЬКО ЛЕТ
ПИЛОТУ?"
300 DATA"1. 30 ЛЕТ; 2. 45 ЛЕТ; 3. СТОЛЬКО ЛЕТ, СКОЛЬКО
ВАМ",3
310 DATA"ОБЫЧНО МЕСЯЦ ЗАКАНЧИВАЕТСЯ 30-М ИЛИ 31-
М ЧИСЛОМ. В КАКОМ МЕСЯЦЕ ЕСТЬ 28-Е ЧИСЛО?"
320 DATA"1. В ФЕВРАЛЕ; 2. ВО ВСЕХ; 3. ТОЛЬКО В ТОМ, ГДЕ ЕСТЬ
29-Е ЧИСЛО",2
330 DATA"ВЫ ЗАХОДИТЕ В МАЛОЗНАКОМУЮ КОМНАТУ, КОТОРАЯ
ЗАТЕМНЕНА. В НЕЙ ЕСТЬ ДВЕ ЛАМПЫ - ГАЗОВАЯ И БЕНЗИНОВАЯ.
ЧТО ВЫ ЗАЖЖЕТЕ В ПЕРВУЮ ОЧЕРЕДЬ?"
```

```
340 DATA"1. СПИЧКУ; 2. ГАЗОВУЮ ЛАМПУ; 3. БЕНЗИНОВУЮ
ЛАМПУ",1
350 DATA"ОДИН ПОЕЗД ИДЕТ ИЗ КИЕВА В ДОНЕЦК, А ДРУГОЙ -
ИЗ ДОНЕЦКА В КИЕВ. ВЫШЛИ ОНИ ОДНОВРЕМЕННО, НО СКОРОСТЬ
ПЕРВОГО В ТРИ РАЗА БОЛЬШЕ СКОРОСТИ ВТОРОГО. КАКОЙ ПОЕЗД
БУДЕТ ДАЛЬШЕ ОТ КИЕВА В МОМЕНТ ИХ ВСТРЕЧИ?"
360 DATA"1. КОТОРЫЙ ИДЕТ ИЗ КИЕВА В ДОНЕЦК; 2. КОТОРЫЙ
ИДЕТ ИЗ ДОНЕЦКА В КИЕВ; 3. ОДИНАКОВО",3
370 DATA"ОТЕЦ С СЫНОМ ПОПАЛИ В КАТАСТРОФУ. ОТЕЦ СКОНЧАЛСЯ
В ГОСПИТАЛЕ. К СЫНУ В ПАЛАТУ ЗАХОДИТ ХИРУРГ И ГОВОРИТ,
ПОКАЗЫВАЯ НА НЕГО: ЭТО МОЙ СЫН. МОГУТ ЛИ ЭТИ СЛОВА БЫТЬ
ПРАВДОЙ?"
380 DATA"1. ДА; 2. НЕТ",1
390 DATA"АРХЕОЛОГИ НАШЛИ МОНЕТУ, ДАТИРОВАННУЮ 35 ГОДОМ
ДО НАШЕЙ ЭРЫ. ВОЗМОЖНО ЛИ ЭТО?"
400 DATA"1. ДА; 2. НЕТ",2
410 DATA"ПАЛКУ НУЖНО РАСПИЛИТЬ НА 12 ЧАСТЕЙ. СКОЛЬКО
ПОТРЕБУЕТСЯ РАСПИЛОВ?"
420 DATA"1. 11; 2. 12; 3. 13",1
430 DATA"НА РУКАХ - 10 ПАЛЬЦЕВ. СКОЛЬКО ПАЛЬЦЕВ НА
10 РУКАХ?"
440 DATA"1. 100; 2. 50; 3. 10",2
450 DATA"В КАКОМ КОЛИЧЕСТВЕ ВЗЯЛ НОЙ ЗВЕРЕЙ В СВОЙ
КОВЧЕГ?"
460 DATA"1. 3500 ВИДОВ; 2. 397400 ОСОБЕЙ; 3. КАЖДОЙ
ТВАРИ - ПО ПАРЕ",3
470 DATA"ВРАЧ ПРОПИСАЛ БОЛЬНОМУ ТРИ УКОЛА ЧЕРЕЗ
КАЖДЫЕ ПОЛЧАСА. СКОЛЬКО ПОТРЕБУЕТСЯ ВРЕМЕНИ, ЧТОБЫ
СДЕЛАТЬ ВСЕ УКОЛЫ?"
480 DATA"1. ОДИН ЧАС; 2. ПОЛТОРА ЧАСА; 3. ДВА ЧАСА",1
490 DATA"СКОЛЬКО ЦИФР 9 В РЯДУ ЧИСЕЛ ОТ '1' ДО '100'?"
500 DATA"1. 11; 2. 10; 3. 20",3
510 DATA"ОДИНОКИЙ СТОРОЖ УМЕР НОЧЬЮ. ДАДУТ ЛИ ЕМУ
ПЕНСИЮ?"
520 DATA"1. НЕТ; 2. ДА",1
530 DATA"ГОРЕЛО 7 СВЕЧЕЙ. ТРИ ПОГАСЛО. СКОЛЬКО СВЕЧЕЙ
ОСТАЛОСЬ?"
540 DATA"1. 1. 4 СВЕЧИ; 2. 3 СВЕЧИ; 3. 7 СВЕЧЕЙ",2
550 DATA"КИРПИЧ ВЕСИТ ОДИН КИЛОГРАММ ПЛЮС ЕЩЕ
ПОЛКИРПИЧА. СКОЛЬКО ВЕСИТ КИРПИЧ?"
560 DATA"1. 1 КГ; 2. 1,5 КГ; 3. 2 КГ",1
570 DATA"ПОД КАКИМ КУСТОМ СИДИТ ЗАЯЦ ВО ВРЕМЯ ДОЖДЯ?"
580 DATA"1. ПОД КЛЕНОВЫМ; 2. ПОД СУХИМ; 3. ПОД МОКРЫМ",3
650 IF X<>T THEN PRINT"НЕВЕРНО! ПРАВИЛЬНЫМ ОТВЕТОМ
ЯВЛЯЕТСЯ ";K;" ОТВЕТ" ELSE 680
660 S=S+1
670 GOTO 690
680 IF X=T THEN PRINT"ВЕРНО!"
690 RETURN
```

К § 46

I. Не являются суждениями: 1, 3, 5, 7, 9
Суждения: 2, 4, 6, 8, 10 (2 – истинно; 4 – ложно; 6 – истинно; 8 – истинно; 10 – ложно).

II. С = "то такое число делится на 7" (Это признак делимости на 7)

III. Общие суждения: 1, 2, 4, 6, 7, 10.Частные: 3, 5, 8, 9. Истинные суждения: 1, 2, 3, 4, 5, 7, 9. Ложные суждения: 6, 8, 10.

