МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
«БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

В. Л. БЫКОВ

Основы информатики
(конспект лекций)

БРЕСТ 2003

УДК 004(75)
ББК 22.183.492 : 73Я73
Б 95

Рецензенты: заведующий кафедрой Информатики и прикладной математики кандидат физико-математических наук Мадорский Виладий Меерович и доцент кафедры Информатики и прикладной математики Силаев Николай Васильевич Брестского государственного университета им. А. С. Пушкина

В. Л. Быков
Б 95 Основы информатики: конспект лекций -- Брест: БГТУ, 2003.

ISBN 985-6584-61-2

Конспект лекций разработан в соответствии с рабочей программой по дисциплине "Информатика" Брестского государственного технического университета и предназначен для студентов очной и заочной форм обучения. Излагаются общие сведения об информатике, как основы познания, и ее категориях, системах счисления, применяемых в вычислительной технике, об истории развития вычислительной техники, устройстве компьютера и его составных частей, программном обеспечении: операционные системы MS-DOS и Windows. Рассмотрены сервисные оболочки Volcov Commander и Windows Commander, программы архивации, приведены сведения об антивирусных программах. Изложены общие сведения о работе в вычислительных системах Derive, Mercury и Mathcad. Достаточно подробно рассмотрены вопросы алгоритмизации и программирования в среде QBasic.

ISBN 985-6584-61-2

УДК 004(75)
ББК 22.183.492 : 73Я73

Оглавление

Введение

Предмет "Информатика" является естественнонаучной дисциплиной для всех технических направлений и специальностей. Современный уровень развития науки и техники, который не мыслим без знания вычислительной техники, по новому поставил вопрос о подготовке специалистов в ВУЗах. Если в начале 20 века, после Октябрьской революции 1917 года, молодое правительство России поставило вопрос о ликвидации общей неграмотности, то в настоящее время, в начале 21 века, должен быть поставлен вопрос о ликвидации компьютерной неграмотности. В соответствии с общеобразовательными стандартами студенты ВУЗов должны иметь представление об информатике как особом способе познания мира, роли информатики в обеспечении экономической мощи государства, о современных достижениях в области вычислительной техники и информационных технологий; знать и уметь использовать современные средства вычислительной техники и численные методы решения инженерных и экономических задач; иметь навыки численного решения математических задач, задач математической физики, использования стандартных и технологических программ, смысловой постановки прикладных задач, алгоритмизации и программирования задач отрасли, математического моделирования объектов и оценки пределов применения полученных результатов.

Целью данной дисциплины является приобретение студентами знаний, умений и навыков по эффективному использованию современной вычислительной техники и ее программного обеспечения.

Предметом изучения в дисциплине "Информатика" в рамках учебной программы будет базовое программное обеспечение, операционная система, сервисные оболочки, языки программирования QBasic, VisualBasic, операционная система Windows, пакеты прикладных программ Word и Excel, сетевое программное обеспечение.

Объем часов, отводимых на изучение данной дисциплины различен для каждой специальности. Например, на очном отделении электронно-механического факультета по специальности 36 01 01 на изучение дисциплины отводится 270 часов, в том числе 87 часов лекций и 104 часа лабораторных занятий, на заочном факультете по этой же специальности на изучение дисциплины отводится также 270 часов, в том числе 16 лекций и 30 часов лабораторных занятий. То есть на заочном факультете изучение дисциплины студентами должно осуществляться практически самостоятельно.

Конспект лекций содержит материал по программе первого курса дисциплины "Информатика". По языку программирования Visual Basic 6.0, издано отдельное пособие: Быков В. Л. Основы программирования на языке Visual Basic 6.0. – Брест, БГТУ, 2002.

Конспект лекций предназначен для студентов технических специальностей БГТУ, может быть полезен и для студентов других специальностей, а также для всех желающих изучить основы дисциплины "Информатика".

1. Основные понятия и категории информатики

1.1. Информация - основа познания и преобразования мира

Термин "информация" происходит от латинского *informatio* - изложение, разъяснение. Под информацией понимают сведения, передаваемые людьми устным, письменным или иным способом.

Об информации можно говорить в широком и узком смысле слова.

В широком смысле слова информация - это отражение реального мира во всём его многообразии, в узком смысле слова - это любые сведения, являющиеся объектом регистрации, хранения, передачи и преобразования.

До недавнего времени об информации, как объекте научного исследования не было и речи, хотя информацией человечество пользовалось во все времена. Информация имеет ряд важных характеристик: структура, форма, измерение, ценность, достоверность и др.

Структура информации – это то, что определяет взаимосвязи между ее составными элементами. Фундаментальным свойством информации является ее *системность*. Это означает, что информация обладает в совокупности такими свойствами, какими не обладают ни один из составляющих ее элементов. Например, фраза несет больше информации, чем отдельные слова, из которых она построена.

Форма информации – форма информации определяется способом ее представления. В зависимости от этого традиционно различают символьно-текстовую, графическую, звуковую информацию. Можно также выделить и другие, нетрадиционные формы представления информации, такие как: запах, тепло, нервное раздражение кожи и др., которые современная техника в состоянии измерять и представлять в ЭВМ.

Измерение информации – информация может быть измерена. Потребность в измерении информации возникла в связи с необходимостью ее передачи, накопления и хранения.

Информация возникает в процессе взаимодействия трех объектов: источника информации, среды, через которую передается информация, и приемника информации. (Человек стоит на берегу реки и слышит плеск волн. Плеск волн – источник информации, воздух – среда в которой распространяется звук, человек - приемник информации).

Ценность информации. Сообщение, которое тем или иным способом уменьшает наше незнание о предмете или явлении, может считаться ценным. В противном случае это сообщение не представляет для нас никакой ценности. Основываясь на этом подходе, один из основоположников теории информации Клод Шеннон определил информацию как *снятую неопределенность*.

Место информации в системе познания человека отражает схема, представленная на рис. 1.1.1. Процесс усвоения знаний и применений их на практике представляет замкнутый цикл. Человек, изучая объекты окружающего мира, получает информацию, которую фиксирует на различных носителях: книгах, магнитных лентах, магнитных дисках. Собранная информация обрабатывается, анализируется, обобщается и накапливается на тех же носителях информации.

Рис.1.1.1 Цикл получения информации

Теперь эта научная информация (нас интересует прежде всего она) может быть достоянием каждого желающего. Появление и внедрение в научные учреждения компьютеров ускорило процесс поиска и использование этой информации. Сведения, полученные при использовании научной информации позволяют расширить наши знания об объекте, создавать новые методы исследования, получать новую информацию.

В одном терминологическом ряду с понятием информация стоят понятия "данные" и "знания".

Под *данными* понимают информацию, представленную в конкретных формах, которые адекватны возможным способам ее обработки.

Знания – это информация, на основании которой путем логических рассуждений могут быть получены определенные выводы.

В зависимости от области знаний различают научную, техническую, экономическую, производственную, правовую, патентную и другие виды информации. Каждый из этих видов информации имеет особую смысловую нагрузку и ценность, свои требования к точности и достоверности, преимущественные технологии обработки и формы представления и хранения информации.

В наш бурно развивающийся век информация с каждым годом приобретает все большее значение. Она оказывает влияние на все стороны человеческой деятельности: науку, экономику, политику, социальную сферу.

Информация оказывает непосредственное влияние также и на производство. Современные технологии выпуска продукции все в большей степени связаны с переработкой информации. Любая промышленная технология может быть охарактеризована объёмом требуемой информации на единицу продукции. Причём технология большей информационной ёмкости характеризуется большей эффективностью, меньшим потреблением материальных ресурсов. Происходит удивительное явление: материальные ресурсы как бы замещаются "невещественной" информацией. Происходит экономия энергии, металлов, нефти и других ресурсов за счёт процессов переработки информации. Благодаря этому информация становится товаром, ресурсом, и доступ к этому ресурсу определяет эффективность всех видов деятельности.

Интенсивное развитие всех отраслей знания вызвало колоссальный рост количества информации, информационный взрыв. Особенно ярко это стало проявляться с середины XX века. В 70-е годы объем информации удваивался каждые 5-7 лет. В 80-е годы удвоение информации происходило уже за 20 месяцев, а в 90-е годы – ежегодно. Например, в конце 80-х годов в СССР в области естественных и общественных наук ежегодно регистрировалось 1,5 млн. отечественных и зарубежных первоисточников. Найти, однако, интересующие сведения в этом потоке информации чрезвычайно сложно, возник информационный голод. Это *противоречие* века информации: рост объёма информации и трудность её получения.

Разрешение этого противоречия возможно только на пути информатизации и компьютеризации общества: создания интегрированных банков данных, сетей обработки и передачи информации, широкого внедрения компьютеров, обеспечивающих доступ к всемирной энциклопедии знаний.