IV. Да, все три суждения равносильны между собой: A=B=C. Все три высказывания определяют прямоугольник.

V. 1) A = "три стороны одного треугольника соответственно равны трем сторонам другого"; B = "такие треугольники равны". 2) C = "есть мера вещей"; D – "существуют известные границы". 3) U = "разрешаются от бремени горы"; F = "рождается смешная мышь". 4) G = "сумма цифр числа через одну равна сумме остальных цифр через одну"; Q = "разность этих сумм делится на 11"; H = "данные числа делятся на 11". 5) K = "'Шахтер' выиграл встречу у 'Динамо'"; M= "'Шахтер' сыграл с 'Таврией' вничью"; L = "'Шахтер' сыграл со 'Спартаком' вничью". 6) N = "студент запланировал подготовиться к зачету"; O = "студент запланировал побывать на тренировке"; P = "студент запланировал почитать интересную книгу"; R = "студент запланировал поиграть в шахматы". 7) S = "завтра будет туман"; T = "мы не сможем вылеть на соревнования".

К § 47

I. 1) "Если хотя бы один из сомножителей равен 0" – предпосылка; "произведение двух чисел равно 0" – заключение. Суждение истинно; 2) Если A*B>0 – предпосылка; A>0 и B>0 – заключение; суждение – ложно. 3) "Если в треугольнике все стороны равны" – предпосылка; "все углы его равны" – заключение. Суждение истинно. 4) "Если на улице сыро" – предпосылка; "Прошел дождь" – заключение. Суждение ложно, т.к. на улице может быть сыро, если прошла поливочная машина.

II. 1) "Каждый равносторонний шестиугольник является также равноугольным"; 2) "существует правильный многогранник с любым наперед заданным числом граней N≥3". Такое суждение ложно, т.к. число граней должно быть >4. (Вспомните правильный тетраэдр – число граней уже равно 3).

III. Вариант I: 1-1-й вопрос: "2*2=5?"; 2-й вопрос: "Эта дорога ведет к озеру?"; вариант II: "Правда ли, что 2*2=5 и эта дорога ведет к озеру?" (здесь требуется анализ ответов); вариант III: "Что ответил бы твой приятель на вопрос: "Ведет ли эта дорога к озеру?" (Здесь тоже требуется анализ ответов).

IV. Мудрец рассуждал так:
– Я вижу перед собой два колпака. Предложим, что на мне белый. Тогда второй мудрец, видя перед собой черный и белый колпаки,

должен рассуждать так: "Если бы на мне был тоже белый колпак, то третий сразу бы догадался и заявил, что у него черный. Но он молчит, значит, на мне не белый, а черный". А так как второй не говорит этого, значит, на мне тоже черный.

V. Для данных трех суждений допустимы три комбинации типа "истина–ложь": И – Л – Л; И – Л – И; И – И – Л.

Единственной непротиворечивой комбинацией является в данном случае комбинация И – Л – И, означающая, что у Димы вообще нет книг.

К § 48

I. 1) A = "Утром мы пойдем на рыбалку"; B = "Утром мы позагораем"; C = "Уху варить будем" ; $\overline{C}$ = "Уху варить не будем". **A*B*$\overline{C}$**

2) X = "Число 156 делится на 3"; Y = "Число 156 делится на 4"; $\overline{Z}$ = "Число 156 не делится на 7"; **X*Y*$\overline{Z}$**;

3) E = "Я поступлю на экономический факультет"; K = "Я поступлю на юридический факультет"; L = "Я поступлю на факультет иностранных языков". **E+K+L**;

4) M = "Я поеду на лечение в Трускавец"; N = "Я поеду на лечение в Нимиров"; O = "Мой класс поедет на экскурсию в Ялту"; P = "Мой класс поедет на экскурсию в Севастополь". **(M+N)*(O+P)**;

5) A = "Завтра будет дождь"; B = "Мы пойдем купаться на речку"; C = "Мы пойдем собирать грибы в лес". **$\overline{A}$ =>(B+C)**;

6) A = "Число делится на 6"; B = "Число делится на 2"; C = "Число делится на 3"; X = "Число делится на 6 тогда и только тогда, когда оно делится на 2 и на 3". **X = (A≡(B*C))**;

7) X – исходное сложное суждение; Y = "Он был ромбом"; Z = "Он имел прямой угол"; K = "Он был прямоугольником"; L = "Он имел равные смежные стороны"; M = "Был квадратом". **X= (M≡ (Y*Z)+(K*L))**.

II. а) A+B*C "Доктор Уотсон – отставной офицер или друг знаменитого сыщика, и он окончил лондонский университет"; б) B*C*$\overline{A}$ "Доктор Уотсон – друг знаменитого сыщика, и он окончил лондонский университет, и он – не отставной офицер"; в) $\overline{C}$*$\overline{B}$*$\overline{A}$ "Доктор Уотсон не оканчивал лондонскогого университета, не был другом знаменитого сыщика, и он не отставной офицер"; г) B⇒ (C*A) "Если доктор Уотсон – друг знаменитого сыщика, то он окончил лондонский университет и отставной офицер"; д) (C+A)⇒B "Если доктор Уотсон окончил лондонский университет или отставной офицер, то он друг знаменитого сыщика"; е) ($\overline{A}$*$\overline{C}$)⇒B "Если доктор Уотсон не отставной офицер и не оканчивал лондонского университета, то он друг знаменитого сыщика";

III. Дизъюнкцию $\overline{A}$+B+$\overline{C}$ можно представить ($\overline{A}$+B)+$\overline{C}$ (по распределительному закону), тогда эта дизъюнкция будет ложна, когда ложна каждая входящая часть (смотри последнюю строку таблицы), тогда ($\overline{A}$+B)=0 и $\overline{C}$=0, отсюда C=1, а так как $\overline{A}$+B=0, следовательно $\overline{A}$=0 и B=0. Отсюда A=1. Вывод: A=1; B=0; C=1.

IV. 1) $A*\bar{B}$

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

2) $A*B*\bar{C}$

A	B	C	A*B	$\bar{C}$	$A*B*\bar{C}$
1	1	1	1	0	0
1	0	1	0	0	0
0	1	1	0	0	0
0	0	1	0	0	0
1	1	0	1	1	1
1	0	0	0	1	0
0	1	0	0	1	0
0	0	0	0	1	0

3) $\bar{A}*\bar{B}*\bar{C}$

A	B	C	$\bar{A}$	$\bar{B}$	$\bar{A}*\bar{B}$	$\bar{C}$	$\bar{A}*\bar{B}*\bar{C}$
1	1	1	0	0	0	0	0
1	0	1	0	1	0	0	0
0	1	1	1	0	0	0	0
0	0	1	1	1	1	0	0
1	1	0	0	0	0	1	0
1	0	0	0	1	0	1	0
0	1	0	1	0	0	1	0
0	0	0	1	1	1	1	1

4) $A*(B+\bar{A})$

A	$\bar{A}$	B	$B+\bar{A}$	$A*(B+\bar{A})$
1	0	1	1	1
1	0	0	0	0
0	1	1	1	0
0	1	0	1	0

5) $(A*B)+\bar{C}$

A	B	A*B	C	$\bar{C}$	$(A*B)+\bar{C}$
1	1	1	1	0	1
1	0	0	1	0	0
0	1	0	1	0	0
1	1	1	0	1	1
1	0	0	0	1	1
0	1	0	0	1	1
0	0	0	0	1	1

6) $A+(\overline{A*B})$

A	B	A*B	$\overline{A*B}$	$A+(\overline{A*B})$
1	1	1	0	1
1	0	0	1	1
0	1	0	1	1
0	0	0	1	1

7) $(A\Rightarrow B)+\bar{B}$

A	B	$A\Rightarrow B$	$\bar{B}$	$(A\Rightarrow B)+\bar{B}$
1	1	1	0	1
1	0	0	1	1
0	1	1	0	1
0	0	1	1	1

8) $A*B*(A\Rightarrow\bar{B})$

A	B	A*B	$\bar{B}$	$A\Rightarrow\bar{B}$	$A*B*(A\Rightarrow\bar{B})$
1	1	1	0	0	0
1	0	0	1	1	0
0	1	0	0	1	0
0	0	0	1	1	0

9) $\overline{(A*\bar{B})}\Rightarrow(\bar{B}\Rightarrow A)$

A	B	$\bar{B}$	$A*\bar{B}$	$\overline{A*\bar{B}}$	$\bar{B}\Rightarrow A$	$\overline{(A*\bar{B})}\Rightarrow(\bar{B}\Rightarrow A)$
1	1	0	0	1	1	1
1	0	1	1	0	1	1
0	1	0	0	1	1	1
0	0	1	0	1	0	0

10) $\overline{(A*\bar{B})}\equiv(\bar{A}\div(A*B))$

A	B	$\bar{B}$	$A*\bar{B}$	$\overline{A*\bar{B}}$	$A*B$	$\bar{A}$	$\bar{A}\div(A*B)$	$\overline{(A*\bar{B})}\equiv(\bar{A}*(A*B))$
1	1	0	0	1	1	0	1	1
1	0	1	1	0	0	0	0	1
0	1	0	0	1	0	1	1	1
0	0	1	0	1	0	1	1	1

11) $(A\equiv B)*(A*\bar{B})+(\bar{A}*B)$

A	B	$A\equiv B$	$\bar{B}$	$A*\bar{B}$	$(A\equiv B)*(A*\bar{B})$	$\bar{A}$	$\bar{A}*B$	$(A\equiv B)*A*\bar{B})+(\bar{A}*B)$
1	1	1	0	0	0	0	0	0
1	0	0	1	1	0	0	0	0
0	1	0	0	0	0	1	1	1
0	0	1	1	0	0	1	0	0

12) $\overline{(A\Rightarrow(B\Rightarrow C))}\equiv(A*B*\bar{C})$

A	B	C	$B\Rightarrow C$	$A\Rightarrow(B\Rightarrow C)$	$\overline{A\Rightarrow(B\Rightarrow C)}$	$A*B$	$\bar{C}$	$A*B*\bar{C}$	$\overline{(A\Rightarrow(B\Rightarrow C))}\equiv(A*B*\bar{C})$
1	1	1	1	1	0	1	0	0	1
1	0	1	1	1	0	0	0	0	1
0	1	1	1	1	0	0	0	0	1
0	0	1	1	1	0	0	0	0	1
1	1	0	0	0	1	1	1	1	1
1	0	0	1	1	0	0	1	0	1
0	1	0	0	1	0	0	1	0	1
0	0	0	1	1	0	0	1	0	1

V. 1) $\overline{(A\Rightarrow B)}$

A	B	$A\Rightarrow B$	$\overline{A\Rightarrow B}$
1	1	1	0
1	0	0	1
0	1	1	0
0	0	1	0

$A*\bar{B}$

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

Последние колонки одинаковы, следовательно, $\overline{(A\Rightarrow B)}=(A*\bar{B})$

2) $\bar{A}*B$

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

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

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

Последние колонки одинаковы, следовательно, $\bar{A}*B\equiv\overline{(A+\bar{B})}$

3) $A+B*C$

A	B	C	$B*C$	$A+B*C$
1	1	1	1	1
1	0	1	0	1
0	1	1	1	1
0	0	1	0	0

$(A+B)*(A+C)$

A	B	$A+B$	C	$A+C$	$(A+B)*(A+C)$
1	1	1	1	1	1
1	0	1	1	1	1
0	1	1	1	1	1
0	0	0	1	1	0

1	1	0	0	1
1	0	0	0	1
0	1	0	0	0
0	0	0	0	0

1	1	1	0	1	1
1	0	1	0	1	1
0	1	1	0	0	0
0	0	0	0	0	0

Последние колонки одинаковы, следовательно, $(A+B*C)\equiv(A+B)*(A+C)$.