***Информатизация** - это процесс, включающий комплекс взаимосвязанных и взаимообусловленных мер по обеспечению полного использования достоверных, исчерпывающих и современных знаний во всех общественно значимых сферах человеческой деятельности.* Это процесс существенного изменения роли информации как совокупности знаний и зависимостей между ними в общественной жизни. Создание индустрии информатики и превращение информационного продукта в товар приводит к глубоким социальным изменениям в обществе, меняет само общество. Оно трансформируется из индустриального в информационное.

Темпы экономического развития многих капиталистических стран (США, Япония, Англия и др.) в настоящее время определяются, главным образом, ростом производительности труда именно в сфере информатизации. По официальным оценкам, доля информационных видов деятельности в валовом национальном продукте США достигла 50% уже к концу 70-х годов. Во второй половине 70-х годов в США в создании вычислительного потенциала и организации его эффективного использования начался качественно новый этап интеграции его элементов. Этот этап характеризуется формированием многоотраслевого информационно - вычислительного комплекса, ядром которого служит индустрия переработки информации, представляющая собой особую отрасль экономики США. Основная функция этой отрасли - создание материально-технической базы для удовлетворения информационных потребностей промышленной и деловой сфер, органов государственного управления, других отраслей деятельности общества. Основу такой базы составляют ЭВМ, системы связи и передачи информации, базы данных, базы знаний, математические модели, программные средства, информационные технологии, контингент специалистов в области информатики и вычислительной техники, другие составляющие инфраструктуры информационного обслуживания.

В индустрии переработки информации США и других стран всё возрастающую роль приобретают так называемые новые ***информационные технологии** – машинизированные (инженерные) способы обработки, хранения, передачи и использования семантической информации – данных и знаний, реализуемых посредством автоматизированных информационных систем.* Информационные технологии включают два основных элемента – машинный и человеческий

(социальный), причем социальный элемент выступает главным. Новые информационные технологии имеют большое значение в решении задач информатизации. Поэтому информатизацию можно рассматривать как целенаправленную деятельность по созданию и широкомасштабному использованию во всех сферах жизни общества новых информационных технологий с целью интенсификации экономики, ускорения научно-технического прогресса, совершенствования стиля и повышения уровня жизни, развития производственных и общественных отношений.

Информационные технологии выступили новым средством превращения знаний в информационный ресурс общества. ***Информационный ресурс – это организованная совокупность документированной информации, включающей базы данных и знаний, другие массивы информации в информационных системах (библиотеках, архивах, фондах и пр.)*** В последней четверти двадцатого столетия информация, по утверждению специалистов, стала для промышленно развитых стран одним из наиболее важных национальных ресурсов. В начале нового века информационные ресурсы станут основным национальным богатством, а эффективность их промышленного использования будет определять экономическую и оборонную мощь страны в целом.

Контрольные вопросы

1. Что такое информатика?
2. Что понимают под информацией в широком смысле слова?
3. Что понимают под информацией в узком смысле слова?
4. Как влияют научно-технические достижения на объём и распространение информации?
5. Что составляет научно-техническую базу процесса информатизации общества?
6. Какое место занимает информатика в системе познания человеком окружающего мира и научных знаний?
7. Как влияет информация на уровень жизни общества?
8. Что понимается под информационными технологиями?
9. Что такое информационный ресурс общества?

1.2. Информатика

1.2.1. История развития информатики

В истории развития информатизации общества выделяют три информационных скачка:

- освоение устной речи;
- возникновение письменности и книгопечатание;
- развитие компьютерной техники и технических средств связи.

Первый информационный скачок произошел еще на заре развития человечества, когда люди путём обмена информацией об окружающем мире благодаря членораздельной речи стали получать больше сведений, чем путём наследственной передачи. В начале зарождения человеческого общества объём информации был невелик. Она, то есть информация, представляла собой сведения о природе, роде, племени. Информация эта накапливалась и передавалась

из поколения к поколению в виде устных преданий. В обществе проявилось такое качество информации, как *экономия познания*, при которой процесс познания через пробы и ошибки одного из членов сообщества служит источником знаний для остальных членов сообщества. Способность передавать информацию через пространство и время при помощи знаков присуща лишь человеку. Это качество сыграло важнейшую роль в развитии цивилизации, а в будущем оно, как утверждают специалисты, станет важнейшим.

Второй информационный скачок произошёл примерно пять тысяч лет назад с появлением различных форм письменности. Развитие книгопечатания было на этом пути хотя и эволюционным, но значительным шагом вперёд. Появилась возможность накапливать, хранить, передавать информацию. Каждое очередное техническое новшество-изобретение паровой машины, радио, телеграфа, телевидения - ускоряло процесс распространения информации, в том числе и научной.

Третий информационный скачок происходит в настоящее время в связи с появлением персональных компьютеров. Впервые стало возможным не только запоминание и хранение информации, а "накопление интеллекта" в виде профессиональных знаний и их тиражирование в виде программ для компьютеров.

Само понятие "информатика" возникло уже с появлением ЭВМ и подразумевалось вначале как наука о вычислениях. Благодаря ЭВМ появилась возможность представлять, хранить и передавать информацию в одной форме – двоичной, независимо от ее вида. Рост объёмов информации и проблемы, связанные с её получением, привели к возникновению новой отрасли науки - информатики. ***Информатика - это наука о законах и методах организации и переработки информации в естественных и искусственных системах с применением ЭВМ.***

На сегодняшний день информатика представляет собой *комплексную научно-техническую дисциплину*. Она объединяет под своим названием довольно обширный комплекс наук, каждая из которых занимается изучением одного из аспектов понятия информация. Предпринимаются интенсивные усилия ученых по сближению наук, составляющих информатику. Информатика представляет собой основу для информатизации общества и перехода его от индустриального к информационному.

Основными задачами информатики, как науки, являются:

1) разработка методов и алгоритмов автоматизированного сбора, хранения, поиска и передачи информации;

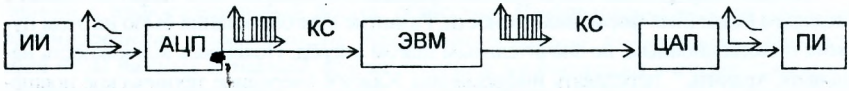
2) разработка методов и алгоритмов обработки и преобразования информации;

3) разработка технологий и электронно-вычислительной техники, позволяющих развивать первые два направления.

В развитии ЭВМ, в связи с рассматриваемым вопросом, просматриваются три этапа: вычислительный, общеинформационный, интеллектуальный. Вычислительный этап охватывает период 40-х – 60-х годов, когда доступ пользователей к ЭВМ был ограничен. Общеинформационный этап охватывает период с 60-х годов до конца 80-х годов. Наука и техника находятся сейчас на третьем этапе, этапе развития машинного интеллекта.

1. 2.2. Способы представления информации.

С практической точки зрения информация всегда представляется в виде *сообщения*. Информационное сообщение связано с источником информации (ИИ), каналами связи (КС), приемником информации (ПИ). Кроме того в канале передачи информации могут присутствовать аналого-цифровые (АЦП) и цифроаналоговые (ЦАП) преобразователи ((рис.1.2.1).



Источник информации служит для измерения параметров контролируемой среды и преобразования её к виду, удобному для передачи по каналам связи или использования в ЭВМ. Источником информации могут быть различные электронные устройства, хранящие, преобразующие и воспроизводящие информацию, а также сами ЭВМ.

Приёмником информации в одних случаях выступает ЭВМ, в других случаях - различные устройства автоматики: электромагниты, машинные усилители, пусковая аппаратура электродвигателей и др.

Сообщение от источника информации к приёмнику информации передаётся в *материально-энергетической форме* (электрический, звуковой, световой сигналы и т.д.) В качестве каналов передачи информации могут использоваться воздушная среда, проводные, кабельные или волоконно-оптические линии связи. Информационное сообщение можно представить как изменение во времени параметров материально-энергетической среды и описать его функцией от времени, например, $y = x(t)$.

В зависимости от физической природы измеряемой величины, а также способа измерения и преобразования, измеряемая величина может быть представлена в непрерывной - аналоговой форме или в виде дискретных сообщений. В ряде случаев переход от непрерывного сообщения к дискретному даёт преимущества при передаче, хранении и обработке информации.