К § 49

I. а) 1) $A+B$ $=$ $\overline{\overline{A}*\overline{B}}$

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

A	B	$\overline{A}$	$\overline{B}$	$\overline{A}*\overline{B}$	$\overline{\overline{A}*\overline{B}}$
1	1	0	0	0	1
1	0	0	1	0	1
0	1	1	0	0	1
0	0	1	1	1	0

2) $A*B$ $=$ $\overline{\overline{A}+\overline{B}}$

A	B	A*B
1	1	1
1	0	0
0	1	0
0	0	0

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

б) $(A\Rightarrow B)=\overline{A}+B$

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

в) $(A\equiv B)=((A*B)+(\overline{A}*\overline{B}))$

A	B	$A\equiv B$	A*B	$\overline{A}$	$\overline{B}$	$\overline{A}*\overline{B}$	$(A*B)+(\overline{A}*\overline{B})$
1	1	1	1	0	0	0	1
1	0	0	0	0	1	0	0
0	1	0	0	1	0	0	0
0	0	1	0	1	1	1	1

II. а) $\underbrace{(P\Rightarrow\overline{C})}_{\text{по 20}}*\underbrace{(C\Rightarrow P)}_{\text{по 20}}=\underbrace{(\overline{P}+\overline{C})}_{\text{по 5}}*(\overline{C}+P)=\overline{P}*\overline{C}+\underbrace{\overline{P}*P}_{\text{по 15}}+\underbrace{\overline{C}*\overline{C}}_{\text{по 8}}+\overline{C}*P=\underbrace{\overline{P}*\overline{C}+0}_{\text{по 3}}+\overline{C}+\underbrace{\overline{C}*P}=$

$=\underbrace{(\overline{P}*\overline{C}+\overline{C}*P)}_{\text{по 5}}+\overline{C}=\underbrace{\overline{C}*(\overline{P}+P)}_{\text{по 14=1}}+\overline{C}=\overline{C}*1+\overline{C}=\underbrace{\overline{C}+\overline{C}}_{\text{по 7}}=\overline{C}$

б) $\underbrace{A*\overline{B}*\overline{C}+\overline{B}*\overline{C}}_{\text{по 5}}+A=\overline{B}*\overline{C}\underbrace{(A+1)}_{\text{по 16}}+A=\overline{B}*\overline{C}*1+A=\overline{B}*\overline{C}+A$

в) $(A*B)+(\overline{A}*B)+(\overline{A}*\overline{B})=\underbrace{((A*B)+(\overline{A}*B))}_{\text{по 5}}+(\overline{A}*\overline{B})=\underbrace{(B*(A+\overline{A}))}_{\text{по 14}}+(\overline{A}*\overline{B})=$

$=B*1+(\overline{A}*\overline{B})=\underbrace{B+\overline{A}*\overline{B}}_{\text{по 10}}=\underbrace{\overline{A}+B}_{\text{по 20}}=(A\Rightarrow B)$

г) $(\bar{A}*P*C)+(A*\bar{P}*C)+(A*P*\bar{C})+(A*P*C)=(\bar{A}*P*C)+(A*\bar{P}*C)+(A*P*\bar{C})+(A*P*C)+\underbrace{(A*P*C)+(A*P*C)}_{\text{добавили по 7}}=((\bar{A}*P*C)+(A*P*C))+((A*\bar{P}*C)+(A*P*C))+$

$+(A*P*\bar{C})+(A*P*C))=P*C*\underbrace{(\bar{A}+A)}_{\text{по 14}}+A*C*\underbrace{(\bar{P}+P)}_{\text{по 14}}+A*P*\underbrace{(\bar{C}+C)}_{\text{по 14}}=P*C*1+A*C*1$

$+A*P*1=P*C+A*C+A*P$

III. а) $(X*Y)\Rightarrow\underbrace{(\bar{X}\Rightarrow Z)}_{\text{по 20}}=\underbrace{(X*Y)\Rightarrow\bar{\bar{X}}+Z)}_{\text{по 13}}=\underbrace{(X*Y)\Rightarrow(X+Z)}_{\text{по 20}}=\underbrace{\bar{X*Y}}_{\text{по 12}}+X+Z=$

$=\bar{X}+\bar{Y}+X+Z=\underbrace{\bar{X}+X}_{\text{по 14}}+\underbrace{\bar{Y}+Z}_{\text{по 1}}=\underbrace{1+Z}_{\text{по 14}}+\bar{Y}=1+\bar{Y}$

б) $\underbrace{(\bar{A}*P)\Rightarrow(\bar{P}*A)}_{\text{по 20}}=\underbrace{\bar{\bar{A}}*P}_{\text{по 12}}+\bar{P}*A=\underbrace{\bar{\bar{A}}+\bar{P}}_{\text{13}}+\underbrace{\bar{P}*A}_{\text{по 9}}=A+\bar{P}+(A+P)*\bar{P}=A+\bar{P}+$

$A*P+\underbrace{P*\bar{P}}_{\text{по 15}}=A+\bar{P}+A*\bar{P}+0=A+\bar{P}+A*\bar{P}=A+\bar{P}*\underbrace{(1+A)}_{\text{по 16}}=A+\bar{P}*1=A+\bar{P}$

IV. а) $\underbrace{A+P+\bar{C}}_{\text{по 11}}=\underbrace{(\bar{\bar{A}*\bar{P}})+\bar{C}}_{\text{по 12}}=\bar{A*\bar{P}*\bar{C}}$

б) $(X+Y)\Rightarrow\underbrace{(\bar{X}\Rightarrow Z)}_{\text{по 20}}=(X+Y)\Rightarrow\underbrace{(\bar{\bar{X}}+Z)}_{\text{по 13}}=\underbrace{(X+Y)\Rightarrow(X+Z)}_{\text{по 20}}=$

$=\underbrace{(\overline{X+Y})}_{\text{по 11}}+(X+Z)=\underbrace{(\bar{X}*\bar{Y})}_{\text{по 10}}+X+Z=\underbrace{\bar{Y}+X+Z}_{\text{по 11}}=\bar{Y}+\underbrace{(\bar{\bar{X}}*\bar{Z})}_{\text{по 12}}=\bar{\bar{Y}}*\bar{\bar{X}}*\bar{\bar{Z}}=\bar{\bar{X}}*\bar{\bar{Y}}*\bar{\bar{Z}}.$

V. Рассуждения: А–"свидетель А сказал правду"; В–"свидетель В сказал правду". $(A\Rightarrow B)+(A*\bar{B})$

VI. График дежурства: Понедельник – Борис. Вторник – Дима. Среда – Володя. Четверг – Гена. Пятница – Андрей.

К § 50

I. Суждение каждого из ребят будет истинным, когда верно первое и второе и неверно третье показание, или верно первое и третье и неверно второе, или верно второе и третье и неверно первое.

Суждение каждого обозначим буквой его имени с цифрой номера показания. Тогда:

$L=L1*L2*\bar{L3}+L1*\bar{L2}*L3+\bar{L1}*L2*L3$

$D=D1*D2*\bar{D3}+D1*\bar{D2}*D3+\bar{D1}*D2*D3$

$T=T1*T2*\bar{T3}+T1*\bar{T2}*T3+\bar{T1}*T2*T3$

$M=M1*M2*\bar{M}$ [illegible] $*\bar{M2}*M3+\bar{M1}*M2*M3$

Но первое и тре[illegible]показания Толи равносильны, следовательно: $T3=T1$, тогда $T=T1*T2*\bar{T1}+T1*\bar{T2}*T1+\bar{T1}*T2*T1=\underbrace{(T1*\bar{T1})}_{\text{по 15 равно 0}}*$

$*T2+\underbrace{(T1*T1)}_{\text{по 8 равно T1}}*\bar{T2}+\underbrace{(\bar{T1}*T1)}_{\text{по 15 равно 0}}*T2=T1*\bar{T2}=1$

равно единице, т.к. это суждение истинно, следовательно, и T1 = 1 и $\overline{T2}$ = 1. То есть первое суждение Толи истинно, а второе – ложно. А так как он сказал: 1. Я не виновен; 2. Это сделал Миша; 3. Дима говорит неправду, утверждая, что я разбил стекло, – то, ясно, стекло разбил не Толя и не Миша.

Но теперь ясно и то, что третье показание Димы, в котором он обвиняет Толю, ложно. Значит: D3=0 и $\overline{D3}$=1, а раз так, то D=D1* *D2*D3. А мы уже нашли, что $\overline{D3}$=1, следовательно: D1=1 и D2=1

Первое и третье показания Димы верны. Выходит, он не виновен. Третье показание Миши противоположно второму показанию Димы: M3=$\overline{D2}$. Значит, M3=0, а $\overline{M3}$=1.Заявление Миши теперь: M=$\overline{M1}$*M2*$\overline{M3}$.

Оно истинно только в том случае, когда M1=1, M2=1, $\overline{M3}$=1. Второе показание Миши истинно: стекло разбил Леня.

II. Нет. Виновен в поломке или Витя, или Коля.

III. Левченко – первое; Сильченко – второе; Васильченко – третье.

IV. Первое место занял 9А, второе – 9В, третье – 9г, четвертое – 9Б.

К § 51

I. а) A+D < 1 OR C–D > = 0; б) (X+Y)^2+C<7.5 AND X>=0; в) SIN(X)< COS(X) OR N<1; г) (NOT(A AND B)) EQV (NOT(A OR (A AND B))); д) NOT(A AND NOT B) IMP NOT B IMP A; е) NOT(A IMP(B IMP C)) EQV A AND B AND NOT C

II. IF Y^2<=X AND Y>=X THEN PRINT "ТОЧКА ПРИНАДЛЕЖИТ ОБЛАСТИ" ELSE PRINT "ТОЧКА НЕ ПРИНАДЛЕЖИТ ОБЛАСТИ"

III. а) Рис. б; б) Рис. а; в) Рис. в

IV. X>–1 AND X<1 AND Y<=1 AND Y>=–1

К § 52

Обозначим высказывания, задающие условие задачи: A1="Адамов был первым"; D2="Дронов – вторым"; E1="Ершов был первым", G2="Глебов – вторым"; G3="Глебов – третьим"; B4="Белов – четвертым"; B5="Белов – пятым"; A2="Адамов – вторым"; D5="Дронов – пятым"; W4="Ветров – четвертым".

Высказывания каждого болельщика о двух спортсменах можно задать формулами:

A1*$\overline{D2}$+$\overline{A1}$*D2=1; E1*$\overline{G2}$+$\overline{E1}$*G2=1; G3*$\overline{B4}$+$\overline{G3}$*B4=1; B5*$\overline{A2}$+ +$\overline{B5}$*A2=1; D5*$\overline{W4}$+$\overline{D5}$*W4=1. [Формулы(1)]

В высказывании каждого болельщика одно высказывание истинно, а другое – ложно. А также ведь ни одно место не было разделено участниками. Это можно задать формулами:

A1*E1=0; A2*D2=0; W4*B4=0; B5*D5=0 или $\overline{A1}$+$\overline{E1}$=1; $\overline{A2}$+$\overline{D2}$=1; $\overline{W4}$+$\overline{B4}$=1; $\overline{B5}$+$\overline{D5}$=1. [Формулы(2)]

И, наконец, учтем, что ни один из спортсменов не может занять два разных места. Это обстоятельство выражается формулами: $A1*A2=0$; $D2*D5=0$; $B4*B5=0$; $G2*G3=0$ или $\overline{A1}+\overline{A2}=1$; $\overline{D2}+\overline{D5}=1$; $\overline{B4}+\overline{B5}=1$; $\overline{G2}+\overline{G3}=1$. [Формулы(3)]

Решить поставленную задачу – это значит учесть все логические условия, задающие содержание и суть взаимосвязей в задаче, т.е. узнать, при каких значениях указанных выше переменных логическое произведение условий (1), (2) и (3) принимает значение 1, то есть истинно.

Итак, решение задачи свелось к задаче вычисления значений сложного логического выражения:

F=(A1*D2+F1*D2)*(E1*G2+E1*G2)*(G3*B4+G3*B4)*(B5*A2+B5*A2)
*(D5*W5+D5*W4)*(F1+E1)*(F2+D2)*(W4+D2)*(W4+D2)*(W4+B4)
(D5+D5)(A1+A2)*(D2+D5)*(B4+B5)*(G2+G3)

А программа выглядит так:

```
10 FOR A1=0 TO 1
20 FOR D2=0 TO 1
30 FOR E1=0 TO 1
40 FOR G2=0 TO 1
50 FOR G3=0 TO 1
60 FOR B4=0 TO 1
70 FOR B5=0 TO 1
80 FOR A2=0 TO 1
90 FOR D5=0 TO 1
100 FOR Ш4=0 TO 1
110 F1=(A1 AND NOT D2)
OR (NOT A1 AND D2)
120 F2=(E1 AND NOT G2)
OR (NOT E1 AND G2)
130 F3=(G3 AND NOT B4)
OR (NOT G3 AND B4)
140 F4=(B5 AND NOT A2)
OR (NOT B5 AND A2)
150 F5=(D5 AND NOT Ш4)
OR (NOT D5 AND Ш4)
160 F6=(NOT A1 OR NOT E1)
AND (NOT A2 OR NOT D2)
170 F7=(NOT B4 OR NOT Ш4)
AND (NOT B5 OR NOT D5)
180 F8=(NOT B4 OR NOT B5)
AND (NOT G2 OR NOT G3)
190 F9=(NOT D2 OR NOT D5)
AND (NOT A1 OR NOT A2)
200 F10=F1 AND F2 AND F3
AND F4 AND F5
210 F11=F6 AND F7 AND F8
AND F9
220 F=F10 AND F11
230 IF F=1 THEN 360
240 NEXT Ш4
250 NEXT D5
260 NEXT A2
270 NEXT B5
280 NEXT B4
290 NEXT G3
300 NEXT G2
310 NEXT E1
320 NEXT D2
330 NEXT A1
340 PRINT"ЗАДАЧА РЕШЕНИЙ НЕ ИМЕЕТ"
350 GOTO 460
360 IF A1=1 THEN PRINT"1-Е АДАМОВ"
370 IF D2=1 THEN PRINT"2-Е ДРОНОВ"
380 IF E1=1 THEN PRINT"1-Е ЕРШОВ"
390 IF G2=1 THEN PRINT"2-Е ГЛЕБОВ"
400 IF G3=1 THEN PRINT"3-Е ГЛЕБОВ"
410 IF B4=1 THEN PRINT"4-Е БЕЛОВ"
420 IF B5=1 THEN PRINT"5-Е БЕЛОВ"
430 IF A2=1 THEN PRINT"2-Е АДАМОВ"
440 IF D5=1 THEN PRINT"5-Е ДРОНОВ"
450 IF Ш4=1 THEN PRINT"4-Е ВЕТРОВ"
460 END
```

ПРЕДМЕТНЫЙ УКАЗАТЕЛЬ

Адаптер, 376
Алгебра
– линейная, 220
– суждений, 335
Алгоритм, 58, 60, 78
– базовый:
"развилка", 183,184,199
"следование",201
"цикл", 187
– графический, 70
– свойства, 61
– структурный, 313
Алфавит Бейсика, 138
Анкета, 287
Антивирусная программа,103
Архиватор, 102
Архитектура ЭВМ:
– внешняя, 11, 24
– внутренняя, 13, 24
Ассемблер, 83
Ассемблирование, 122
Атака вирусная, 104

Базы
– данных, 112
– знаний, 237
Байт, 29, 52
Безусловный переход, 174
Биокомпьютер, 393
Биочип, 291
Бит, 29, 52
Блок–схема, 71
– "переключение", 252
– "развилка неполная", 184, 200
– "развилка полная", 183, 200
– "следование", 201
– "цикл до", 196, 201
– "цикл пока", 187, 200
Блочная структура, 86
Булева алгебра, 346
Буль Джордж, 346
Быстродействие процессора, 374
Ввод
– с клавиатуры, 19
Винер Норберт, 61
Виртуальная реальность,396
Вирус компьютерный, 103
Вывод умозаключений, 330
Высказывание (логическое), 325
– истинное, 326
– ложное, 326
– общее, 328
– простое, 328
– равносильное (эквивалентное), 328
– составное (сложное), 328
– частное, 327

Гигабайт, 30
Глушков В.М., 373
График функции:

– y=x, 213
– $y=x^2$, 214
– $y=\sqrt{x}$, 215
Графический редактор, 108
– вычерчивающий (деловая графика), 109