В нестоящее время для управления технологическими процессами и техническими системами, например, изготовления полимерных волокон, управления сложными механизмами, применяются цифровые ЭВМ. Поэтому возникает объективная необходимость преобразования непрерывных параметров, характеризующих состояние технологического процесса (температуры, давления, влажности, процентного содержания отдельных компонентов и т.д.) в дискретную форму. Устройства, осуществляющие такие преобразования, называются *аналого-цифровыми преобразователями* (АЦП). Цифровые ЭВМ выдают информацию также в дискретной форме, однако для управления некоторыми устройствами (машинные усилители, магнитные усилители, преобразователи энергии, двигатели постоянного тока) необходимы непрерывные сигналы. Обратное преобразование сигнала из дискретной в цифровую форму осуществляется с помощью *цифро-аналоговых преобразователей* (ЦАП).

Для представления информации в ЭВМ используется *алфавитный способ*, основой которого является использование фиксированного конечного набора символов любой природы, называемого *алфавитом*. Символы из набора алфавита называются *буквами*, а любая конечная последовательность букв этого алфавита - *словом*. При этом не требуется, чтобы слово обязательно имело языковое смысловое значение. Примеры слов: ААВ, 10110110.

Символы алфавита при вводе в ЭВМ должны быть преобразованы в код. В цифровых ЭВМ преимущественное распространение получило двоичное кодирование, при котором символы вводимой в ЭВМ информации представляются средствами двоичного алфавита, состоящего из двух символов "0" и "1". Двоичный алфавит по числу входящих в него символов является минимальным, поэтому при двоичном кодировании алфавита, включающего большее число букв, каждой букве ставится в соответствие последовательность нескольких двоичных знаков или двоичное слово. Такие последовательности называются *кодовыми комбинациями*. Процесс получения кодовых комбинаций для представления букв одного алфавита средствами другого алфавита называется *кодированием*. Процесс обратного преобразования информации, относительно ранее выполненного кодирования, называется *декодированием*. Например, представление буквы А русского алфавита с помощью двоичных символов 11100001 есть процесс кодирования, обратный процесс получения буквы А из двоичного кода 11100001- есть декодирование. Полный набор кодовых комбинаций, соответствующий представлению всех букв одного алфавита средствами другого алфавита, называется *кодом*.

Число символов, составляющих кодовую комбинацию, называется *длиной кода* или *разрядностью кода*. Максимальное число кодовых комбинаций N при заданной разрядности n определяется выражением $N = 2^n$.

Различают коды равномерные и неравномерные. В равномерных кодах число символов во всех кодовых комбинациях одинаковое, в неравномерных - разное. Примером неравномерных кодов является азбука Морзе, где, например, буква Е имеет один короткий сигнал, а буква Ш - четыре длинных сигнала.

В вычислительной технике используются обычно равномерные коды. В IBM – совместимых ЭВМ для внутреннего представления используется ASCII код.

Для измерения объёма информации используются следующие единицы измерения:

бит - один двоичный символ;

байт - восемь двоичных символов;

слово - два байта, для 16 разрядных, или 4 байта, для 32 разрядных ЭВМ;

килобайт - 1024 бита (2^{10});

мегабайт - 1048576 байт (2^{20}) и более крупные единицы измерения – гигабайт – 2^{30} байт и терабайт – 2^{40} байт.

Контрольные вопросы

1. Назовите основные этапы развития информатики.
2. Дайте определение понятию информатика.
3. Какие основные задачи призвана решать информатика как наука.

4. Изобразите общую структурную схему передачи сообщений.

5. Что представляет собой информационное сообщение, в каком виде оно передаётся?

6. Опишите принцип работы системы передачи информации.

7. Что такое код? Какое количество кодовых комбинаций необходимо для кодирования всех символов русского алфавита, белорусского алфавита?

1.2.3. Форматы представления информации

При работе с ЭВМ приходится иметь дело с различными видами информации: числовой, буквенной, графической, звуковой. Для представления этой информации разработаны определённые форматы. Форматы представления информации определяются *разрядной сеткой* ЭВМ.

Под разрядной сеткой понимают совокупность двоичных разрядов, используемых для хранения и обработки машинных слов.

В ЭВМ число этих разрядов фиксировано и кратно восьми: 8, 16, 32 и т.д.

Форматы представления числовых данных.

Числа могут быть представлены в двух формах: естественной и нормальной.

При естественном представлении чисел в ЭВМ, например, таких чисел как: 12560; 0,003572; 4,89760, - устанавливается длина разрядной сетки, а также длина целой и дробной частей. При этом распределение разрядов между целой и дробными частями не изменяется и остаётся постоянным независимо от величины числа. Такое представление чисел называется представлением чисел в форме с фиксированной запятой. В современных ЭВМ эта форма используется преимущественно для представления целых чисел (рис. 1.2.2.).

Рис.1.2.2. Формат представления числа с фиксированной точкой

При таком представлении целых чисел один разряд (старший) отводится под знак числа, остальные пятнадцать разрядов - под поле числа. Диапазон изменения чисел от $-(2^n-1)$ до $+(2^n-1)$.

Представлением чисел в нормальной форме называют представление числа в форме с плавающей запятой. Общий вид числа: $A_н = m_A * q^{P_A}$, где m_A - мантисса числа A, P_A - характеристика числа A, q - основание системы счисления. Например, в числе $0{,}14518 \cdot 10^4$ число 0,14518 - мантисса числа, 4 - характеристика или порядок числа, 10 - основание системы счисления. Мантисса числа во избежание неоднозначности представления чисел должна находится в пределах

$$q^{-1} \le |m_A| < 1$$

В шестнадцатиразрядном слове десять разрядов отводится для представления мантиссы, в т.ч. один разряд под знак мантиссы и девять - под поле мантиссы, а шесть разрядов отводится для записи порядка чисел, в т.ч. один разряд под знак порядка и пять разрядов под поле порядка (рис. 1.2.3.).

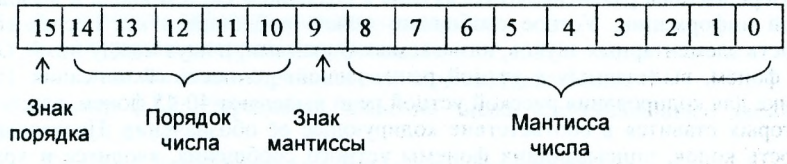


Рис.1.2.3. Формат представления числа с плавающей точкой

Форматы представления букв

Для представления букв, цифр, знаков арифметических и логических операций, знаков сравнения и специальных знаков используется один байт. При этом можно закодировать 256 символов.

Особенности представления графической информации

В режиме отображения графической информации экран видеомонитора представляет собой поле точек 320 x 200, 640 x 200, 640 x 480, 800 x 600, 1280 x 1024, возможны и другие режимы Для каждой точки графического изображения необходимо хранить значения уровней яркости, а при цветном изображении - цвет и оттенок. Современные дисплеи позволяют получать 16, 64, 256 и более различных цветов. В каждый момент времени необходимо хранить информацию о всех точках экрана, для чего требуется большой объём памяти. Кроме того в ЭВМ могут храниться стандартные графические символы, сформированные самим пользователем. Для этой цели ЭВМ имеет специально отведённую область памяти.

Изображение, выдаваемое на экран дисплея в закодированном виде, хранится в специально отведённой для этого области памяти компьютера, называемой памятью регенерации изображения. Данные из памяти регенерации периодически считываются, преобразуются в видеосигнал и отображаются на экране. Изображение на экране меняется с определённей частотой, обеспечивающей ему чёткость и стабильность (на современных мониторах частота обновления может изменяться в пределах от 60 до 100 Гц). Электронный луч обегает экран построчно с верхнего левого угла в нижний правый угол. Время возвращения луча в верхний левый угол (когда он гасится) используется для изменения информации в памяти регенерации.

Память регенерации может быть разделена на страницы, каждая из которых содержит информацию об изображении полного экрана. Многостраничная организация памяти регенерации позволяет удобно реализовать эффекты движения графических изображений.

Особенности представления звуковой информации

Простейшим примером, поясняющим особенности кодирования звуковой информации, может служить нотная запись музыкального произведения. Обозначение нот, их длительности, пауз, знаков усиления и снижения громкости звука, музыкального ударения и другие знаки, составляющие набор нотной аз-

буки, образуют алфавит, с помощью которого можно представить в закодированном виде музыкальную мелодию. Последовательность символов, кодирующих музыкальную мелодию, хранится и обрабатывается в ЭВМ как обычная алфавитно-цифровая информация. Это позволяет использовать ЭВМ не только для анализа, но и для создания музыкальных произведений.

Перспективы развития ЭВМ связаны с созданием систем ввода и вывода речевой информации. Устное сообщение можно представить как последовательность элементарных звуков, называемых фонемами, и пауз между ними. От числа фонем, выделяемых в устной речи, зависит точность её описания. На практике для кодирования русской устной речи выделяют 40-45 фонем, каждой из которых ставится в соответствие кодирующее её обозначение. Последовательность кодов, описывающих фонемы устного сообщения, вводится и хранится в памяти ЭВМ и при необходимости выводится из неё через специальные устройства, называемые синтезаторами речи.