– рисования, 108

Датчик случайных чисел, 283
Двоичный код (разряд), 36
Дедукция, 333
Дерево перебора, 88
Дешифровка, 276
Дизъюнкция, 337
Диск
– гибкий магнитный, 91
– винчестерский (жесткий), 92, 93
– системный, 97
– структура, 92
– разметка, 92
Дискета (флоппи–диск), 91
Дисковод, 375
Дисплей, 376
– графический, 377
– монохромный, 377
– цветной, 377
Дорожка (магнитная), 92
Драйвер, 96

Заключение (логическое), 331
Запоминающее устройство
– оперативное (ОЗУ), 9
– постоянное (ПЗУ), 9
Зарезервированные слова языка Бейсик, 221
Зацикливание программы, 174
Знак присваивания, 72
Знаки операций
– арифметические, 144
– логические, 145
– отношения, 144
Зона (в операторе PRINT), 158

Идентификатор, 141
Инверсия (лог. отриц. "не"), 336
Имя файла, 95
Импликация, 340
Индукция, 333
Инициализация магнитного диска, 93
Инструментальная среда, 116, 117
– собственная, 116
Интегрированные программные средства, 113
Интерпретатор, 122
Интерпретация, 84, 122
Интерфейс, 117, 122
– графический, 115
Информатика, 8, 24
Информация, 27, 52
– количество, 28
Искусственный интеллект, 88, 320
Исполнитель, 12
Истинность (логическая), 326

Каталог, 95
– древовидная структура,97
– корневой, 98
– уровневый подкаталог, 98
– прямого доступа, 95
– разделы, 98
– родительский, 98
– текущий, 98
Квантор, 327
Килобайт, 30
Клавиатура, 12
Клавиши, 16
– программируемые, 16
– функциональные, 16
Код, 40
– двоично–десятичный, 49
Кодирование сообщений, 272
– ключевой фразой, 275
– с ключевым числом, 274
– с перестановочным кодом, 275

– с цикловым кодом, 274
Команда, 130
– системная:
delete, 134, 314
list, 132, 314
load, 136, 314
new, 134, 314
run, 131, 314
save, 136, 314
Командная строка, 101
Комментарий (rem), 175
Компакт-диск (CD ROM), 383
Компилятор, 122
Компиляция, 84, 121
Компьютер, 7, 24
Конкатенация, 145
Константа, 140
– числовая, 140
– символьная, 141
– целая, 140
Конструкция языка языка Бейсик, 140
– константы:
символьные, 141
целые, 142
числовые, 140
– переменные, 141
Конъюнкция, 336
Курсор, 18

Логика, 320
– математическая, 323
– формальная, 321
Логические операции, 356
Логическое отрицание, 336, 369
Логическое следование, 340, 369
Логическое сложение, 337, 369
– объединяющее, 338, 339
– разъединяющее (строгое), 338, 339
Логическое умножение, 336, 368
Ложность (логическая), 326
Локальная сеть, 116

Магистраль, 13
Магнитофон, 13, 169
Массив, 143, 220, 315
– двухмерный, 143, 221
– одномерный, 143, 221
– с перекрестными ссылками, 235
– трехмерный, 222
Матрица, 219
– квадратная, 216
– прямоугольная, 217
Мегабайт, 30
Меню, 18
Метка, 130
– магнитная, 95
Микропроцессор (МП), 13, 374
Модем, 382
Молекулярный компьютер, 393
Монитор, 377
– графический, 377
– монохромный, 377
– цветной, 377
Мультимедиа, 383
Мультипрограммирование, 114
"Мышь", 12, 382

Нейман Фон Джон, 9
Нортон (Питер), 100
– интегратор, 100
– командер, 100
– утилит, 100
Нумерация строк, 130

Обеспечение
– математическое, 81, 121
– программное, 121
– системное, 122
Объем памяти, 29, 31
Оверлей, 307

Окно (экранное), 101
Оперативное запоминающее устройство (ОЗУ), 9
Оператор, 83, 130, 314
Оператор Бейсика:
CLS, 135
DIM, 144
END, 155
FOR/NEXT, 178, 190,315
GOSUB/RETURN, 239
GOTO, 173, 314
INPUT, 166, 315
IF/THEN, 178, 314
LET, 149, 152, 314
ON GOSUB, 251
ON GOTO, 250
PRINT, 154, 314,
READ/DATA, 226, 315
REM, 175
STOP, 150, 155
Операционная система (ОС), 90, 121
– стандарт ОС, 97
– MS–DOS, 97
– RT–11 (Рафос, Фодос), 97
Операционная среда, 114
Определитель матрицы, 219
Оптимизация, 297
Отрицание (логическое), 335
Отладка программы, 303
Очерчивание, 320

Память (машинная), 29
Панель, 101
Параметр цикла, 191
Переключение, 252
Переменная, 141
– символьная, 141
– управляющая, 191
– целая, 142
– числовая, 141
Переходы в программе, 172
– безусловный (по опер. GOTO), 174
– условный (по опер. IF/THEN), 178
Печать
– на экран, 154
– вслепую, 20
Плоттер, 12
Подпрограмма, 239, 315
Понятие (логическое), 325
Постоянное запоминающее устройство (ПЗУ), 9
Посылка (логическая), 340
Правило
– логических операций, 341, 357
– перевода из восм. сист. счисл. в двоичн., 49
– перевода из дес. сист. счисл. в двоичн., 39
– перевода из шестн. сист. счисл. в двоичн., 50
– операторов GOSUB/RETURN, 242
– оператора LET, 152
– операторов READ/DATA, 227
– функции RND, 286
Предикат, 327
Пробел, 41, 139
Программа, 129, 313
– анкета, 287
– антивирусная, 103
– архиватор, 103
– база данных, 112
– вычерчивающая, 109
– графический редактор, 109
– интегрированная, 113
– интерпретирующая, 122
– информационно–поисковая система (ИПС), 112
– компилирующая, 121
– оболочка, 100
– контролирующая, 287
– оверлейная, 307
– прикладная, 123
– прикладных сред, 114

– рисования, 108
– системная, 99, 122
– системы управления базами данных (СУБД), 111, 113
– скелетная, 305
– текстовый редактор, 167
– транслирующая, 121
– трассировочная, 308
– удобочитаемая, 306
– универсальная, 64
– электронные таблицы, 111
– эмулятор, 122
Программирование, 313
– в машинных кодах, 83