В настоящее время разработаны устройства, позволяющие переводить письменный текст в соответствующее фонемное представление, что позволяет воспроизводить этот текст на экране видеомонитора или через синтезаторы речи.

Весьма перспективным является создание средств общения человека с ЭВМ посредством голоса. Такие системы находятся в стадии развития.

Контрольные вопросы

1. Что такое разрядная сетка ЭВМ? Чем она определяется?
2. Какие форматы используются для представления: а) целых; б) вещественных чисел?
3. Сколько кодовых комбинаций можно представить с помощью одного байта?
4. В чём состоит особенность представления графической информации?
5. Что такое фонема? Сколько фонем необходимо для представления звуков русского алфавита?
6. Какие устройства используются для воспроизведения речи?

1.2.4. Системы счисления

Под системой счисления понимают совокупность приёмов и правил для записи чисел цифровыми знаками.

Различают позиционные и непозиционные системы счисления.

Примером *непозиционной* системы счисления является римская система использующая набор символов I, V, X, C, L, D. Эта система неэффективная, т.к. чем больше число, тем длиннее получается запись. Кроме того, значение некоторых символов меняется в зависимости от положения его в числе. Так в числах LX и XL символ X принимает соответственно значения +10 и -10.

В *позиционной* системе счисления значение цифры определяется положением в числе: один и тот же знак принимает различные значения. Примером позиционной системы счисления является десятичная система счисления, известная всем со школьной скамьи. Например, в числе 222 первая цифра слова означает число двести, вторая - двадцать, третья - два. Вес числа возрастает в 10 раз при движении справа налево. Рассмотрим вещественное число 135,28:

$$135{,}28=100+30+5+0{,}2+0{,}08=1\text{x}10^2+3\text{x}10^1+5\text{x}10^0+2\text{x}10^{-1}+8\text{x}10^{-2}.$$

Приведённый пример записи вещественного числа в десятичной системе счисления позволяет установить общую форму записи чисел в позиционной системе счисления. Для произвольного числа А можно записать:

$$A(q)=a_{n-1}q^{n-1}+a_{n-2}q^{n-2}+\ldots+a_iq^i+\ldots+a_1q^1+a_0q^0+a_{-1}q^{-1}+\ldots+a_{-m}q^{-m},$$

где a_i - символы алфавита системы счисления;

q - основание системы счисления;

n, m - число целых и дробных разрядов соответственно.

Пример 1.2.1. Определите значение числа, представленного двоичным кодом 101101.011_2.

Решение. Число содержит целую и дробную части n=6, m=3

$$A_{(2)}=1\cdot2^5+0\cdot2^4+1\cdot2^3+1\cdot2^2+0\cdot2^1+1\cdot2^0+0\cdot2^{-1}+1\cdot2^{-2}+1\cdot2^{-3}=$$

$$1\cdot32+0\cdot16+1\cdot8+1\cdot4+0\cdot2+1\cdot1+0\cdot1/2+1\cdot1/4+1\cdot1/8=32+8+4+1+1/4+1/8=45{,}375$$

Любая позиционная система счисления характеризуется *основанием* (базисом). Основание позиционной системы счисления - это число знаков или символов, для изображения цифр в данной системе.

В вычислительной технике нашли применение четыре позиционные системы счисления, приведённые в табл. 1.2.1.

Таблица 1.2.1.

Позиционные системы счисления

Наименование	***Алфавит***	***Основание системы счисления***
Двоичная	0,1	2
Десятичная	0,1,2,3,4,5,6,7,8,9	10
Восьмеричная	0,1,2,3,4,5,6,7	8
Шестнадцатеричная	0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F	16
Двоично-десятичная	0,1,2,3,4,5,6,7,8,9	10

Если задано число $A_{max}(q)$, можно определить требуемое число разрядов для его представления - *n*:

$$n=\log_q(A_{max}(q)+1)$$

И наоборот, если известна длина разрядной сетки *n*, то можно определить максимальное число $A_{max}(q)$, которое можно представить с использованием данной разрядной сетки $A_{max}(q)=q^n-1$.

Например, с помощью одного байта (восьми разрядов) можно представить число 255:

$$2^8 - 1 = 256 - 1 = 255,$$

а с помощью 15 разрядов - число 32767:

$$2^{15} - 1 = 32768 - 1 = 32767.$$

Интервал числовой оси, заключённый между максимальным и минимальным числами, называют *динамическим диапазоном*.

Вычислительная машина, как известно, может оперировать только с двоичными знаками, поэтому десятичные числа она предварительно переводит в

двоичную систему счисления. С клавиатуры числа могут быть введены также в восьмеричном или шестнадцатеричном кодах. При выводе числовой информации на экран ЭВМ производит обратные преобразования, т.е. переводит числа из двоичной системы счисления в десятичную. Указанные преобразования осуществляются в соответствии с определёнными алгоритмами. Правила перевода приведены в табл. 1.2.2 и 1.2.3 .

Для перевода двоичных чисел в восьмеричные двоичное число делится на триады, каждая из которых переводится в соответствующее восьмеричное число:

$$110\ 111\ 001\ 101\ 010_2 = 671352_8$$

При обратном переводе каждое число восьмеричного кода заменяется тремя двоичными знаками:

$$751254_8 = 111\ 101\ 001\ 010\ 101\ 100_2$$

Для перевода двоичных чисел в шестнадцатеричный или двоично-десятичный код двоичное число делится на тэтрады, каждая из которых переводится в соответствующий символ шестнадцатеричного или двоично-десятичного числа:

$$1100\ 0111\ 1011\ 0101_2 = C7B5_{16},$$
$$1001\ 0110\ 0011\ 0100_2 = 9634_{2\text{-}10}.$$

При обратном переводе шестнадцатеричных и двоично-десятичных чисел в двоичные каждый символ заменяется кодом из четырёх двоичных символов, соответствующих коду шестнадцатеричного или двоично-десятичного числа:

$$AE3C_{16} = 1010\ 1110\ 0011\ 1100_2,$$
$$7954_{2\text{-}10} = 0111\ 1001\ 0101\ 0100_2.$$

Арифметические операции с двоичными кодами

Арифметические операции с двоичными кодами выполняются в соответствии со следующими правилами:

сложение	вычитание	умножение
0 + 0 = 0	0 – 0 = 0	0 x 0 = 0
1 + 0 = 1	1 – 0 = 1	1 x 0 = 0
0 + 1 = 1	1 – 1 = 0	0 x 1= 0
1+ 1 = 0⇒ 1	1⇒ 0 – 1 = 1	1 x 1 = 1
перенос в старший разряд	заём в старшем разряде	

При выполнении операции вычитания с заёмом в старшем разряде следует запомнить, что единица старшего разряда равна двум единицам младшего разряда, например:

$$\begin{array}{r} {\scriptstyle +1+1} \\ 1\ 1\ 0 \\ -1 \\ \hline 1\ 0\ 1 \end{array}$$

Таблица 1.2.2.

Перевод чисел из десятичной системы счисления в двоичную систему счисления и обратно

Перевод целых чисел	*Перевод дробных чисел*
Правило перевода. *Разделить число на основание системы счисления. Остаток от деления записать в виде кода двоичного числа. Целую часть числа снова разделить на основание системы счисления. Вверху младший разряд, внизу – старший разряд.*	**Правило перевода.** *Умножить число на основание системы счисления. Целая часть полученного числа образует код двоичного числа, а остаток используется для последующего умножения. Вверху старший разряд, внизу – младший. Умножение продолжается до достижения требуемой точности.*
29 : 2 = 14 | 1 - мл. разряд 14 : 2 = 7 | 0 7 : 2 = 3 | 1 3 : 2 = 1 | 1 1 : 2 = 0 | 1 – ст. разряд	0.35 x 2 = 0.70 ↑ 0 - ст. разряд 0.70 x 2 = 1.4 | 1 0.4 x 2 = 0.8 | 0 0.8 x 2 = 1.6 | 1 0.6 x 2 = 1.2 | 1 0.2 x 2 = 0.4 | 0 - мл. разряд
Код: 11101	Код числа .01011

Таблица 1.2.3

Перевод чисел из двоичной системы счисления в десятичную

Перевод целых чисел	*Перевод дробных чисел*
Правило перевода. *Умножить старший разряд на основание системы счисления и прибавить к полученному результату следующий разряд. Полученное число снова умножить на основание системы счисления и прибавить следующий разряд.*	**Правило перевода**. *Разделить младший разряд на основание системы счисления. Прибавить к полученному числу следующий разряд и разделить результат на основание системы счисления. Прибавить к полученному результату текущий разряд и снова разделить полученный результат на основание системы счисления.*
1 ↑ =1 - ст. разряд 1 x 2 + 1 | = 3 3 x 2 + 1 | = 7 7 x 2 + 0 | = 14 14 x 2 + 1 | = 29 - мл. разряд	1 | : 2 = 0.5 - мл. разряд (0.5 + 1) | : 2 = 0.75 (0.75 + 0) | : 2 = 0.375 (0.375 + 1) | : 2 = 0.6875 (0.6875 + 0) ↓ : 2 = 0.34375 - ст. разряд
Перевод десятичных чисел в восьмеричную и шестнадцатеричную системы счисления и обратно осуществляется по приведенным выше правилам.	