– деревьями, 88
– доказательное, 319
– логическое, 320
– списками, 88
– структурное, 88
Прорезь защиты записи (на дискете), 92
Процессор, 9, 374
– I4004, 374
– I8088, 374
– I8086, 374
– I80286, 375
– I80386 SX, 375
– I80486 DX, 375
– I 586 (PENTIUM), 375
– MOTOROLA 6800, 375
– MOTOROLA 68000, 375
– Z80, 374
– PENTIUM, 375, 390
Предпосылка (логическая), 324, 331
Принтер, 12, 378
– высокой печати, 379
– лазерный, 380
– матричный, 378
– струйный, 379
Присваивания знак, 72, 179
Присвоить, 72, 179
Принцип программного управления (программируемый режим), 129
ПЭВМ, 11
– APPLE, 192, 388
– IBM PC, 193, 194, 389
Псевдослучайные числа, 283

Редактор
– графический, 108
– текстовый, 106
Режим калькулятора, 127
Разметка (магнт. диска), 95
Разрешение:
– монитора, 376
– печати, 380
Разрядность процессора, 374
Рассуждение (логическое), 331
Расширение имени файлов, 94
Режим калькулятора, 127
Рекурсия, 246
Респондент, 288

Световое перо, 12
Сектор (магн. диска), 92
Система счисления
– восьмеричная, 12, 53
– двоичная, 36, 53
– десятичная, 33
– непозиционная, 34
– позиционная, 34
– шестнадцатиричная, 49,53
Система программирования, 90, 116
Системные команды, 95
DELETE, 95, 134
DIR, 95
ERASE, 95
RENUM, 95
TYPE, 95
Системный блок, 12
Система:
– информационно–поисковая (ИПС), 112
– реляционная, 112

Системы управления базами данных (СУБД), 111
Сканер, 381
– настольный, 381
– ручной, 381
Следствие (логическое), 340
Слепая печать на клавиатуре, 20
Служебные слова, 221
Случайные числа, 283
Сортировка, 278
– методом пузырька, 279
– методом Шелла, 282
Список, 88
Среда
– инструментальная, 117
– прикладная, 114
Стандарт общепризнанный
– на компьютеры, 390
– на операционную систему, 97
Стандартные расширения имен файлов, 94
Столбец матрицы, 223
Строка матрицы, 223
Строка программная (см. метка), 130
Структурное программирование (подход), 88, 199
Суждение (логическое), 325
– общее, 328
– простое, 328
– сложное, 328
– составное, 329
– частное, 327

Таблица
– электронная, 109
Тавтология, 348
Тактовая частота (процессора), 374
Теория массового обслуживания, 299
Транслятор, 84, 121
Трансляция, 84
Текстовый редактор, 106
Тело цикла, 191
Тестирование программы, 303
Технология решения задач на ЭВМ, 301
Том (на каталоге), 95
– прямого доступа, 95

Увеличение скорости выполнения программы, 308
Умозаключение (логическое), 325
– по аналогии, 332
Условие (логическое), 331
Устройство ввода–вывода, 10
Устройство печати, 12
Утилит (программа), 101, 116

Файл, 91, 32, 123
– данных, 311
– расширение имени, 94, 95
Флоппи–диск, 91
Формализация задачи, 302
Формат (в операторе PRINT):
– автоматического смещения изображения (с функцией AT), 162
– зонный, 158
– плотный, 157
– пропуска пробелов (с функцией SPC), 160
– табуляции (с функцией TAB), 160
Форматирование (магн. диска), 93
Функция:
– пользователя: DEF FN, 252
– символьная:
ASC, 263
CHR, 263
INKEY¤, 262
INSTR, 263
LEN, 256
MID¤, 258

STR¤, 270
STRING¤, 267
VAL, 268
– стандартная числовая:
ABS, 146
ATAN, 147
COS, 147
EXP, 146
INT, 147
LOG, 146
PI, 147
RND, 147
SGN, 147
SIN, 147
SQR, 146
TAN, 147
– табуляции, 160

Хранение данных, 310

Цикл
– бесконечный, 174
– вложенный, 208
– "до", 196
– "контролируемый", 190
– "пока", 187

Частота (трактовая), 374
Чип, 291

Шаг цикла, 194
Шеннон Клод, 30

Эквивалентность (логическая), 328
Экспертная система, 237
Экономия места в памяти ЭВМ, 308
Электронно-вычислительная машина (ЭВМ), 7
Электронные таблицы, 109
Эмуляция (программ), 122
Этапы решения задач на ЭВМ, 301

Ядро ОС, 99
Язык
– команд, 95
– компьютера, 36, 52
– машинный, 83
– программирования, 82, 122
алгоритмический, 84
высокого уровня, 90, 123
искусственного интеллекта, 88
машинно-ориентированный, 83, 90
низкого уровня, 83, 122
процедурный, 87
Языки программирования:
Ада, 88
Алгол, 86
Ассемблер, 83, 90
Бейсик, 86, 119
GPSS, 89
Кобол, 86
Кодасил, 89
Лисп, 88
Лого, 89
Паскаль, 88
PL/1, 87
Пролог, 88
Робик, 89
Си, 89
Симкрит, 89
Симула, 89
Форт, 89
Фортран, 85

СПИСОК ЛИТЕРАТУРЫ

1. ***Коляда М.Г.*** и др. Методические указания по работе на учебном компьютере научного центра (УК-НЦ): Части I и II. – Донецк, 1992.

2. ***Коляда М.Г., Петренко А.Г.*** Методические указания к использованию алгоритмического языка Бейсик при составлении контролирующих программ по физике: Для студентов и учителей средних школ. – Донецк: ДонГУ, 1989.

3. Компьютерные программы учебного назначения: Тезисы докладов I, II и III международной конференции/ Отв. ред. Г.А.Атанов. – Донецк: ДонГУ, 1993, 1994, 1995.

4. ***Горбачевский Б.*** Оружие мысли. – М.: Сов. Россия, 1971.

5. Осваиваем микрокомпьютер: Книги I и II. – М.: Мир, 1989.

6. ***Урнов В.А., Климов Д.Ю.*** Преподавание информатики в компьютерном классе. Из опыта работы: Книга для учителя. – М.: Просвещение, 1990.

7. ***Пекелис В.*** Маленькая энциклопедия о большой кибернетике. – М.: Дет. лит., 1970.

8. ***Светозарова Г.И.*** и др. Практикум по программированию на языке Бейсик. – М., 1988.