Пример 1.2.1: Сложение и вычитание двоичных чисел.

101101	1001101
+ 10110	- 11011
1000011	0110010

Операцию вычитания заменяют в ЭВМ сложением дополнительных кодов этих же чисел. Дополнительный код отрицательного числа образуется путём прибавления единицы к младшему разряду обратного кода этого числа:

прямой код	- 101101
обратный код	010010
	+ 1
дополнительный код	010011

Дополнительный код положительного числа равен этому же положительному числу.

При сложении чисел в дополнительных кодах используется правило: сумма дополнительных кодов чисел равна дополнительному коду результата. При этом необходимо учитывать перенос в знаковый (к+1) разряд; перенос в старший (к+2) разряд не учитывается. Если в знаковом разряде единица, то число отрицательное. Если результат сложения дополнительных кодов положительный, то результат сложения чисел будет равен дополнительному коду результата. Если результат сложения дополнительных кодов отрицательный, то результат получают следующим образом:

1) вычесть единицу из дополнительного кода результата;
2) найти код обратный полученному.

Это и будет искомый результат.

Примеры сложения чисел в дополнительных кодах:

Пример 1.2.2: 7 + (-5) = 2

↓ Знаковый разряд

7	0	0	0	0	0	1	1	1
+								
-5	1	1	1	1	1	0	1	1
2	0	0	0	0	0	0	1	0

← код числа 7

← дополнительный код числа -5

← дополнительный код результата (положительный) равен результату

Пример 1.2.3: (-7) + (-5) = -12

↓ Знаковый разряд

-7	1	1	1	1	1	0	0	1
+								
-5	1	1	1	1	1	0	1	1
-12	1	1	1	1	0	1	0	0

← дополнительный код числа -7

← дополнительный код числа -5

← дополнительный код результата (отрицательный)

Преобразование дополнительного кода результата

Вычитаем единицу из дополнительного кода результата и получаем обратный код, из которого затем получаем прямой код (знаковый разряд не преобразуется).

↓Знаковый разряд

-12	1	1	1	1	0	1	0	0
-								
	0	0	0	0	0	0	0	1
	1	1	1	1	0	0	1	1
-12	1	0	0	0	1	1	0	0

← дополнительный код результата

← код числа 1

← обратный код результата

← результат в прямом коде

Операции умножения и деления выполняются в ЭВМ по определенным алгоритмам, сходными с алгоритмами умножения и деления в арифметике. Операции умножения представляются последовательностью операций умножения, сдвига и сложения.

Пример 1.2.4: 22 x 7 = 154

```
   10110
 x   111
   10110
 + 10110
  1000010
+ 10110
  10011010
```

Операция деления в ЭВМ представляется последовательностью операций сравнения, умножения, вычитания и сдвига.

Пример 1.2.5: 154 : 22 = 7

```
10011010 | 10110
 10110     0111
 0100001
  10110
  0010110
   10110
   00000
```

Контрольные вопросы

1. Что такое основание системы счисления?
2. Запишите число 476,15 в общей форме записи чисел в позиционной системе счисления.
3. Составьте таблицу двоичных кодов шестнадцатеричной системы счисления, используя общую форму записи чисел в позиционной системе счисления.
4. Переведите в восьмеричную систему счисления код 0111001111001_2.
5. Переведите в шестнадцатеричную систему счисления код $1011\ 0110\ 1111\ 1010_2$.
6. Переведите код $1001\ 1000\ 0101\ 0010_{2\text{-}10}$ из двоично-десятичной системы счисления в десятичную систему счисления.
7. Переведите число $AC37_{16}$ в двоичную систему счисления.
8. Переведите в двоичную систему счисления число 375_{10}. Сделайте проверку.
9. Переведите в двоичную систему счисления число $0{,}761_{10}$ с точностью до 0,01. Сделайте проверку.
10. Переведите из двоичной системы счисления в десятичную систему счисления следующие коды: 10110110_2, 0.10110011_2

2. История и перспективы развития вычислительной техники

Электронно-вычислительная машина (ЭВМ) представляет собой устройство, предназначенное для выполнения вычислительных и логических операций, а также обработки информации.

В своём развитии ЭВМ прошли достаточно большой путь от замысла до воплощения в реальные машины. В развитии вычислительной техники принято выделять ряд этапов:

- ручной (начало не установлено);
- механический (с середины 17 века);
- электромеханический (с 90–х годов 19 века);
- электронный (с 40-х годов 20 века).

Ручной этап

Человек всегда стремился увеличить скорость и точность своих вычислений. К этому подвигали потребности торговли, мореплавания, развития науки и техники. Процессы развития вычислительной техники и науки шли всегда рука об руку, подгоняя друг друга. Ещё задолго до первого тысячелетия до новой эры в Средиземноморье и на Востоке были широко распространены простейшие приспособления для вычислений: дощечки, покрытые слоем песка, с камешками или бусинками. Такие дощечки назывались "*абак*". Многие годы они совершенствовались, улучшалась техника их использования. Дальние родственники "абак" - конторские счёты дожили кое-где и до сегодняшнего дня. Однако, никаких вычислительных устройств до конца XVI столетия создано не было.

Механический этап

Около 1594г. *Джон Непер* изобрёл логарифмы, и появились вычислительные приспособления известные как "палочка Неппера". Несколько лет спустя *Уильям Отред* усовершенствовал вычислительное устройство, совместив две логарифмические шкалы. Таким образом появилась логарифмическая линейка.

В 1623 году немецкий ученый *Вильгельм Шиккард* построил первую механическую вычислительную машину. Она была выполнена в единственном экземпляре и позволяла выполнять арифметические операции. Этот факт долгое время не был известен.

В 1642 году *Блез Паскаль* (французский математик, физик, философ, писатель), независимо от В. Шиккарда, изобрёл механическую счётную машину, выполняющую сложение, а в 1674 году *Готфрид Лейбниц* расширил возможности машины Паскаля, добавив операцию умножения, деления и извлечения квадратного корня (представители этого семейства - арифмометры).

Ни одно из этих устройств не могло работать без непосредственного и непрерывного участия человека. Но появление этих машин поставило вопрос о

возможности создания полностью автоматической вычислительной машины, способной выполнять разнообразные вычисления без вмешательства человека.

В 1834 году *Чарльз Бэббидж* первым выдвинул подробный проект такой машины, названной им аналитической. Он так и не построил своей машины, так как в то время невозможно было достичь требуемой точности изготовления её узлов. Но его идеи заложили фундамент, на котором со временем были построены ЭВМ. Чарльз Бэббидж предложил управлять своей машиной с помощью перфорированных карт, содержащих коды команд, подобно тому, как использовались перфокарты в ткацких станках Жаккарда, изобретённых примерно в 1800 году для управления изготовлением рисунка на ткани. Этот принцип ввода информации использовался до 90 годов прошедшего столетия на больших ЭВМ в пакетном режиме обработки информации.

В этот период были разработаны и некоторые теоретические основы для развития будущих поколений вычислительной техники. *Ада Лавлейс* – дочь известного поэта Джорджа Байрона составила программу вычисления чисел Бернулли, разработала основные принципы программирования, которые актуальны и по сей день, а также ввела ряд терминов, такие как "цикл", "рабочие ячейки". Английский математик Джордж Буль (1815-1864) заложил теоретические основы современных вычислительных машин. Он разработал алгебру логики, ввел в обиход логические операторы И, ИЛИ, НЕ.

Электромеханический этап

Этому этапу предшествовали такие выдающиеся открытия в области физики как открытие электрона, изобретение электромагнита, электродвигателя. Он оказался достаточно коротким.

В 1888 году *Герман Холлерит* сконструировал первую электромеханическую машину для сортировки и подсчета перфокарт. Эта машина, названная табулятором, содержала реле, счетчики, сортировочный ящик. Изобретение Г. Холлерита было использовано при подведении итогов переписи населения в США.

В 1936 году немецкий инженер *Конрад Цузе* начал конструировать вычислительный аппарат, работающий в двоичной системе счисления. В 1941 году он сумел построить действующую модель такого устройства (Zuse 3), которая состояла из 600 реле счетного устройства и 2000 реле устройства памяти.

В 1943-44 годах в Англии было разработано полностью автоматическое устройство Колосс II, предназначенное для дешифровки военных сообщений. Еще одна полностью автоматическая вычислительная машина была изобретена и построена *Говардом Эйкеном*, профессором Гарвардского университета (США), при участии группы инженеров фирмы IBM. Эта машина называлась *Марк 1*, состояла из 750 тысяч компонентов, на операцию умножения затрачивала около 4 секунд.

Электронный этап

Новые возможности по созданию вычислительных машин открылись с появлением электронных ламп и последующим бурным развитием электроники. Это новый период развития вычислительной техники. Он делится на этапы, непосредственно связанные с уровнем развития элементной базы электронной техники, конструктивно-технологическим исполнением, логической организацией, математическим обеспечением, удобством общения человека с машиной. Смена поколений ЭВМ происходила революционно, ей сопутствовало изменение технико-экономических показателей этих машин: быстродействие, надёжность, потребляемая мощность, стоимость, габариты. Изменение периферийного оборудования и программного обеспечения шло в основном эволюционным путём. Появлению первых ЭВМ предшествовали такие фундаментальные изобретения как изобретение электронной лампы (1879 г.). Электронная лампа - диод имеет два электрода: анод и катод, помещенные в стеклянную колбу, в которой создан глубокий вакуум (рис.2.1). При подаче на электроды напряжения в ней протекает ток, вызванный движением электронов. Изобретение триг-

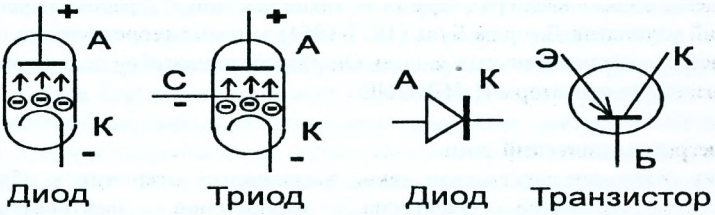


Рис.2.1. Основные элементы электронных схем

гера (1913 г.). Триггер, в отличие от двухэлектродной лампы, имеет еще один электрод – сетку. Благодаря наличию этого электрода появилась возможность управлять потоком электронов в лампе и создавать на их основе элементы памяти.

Принято выделять пять этапов в развитии электронной вычислительной техники, связанных с развитием элементной базы[1] (табл. 2.1.):

- ЭВМ на электронных лампах (1945 – 1956);
- ЭВМ на транзисторах и ферромагнитных ячейках памяти (1956 -1964);
- ЭВМ на интегральных элементах малой плотности (1964-1971);
- ЭВМ на микропроцессорных элементах (1971 –1979)
- ЭВМ на сверхбольших ИС (1979 – по настоящее время).

Работы по созданию отдельных элементов и узлов ЭВМ были начаты в 1937 г. в США *Дж. Атанасовым*. Им были запатентованы первые электронные схемы отдельных узлов ЭВМ. В 1942 г. им совместно с *К. Берри* была построена электронная машина *ABC*.

Первое поколение ЭВМ (с 1945 года)

Первая ЭВМ полностью на электронных лампах была названа ENIAC (ЭНИАК - электронный числовой интегратор и вычислитель). Она была изобретена Эк-

[1] В разных источниках приводятся различные даты начала и конца соответствующего периода

кертом и Маучли и создана в США в 1946 году. Эта ЭВМ содержала 18000 электронных ламп, и была в 1000 раз более быстродействующей, чем первая: за 1 секунду выполняла 5000 операций сложения или 360 операций умножения. Предназначалась ЭВМ для выполнения объёмных научно-технических расчётов.

В СССР электронно-вычислительные машины на электронных лампах были созданы под руководством академика С. А. Лебедева (МЭСМ и БЭСМ). МЭСМ (малая электронная счётная машина), созданная в 1951 году, сыграла важную роль в подготовке первых в стране программистов, инженеров и конструкторов ЭВМ, интенсифицировала разработку электронных элементов для применения в ЭВМ. БЭСМ (большая электронная счётная машина) была создана в 1952 г. и была в то время самой быстродействующей ЭВМ (8000 операций в секунду). Она открыла серию машин, получивших распространение в СССР. В середине 50-х годов в нашей стране появились ЭВМ серий "Стрела", "Урал", а в 60-х годах - "Промень", "Мир", "Минск", "Раздан". Эти машины могли справиться с широким кругом математических и логических задач.

Работать на этих ЭВМ могли, по-прежнему, лишь "избранные" - узкий круг специалистов-профессионалов.

Второе поколение ЭВМ (с 1956 года)

Второе поколение ЭВМ обязано своим появлением транзисторам, изобретённым в 1948 году. Транзисторы полностью заменили в качестве активных элементов электронные лампы. В отличие от ламповых, ЭВМ на транзисторах отличались большим быстродействием, ёмкостью оперативной памяти, надёжностью, меньшим потреблением электроэнергии, значительно лучшими массогабаритными характеристиками. Большой прогресс вызвало применение печатного монтажа. Повысилась надёжность и быстродействие электромеханических устройств ввода-вывода. Особенностью ЭВМ второго поколения стала их дифференциация по применению. Появились машины для решения научно-технических, экономических задач, управления производственными процессами и объектами (управляющие машины). Возникло новое понятие "машина для обработки данных". В отличие от ЭВМ для научно-технических расчётов эта машина обладала свойствами хранения (накопления, запоминания) больших объёмов информации, тогда как процесс обработки (вычислительные операции) отступал на второй план.

Появление быстродействующих устройств ввода (способных пропускать до 1000 перфокарт в минуту), алфавитно-цифровых печатающих устройств (АЦПУ), графопостроителей дало возможность гибко менять форму выдачи результатов, например, печатать данные в виде таблиц, оформлять в виде графиков. Всё это существенно облегчало обработку результатов, повысило производительность человеческого труда. Наряду с техническим совершенствованием развиваются методы и приёмы программирования вычислений, высшей ступенью которых явилось автоматическое программирование, требующее минимальной затраты труда математиков-программистов. Большое развитие получают алгоритмические языки, существенно упрощающие процесс подготовки задач к решению на ЭВМ (АЛГОЛ, ФОРТРАН).

Таблица 2.1.

Основные этапы развития электронной вычислительной техники

Поколение ЭВМ	*Предшествующие научные открытия*	*Год начала этапа*	*Элементная база*	*Назначение или тип*	*Новые свойства*	*Программное обеспечение*
1	1879 – изобретена электронная лампа, 1913 – триггер.	1945	Электронные лампы	Инженерно-технические расчеты	Программное управление, машинный язык,	Операционная система практически отсутствовала. 1949 г. – создан первый язык программирования Short Code. Языки программирования высокого уровня: Фортран - 1957, Лисп – 1956, Кобол – 1959, АЛГОЛ -1960.
2	1947 – изобретен транзистор	1956	Полупроводниковые приборы,	Обработка данных, управление техническими объектами	Языки программирования, широкая периферия	Операционные системы для работы с накопителями на магнитных барабанах и магнитных лентах. Новые языки программирования: Бейсик – 1964, ПЛ/1 – 1964.
3	1956 – изобретен жесткий диск, 1958 – 1959 – ИС.	1964	Интегральные микросхемы (ИС)	СуперЭВМ, малые ЭВМ, настольные ЭВМ	Программная совместимость, модульный принцип организации технического и программного обеспечения	Языки программирования: Паскаль – 1967, Симула –1967, ЛОГО – 1968, 1968 – текстовый процессор,
4	1967– идея МП на одном кристалле	1971	Микропроцессоры (МП)	Персональные, профессиональные ЭВМ	Децентрализация вычислений	Операционная система UNIX, системы программирования СИ – 1972, ПРОЛОГ – 1978, 1973 – операционная система для ПК CP/M.
5		1979	Сверхбольшие интегральные схемы	Экспертные системы	Искусственный интеллект	1981 - операционная система – MS-DOS, языки программирования Ада –1979, СИ ++ - 1980, HTML – 1989, ДЕЛФИ – 1995, Ява – 1995
			Задача разработки ЭВМ пятого поколения сформулирована в 1979 году в Японии. Ведутся разработки. Имеются элементы с биологическими принципами обработки информации. Проводятся научные исследования.			

В период развития и совершенствования машин второго поколения наряду с однопрограммными ЭВМ появляются многопрограммные ЭВМ. В отличие от однопрограммных в многопрограммных ЭВМ стала возможной одновременная реализация нескольких программ за счёт организации параллельной работы основных устройств машины.