9. ***Лапчик М.П.*** Вычисления. Алгоритмизация. Программирование. – М.: Просвещение, 1988.

10. ***Кергаль И.*** Методы программирования на Бейсике (с упражнениями). – М.: Мир, 1991.

11. ***Растригин Л.А.*** С компьютером наедине (Серия: массовая радиобиблиотека). – М.: Радио и связь, 1990.

12. ***Верлань А.Ф., Касаткин В.Н.*** Основы информатики и вычислительной техники. – Киев: Рад. шк., 1987.

13. ***Шевченко В.Е.*** Некоторые способы решения логических задач. – Киев: Вища шк., 1979.

14. ***Геворкян Г.Х., Семенов В.Н.*** Бейсик – это просто. – М.: Радио и связь, 1989.

15. ***Кершан Б.*** и др. Основы компьютерной грамотности. – М.: Мир, 1989.

16. ***Храмов Ю.А.*** Биография физики: Хронологический справочник. – Киев: Техніка, 1983.

17. ***Гутер Р.С., Полунов Ю.Л.*** От Абака до компьютера. – М.: Знание, 1981.

18. ***Бильдюкевич Е.В.*** и др. ЭВМ и микропроцессор. – Минск: Нар. освета, 1990.

19. Информатика в понятиях и терминах: Книга для учащихся/ Под ред. В.А.Извозчикова – М.: Просвещение, 1991.

ОГЛАВЛЕНИЕ

От автора 3

Глава I. Общее знакомство с ЭВМ 5

§ 1. Что такое информатика? 7
§ 2. Общая схема устройства ЭВМ 8
§ 3. Архитектура ЭВМ 11
§ 4. Первый раз в дисплейном классе. Правила работы 14
§ 5. Азы общения с компьютером 17
Подведем итоги 24

Глава II. Представление и измерение информации в ЭВМ 25

§ 6. Что такое информация? 27
§ 7. Количество информации 28
§ 8. Единицы количества информации. Объем памяти 29
§ 9. Десятичная система счисления 33
§ 10. Двоичная система счисления 36
§ 11. Арифметические операции в двоичной системе счисления 44
§ 12. Восьмеричная система счисления и связь ее с двоичной и десятичной 47
§ 13. Шестнадцатиричная система счисления. Двоично-десятичный код 49
Подведем итоги 52

Глава III. Начала алгоритмизации 55

§ 14. Понятие алгоритма 58
§ 15. Свойства алгоритмов 61
§ 16. Графический способ представления алгоритмов 70
Подведем итоги 78

Глава IV. Математическое обеспечение ЭВМ 79

§ 17. Языки программирования 82
§ 18. Понятие об операционной системе. Файловая система.. 90
§ 19. Программное обеспечение ПЭВМ 99
§ 20. Почему мы избрали Бейсик? 119
Подведем итоги 121

Глава V. Начальные понятия языка Бейсик 125

§ 21. Режим калькулятора 127
§ 22. Организация программ 129
§ 23. Основные системные команды языка Бейсик 131
§ 24. Алфавит 138
§ 25. Конструкции языка Бейсик 140
§ 26. Оператор LET (присвоить) 149
§ 27. Оператор PRINT (печать на экран) 154
§ 28. Оператор INPUT (ввод с клавиатуры) 166
§ 29. Переходы в программе 172
§ 30. Оператор IF... THEN (Если... тогда) 178
§ 31. Оператор цикла FOR... NEXT (Для... потом) 190
§ 32. Структурная организация алгоритмов 199
§ 33. Дополнительные возможности при организации циклов...208
§ 34. Матрицы и массивы 216
§ 35. Оператор READ... DATA (читать... данные) 226
§ 36. Примеры использования массивов 231
§ 37. Подпрограммы. Операторы GOSUB... RETURN (Иди к подпрограмме... возврат) 239
§ 38. Операторы ON... GOTO, ON... GOSUB. Функция пользователя 249
§ 39. Символьные функции 256
§ 40. Использование ЭВМ для кодирования сообщений 272
§ 41. Порядок и беспорядок 278
§ 42. Программы анкеты 287
§ 43. Вопросы оптимизации 297
§ 44. Искусство решения задач на ЭВМ 301
Подведем итоги 313

Глава VI. Логика в вычислительной технике и программировании 317

§ 45. Логика и математика 320
§ 46. Понятие, суждение и умозаключение 325
§ 47. Вывод умозаключений 330

§ 48. Алгебра суждений 335
§ 49. Булева алгебра 346
§ 50. Решение задач с помощью алгебры суждений 351
§ 51. Логические операции в программировании 356
§ 52. Программирование логических задач 361
Подведем итоги 368

Глава VII. Заключение 371

§ 53. Аппаратное обеспечение персональных компьютеров . 373
§ 54. Мультимедиа. CD ROM технологии 383
§ 55. Вверх по спирали компьютерного совершенствования..388

Ответы и решения 397
Предметный указатель 435
Список литературы 442

ИЗДАТЕЛЬСКО-КНИГОТОРГОВОЕ ОБЪЕДИНЕНИЕ

Выпускает серии:

"Белый лотос". Знахари, лекари, духовные целители делятся своими секретами с читателями.

"Посох Эскулапа". Книги по нетрадиционной медицине помогут сохранить, укрепить здоровье, восстановить духовное равновесие, обрести гармонию тела и духа.

"Я б в астрологи пошел". Книги этой серии знакомят с основными астрологическими понятиями и дают сведения по истории астрономии.

"Преступники и преступления". Перед вами пройдет галерея террористов и заговорщиков, наемных убийц и маньяков, воров и мошенников с древности до наших дней.

"Школа эйдетики". Эйдос — образ. Развитие образной памяти поможет открыть у читателя такие способности, о которых он и не подозревал.

"Проверьте свои знания": Энциклопедия в 10 томах. Проверять свои знания гораздо интереснее, чем приобретать новые. Объемные, расширенные ответы по многим предметам облегчат накопление новых знаний.

"Энциклопедия развлечений". Эти книги развивают ум, смекалку, сообразительность, содержат интересные сведения из разных областей знаний.

"Для школьников" — помогут в учебе, расширят кругозор, дадут толчок к самостоятельному обучению.

"Деловая литература" знакомит с основными терминами и понятиями, используемыми в финансовой и хозяйственной деятельности предприятий, дает основы юридических знаний, помогает в разнообразных житейских ситуациях.

"Философское наследие". Философские книги — это путеводители по лабиринтам человеческой души, свод законов и правил философской аналитики, кладезь человеческой мудрости.

Приглашаем к сотрудничеству: авторов, литературных агентов, книготорговые организации, предпринимателей-книготорговцев

(0622) 90-80-98, 90-81-70, 35-94-69-Ф,
90-56-81, 35-95-97-Ф, 90-50-81
(0612) 69-54-98, 64-35-62-Ф
(044) 441-71-74, 441-72-80