Общение с ЭВМ стало несколько проще. Однако, по-прежнему в непосредственный контакт с машиной вступал узкий круг специалистов: операторов-программистов, инженеров по эксплуатации ЭВМ.

Яркими представителями отечественных ЭВМ 2-го поколения являлись "Минск-32", "Урал-16". Они имели быстродействие порядка 25000 и 100000 операций в секунду. Их оперативная память хранила соответственно 65000 и 500000 чисел. ЭВМ "Минск-32" могла работать со 136 внешними устройствами, а управлял ею один оператор с помощью устройства наподобие пишущей машинки.

Ещё более совершенной была БЭСМ-6 (выпуск 1967г.). Её быстродействие составляло 1 млн. опер./сек. Это была самая быстродействующая ЭВМ второго поколения. Фирма IBM достигла этих показателей практически десятилетие спустя. Оперативная память позволяла хранить 128000 чисел, а промежуточная на магнитном барабане - 512000 чисел, кроме того каждый из 32 подключаемых к ЭВМ магнитофонов обеспечивал хранение на магнитной ленте до миллиона машинных слов (5000 страниц текста). БЭСМ-6 отличает не только то, что она была одной из самых лучших машин второго поколения, но и удивительная "живучесть", обеспечившая её эксплуатацию до 90-х годов двадцатого столетия.

К концу 60-х годов стало ясно, что для повышения эффективности использования ЭВМ при обработке данных и управлении необходимо создавать модели ЭВМ разной производительности, но одинаковые по своей организации и обладающие программной совместимостью. Последнее означает возможность использовать запас программ, написанных для одной ЭВМ, на машинах других моделей за счёт чего снижаются затраты на обработку информации.

Третье поколение ЭВМ (с 1964 года)

Начало этому этапу положили принцип программной совместимости и технология интегральных схем. Для машин третьего поколения характерно не только улучшение габаритно-стоимостных показателей, но и модульный принцип организации технических и программных средств, обеспечивающий возможность составлять приспособленную для соответствующего конкретного назначения конфигурацию ЭВМ. Машины 3-го поколения обрабатывали не только числа, но и слова, тексты, т.е. оперировали буквенно-цифровой информацией. В машинах этого поколения значительно расширился набор различных электромеханических устройств ввода-вывода информации. Развитие этих устройств носит эволюционный характер: их характеристики улучшаются гораздо медленнее, чем характеристики электронного оборудования. Изменилась и форма общения человека с машиной. Пользователи получили доступ к ЭВМ через абонентские пункты. Математическое обеспечение машин 3-го поколения

получило дальнейшее развитие, особенно это касается операционных систем (ОС). Развитие ОС многопрограммных машин, снабжённых периферийными устройствами ввода-вывода с автономными пультами абонентов, обеспечивали управление работой ЭВМ в различных режимах: пакетной обработки, разделения времени, диалоговый режим (запрос-ответ).

Начало создания машин третьего поколения положила фирма IBM (США), приступившая в 1966 году к выпуску машин серии IBM-360. Выпуск машин данного класса, совместимых с IBM, в рамках единой системы ЭВМ (ЕС ЭВМ) в странах членах СЭВ (Болгария, Венгрия, ГДР, Куба, Польша, Румыния, СССР, Чехословакия) начался в 1972 году.

В ЕС ЭВМ были приняты единые стандарты на технические характеристики всех устройств и узлов, на системы кодов, операций, средств программирования. Все модели ЕС ЭВМ имели общий состав периферийного оборудования, обеспечивающего ввод-вывод информации. В них была предусмотрена возможность связи с абонентами по телефонно-телеграфным линиям связи с использованием терминальных пультов, включающих устройства алфавитно-цифрового и графического отображения данных на экранах электронно-лучевых трубок. Каждая модель ЕС ЭВМ имела свой собственный процессор, являющийся как бы ядром этой модели. Весь ряд таких моделей строился в порядке возрастания их быстродействия, от нескольких тысяч (ЕС 1010, Венгрия) до миллионов (ЕС 1065, СССР) операций в сек.

В странах СЭВ в этот период было создано два семейства ЕС ЭВМ разной производительности:

"Ряд-1: ЕС1010, ЕС1020, ЕС1022, ЕС1030, ЕС1040, ЕС1050;

"Ряд-2": ЕС1035, ЕС1045, ЕС1065.

На этом же этапе большее развитие получили управляющие ЭВМ.

При этом появились новые понятия: малые ЭВМ, малые управляющие ЭВМ, мини-ЭВМ. В 1974 году страны члены СЭВ объединили свои усилия в области создания семейства малых ЭВМ (СМ ЭВМ), предназначенных для использования в информационно-измерительных и управляющих системах. С появлением малых ЭВМ возникло ещё одно направление использования вычислительной техники: децентрализованная обработка данных и использование ЭВМ в непосредственной близости от рабочих мест (настольные ЭВМ). На этом же этапе зародились супер-ЭВМ, целевой установкой при разработке которых было и остаётся достижение максимальной производительности вычислительных процессов (несколько сотен миллионов операций в секунду). Их возникновение определено необходимостью решения научно-технических задач, например, современных задач аэродинамики и ядерной физики, предполагающих выполнение значительного числа операций (для указанного примера не менее 10^{13}) за ограниченный промежуток времени. Очевидно, что суперЭВМ весьма сложны и дороги, а поэтому в настоящее время насчитывается несколько сотен таких машин во всём мире.

Четвёртое поколение ЭВМ (с 1971 года)

Четвёртое поколение ЭВМ служит еще одним примером перехода коли-

чества в качество. Степень интеграции электронных схем повысилась настолько, что стало возможным сосредоточить значительное число функциональных устройств в одной большой интегральной схеме (БИС) и таким образом изготовить по этой технологии большие блоки или всю ЭВМ вцелом. Вначале появились сложные арифметические устройства и полупроводниковые запоминающие устройства; позднее появились микропроцессоры. Появление БИС создало предпосылки для качественного изменения вычислительной техники. Их применение привело к новым представлениям о функциональных возможностях элементов и узлов ЭВМ, децентрализации вычислительной мощности и встраивания вычислительных средств в оборудование и приборы.

Продолжая традиции добрососедского сотрудничества, в рамках СЭВ в этот период создаётся два новых семейства ЕС ЭВМ "Ряд-3" и "Ряд-4":

"Ряд-3": ЕС 1036, ЕС 1046, ЕС 1066;

"Ряд-4": ЕС 1037, ЕС 1077, ЕС 1087, ЕС 1191, ЕС 1766.

Быстродействие семейства ЕС ЭВМ "Ряд-3" находилось в диапазоне от 0,4 до 10 млн. операций в секунду. Последние модели ЕС ЭВМ "Ряд-4" достигли быстродействия более 100 млн. операций в секунду.

Начинается бурный рост числа ЭВМ малых размеров. Появляются одноплатные и многоплатные ЭВМ, встраиваемые и настольные ЭВМ. Резко возрастает быстродействие и объём памяти, габариты и вес напротив - резко снижается. Благодаря децентрализации вычислительных мощностей достигнуто небывалое быстродействие. Например, "Машина связи" (США) с объёмом 1,5 $м^3$ построенная на базе 65000 микропроцессоров при одной из демонстраций за одну двадцатую долю секунды "прочитала" 16000 сообщений типа газетных новостей и за три минуты рассчитала схему кристалла с 4000 транзисторов.

Новые технологии производства сверхбольших интегральных схем (СБИС) в сочетании с новыми концепциями в архитектуре компьютеров (распределённая обработка данных) позволили достичь небывалого быстродействия. В ведущих странах мира (США, Япония, СССР) создаются суперЭВМ, используемые для решения сложных научно-технических задач. Их быстродействие достигает 1млрд. операций с плавающей точкой в секунду (1 мегафлоп), например, SX-2 (Япония), ёмкость оперативной памяти при этом достигает 2^{20} слов (Эльбрус-1, СССР).

Объёмы производства ЭВМ с каждым годом возрастают, одновременно уменьшается их стоимость. Это позволило сделать вычислительную машину "персональной" - поставить её на рабочее место каждого специалиста: инженера, технолога, учёного, администратора и т. д. Появление персональных ЭВМ вызвало также бурный рост числа программных средств, ориентированных на пользователя, обеспечивающих "дружественный" интерфейс человека и машины.

Развал Советского Союза больно ударил по всем республикам бывшего Союза. Разорвались экономические связи. Прекратилось или резко снизилось финансирование базовых отраслей науки и производства. Погоня за сверхприбылью нарождающейся национальной буржуазии привела к массовому ввозу зарубежной вычислительной техники, с которой отечественная техника не могла конкурировать. Это подорвало полностью национальную электронную промышленность. В лучшем случае нам удастся обеспечить сборку электронной техники из деталей, ввозимых из-за рубежа. Именно в последнее десятилетие 20 столетия, когда отечественная электронная промышленность оказалась в глубоком

параличе, в ведущих капиталистических странах, прежде всего США, Японии, Германии был сделан мощный скачок в области микроминиатюризации. Резко возросла степень интеграции микросхем, быстродействие процессоров, емкость памяти, появились новые графические операционные системы и ориентированные на них прикладные программы Word, Excel, Access и другие. Значительное развитие получили и периферийные устройства ввода-вывода информации.

Пятое поколение ЭВМ (с 1979).

Начало разработкам машин пятого поколения положено в 1979 году в Японии. Вычислительная система пятого поколения будет ориентирована на обработку знаний и будет располагать весьма развитыми возможностями логического вывода. Важнейшая черта её должна состоять в том, чтобы используемый интерфейс был непосредственно рассчитан на человека. Основными особенностями ПК пятого поколения будут речевой ввод-вывод информации и самообучаемость.

Технический базис её должна составить развивающаяся технология сверхбольших БИС, создание памяти повышенного объёма, возрастающие возможности высокоскоростных элементов.

Основу архитектуры должны составить системы с распределительными функциями, сетевая архитектура, машина базы данных, быстродействующая машина для численных расчётов, высокоуровневая система человеко-машинного общения.

Основными системами программного обеспечения должны стать системы управления базами знаний, системы решения проблем и логического вывода, системы интеллектуального интерфейса.

Основными прикладными системами могут стать системы машинного перевода, вопросно-ответная система, прикладные системы понимания речи, изображений, рисунков, прикладные системы решения проблем.

О практических результатах в области разработки машины пятого поколения ления говорить пока рано. Первыми практическими результатами в области искусственного интеллекта стали экспертные системы, с помощью которых достигнуты значительные результаты в области медицины, геологии, технической диагностики.

Невиданные успехи в области миниатюризации уже в последнее десятилетие привели к колоссальному росту объемов памяти и быстродействию вычислительной техники, а также к коренному изменению взглядов на ее использование. Ученые предполагают создание голографической памяти, которая позволит хранить терабайты информации на кубический дюйм. При такой емкости голографическая память объемом с кулак вместит все содержимое Библиотеки конгресса США. Кассеты для нового поколения цифровых видеомагнитофонов смогут хранить более 100 Гигабайт информации, то есть на одну единственную ленту удастся записывать все разговоры, которые человек ведет на протяжении жизни. Постепенное развитие компьютеров и технологии производства мониторов приведет к созданию почти невесомой, универсальной электронной книги. В коробке размером с обыкновенную книгу будут находиться дисплей, способный показывать текст, картинки и видеоматериалы с высоким разрешением. Перелистывать страницы можно будет пальцем или отдавать команды голосом.

Будущее развитие вычислительной техники связывают с вычислительными сетями. Предполагают, что глобальные сети превратятся в универсальный рынок и центральный универмаг всего мира. Проникновение вычислительной техники во все сферы человеческой деятельности изменит быт людей, социальные отношения. Появятся электронные банки и клиенты банка смогут оплачивать счета, заказывать и приобретать товары и услуги пользуясь электронным бумажником. Будут созданы системы биометрической защиты, которые смогут опознавать человека по отпечаткам пальцев, цвету радужной оболочки глаза, голосу. Системы виртуальной реальности позволят архитекторам видеть, например, устройство квартиры или офиса как бы изнутри. В костюмах виртуальной реальности, снабженных миллионами сенсорных датчиков, человек сможет путешествовать в космос, в пустыне Сахаре или джунглях Амазонки и ощущать космическую невесомость, зной пустыни или прикосновение лиан.

Успехи в области электроники последнего десятилетия позволяют с оптимизмом смотреть в будущее, все что сказано выше – не пустые фантазии, а вопрос времени и возможно не такого уж далекого.

Контрольные вопросы

1. Назовите основные этапы развития вычислительной техники.
2. Назовите ученых, которые внесли существенный вклад в развитие вычислительной техники.
3. Что положено в основу периодизации развития электронной вычислительной техники?
4. Какие научные открытия предшествовали появлению первых вычислительных машин на электронных лампах?
5. Назовите особенности и направления развития вычислительной техники пятого поколения.

3. Устройство и работа ЭВМ

3.1. Классификация ЭВМ

В настоящее время создан большой парк ЭВМ различного назначения. По мере увеличения количества ЭВМ, изменения их технических характеристик расширялась и область их применения: от стационарных ЭВМ большой производительности до микропроцессоров, встраиваемых в бытовую технику, от ЭВМ для управления космическим аппаратом до бытового компьютера. Можно выделить следующие основные классификационные признаки: вид обрабатываемой информации, вычислительная мощность, назначение, уровень организации, конструктивно-технологическое исполнение.

По виду обрабатываемой информации ЭВМ делятся на два больших класса: аналоговые и цифровые. Аналоговые ЭВМ служат для обработки медленно меняющихся сигналов тока или напряжения. Цифровые ЭВМ обрабатывают информацию, поступающую на их входы в дискретной форме или в виде цифровых кодов. В дальнейшем будут рассматриваться только цифровые ЭВМ, которые для краткости будем называть просто ЭВМ.

По вычислительной мощности ЭВМ делятся на микрокомпьютеры, мини-компьютеры, мэйнфреймы и суперкомпьютеры.

Суперкомпьютеры обладают высоким быстродействием и имеют огромные вычислительные мощности. Они используются для сложных расчетов в аэродинамике, метеорологии, космических и физических исследованиях, экономике и финансовом управлении.

Мэйнфреймы обладают значительными ресурсами для решения сложных задач в финансовой области, в управлении регионами, отраслями промышленности, большими предприятиями, в том числе предприятиями торговли, в военной области.

Мини-компьютеры используются для управления предприятиями и организациями. К ним относятся *серверы* старшего уровня, используемые для управления локальными компьютерными сетями.

Микрокомпьютеры – это самые массовые модели вычислительных машин. К ним относятся персональные компьютеры (ПК), рабочие станции. Рабочие станции используются в локальных вычислительных сетях, в том числе и в учебных заведениях.

По назначению ЭВМ делятся на вычислительные машины общего применения (универсальные), специализированные (проблемно-ориентированные), информационно-вычислительные, управляющие, персональные ЭВМ, проблемно-ориентированные процессоры. Основными параметрами, по которым различаются подклассы определённых средств вычислительной техники, являются разрядность, производительность и тип системного интерфейса.

Универсальные ЭВМ ориентированы на решение широкого круга задач, причём во всех классах задач они развивают одинаковую производительность. Эти ЭВМ имеют архитектуру, позволяющую подключать разнообразные периферийные устройства, варьировать их число и технические параметры, обеспечивать различные виды обработки данных и режимы взаимодействия с пользователями.

Специализированные ЭВМ предназначены для решения определённого класса конкретных прикладных задач, либо для ограниченных сфер применений, и, как правило, отличаются от универсальных ЭВМ рядом ограничений на виды обработки информации и/или возможности подключения периферийных устройств. Они оснащаются специальным программным обеспечением. Например, для обработки информации в геологических партиях при разведке полезных ископаемых.

Информационно-вычислительные ЭВМ ориентированы на обработку больших объёмов текстовой (справочной) информации.

Управляющие ЭВМ применяются для управления производством, техническими системами и технологическими процессами.

Персональные ЭВМ (ПЭВМ) предназначены для облегчения труда инженеров, учёных, все настойчивее проникают в наш быт, ориентированы на использование отдельным пользователем. В настоящее время за этим классом ЭВМ закрепилось название *персональный компьютер* (ПК). По функциональным возможностям персональный компьютер относится к универсальным ЭВМ.

Проблемно-ориентированные процессоры подключаются, как правило, к персональным ЭВМ и служат для повышения быстродействия при решении задач вычислительной математики, преобразования и обработки аналого-цифровой информации.

По уровню организации различают однопроцессорные (автономные) ЭВМ и вычислительные комплексы. Вычислительные комплексы отличаются от однопроцессорных ЭВМ повышенными характеристиками надёжности и готовности, а также концентрацией вычислительных мощностей в сочетании с их лучшим использованием по сравнению с набором независимых ЭВМ.

Признак конструктивно-технологического исполнения можно разделить на два дополнительных признака: место установки и число плат.

По месту установки ЭВМ делятся на: стационарные, бортовые, настольные, переносные встраиваемые, а *по числу плат* - на многоплатные и одноплатные.

Стационарные ЭВМ размещаются в специально оборудованных помещениях, где создаются необходимые условия по температурно-влажностному режиму и уровню шумов.

Бортовые ЭВМ размещаются на морских, речных, воздушных судах, космических аппаратах. К ним предъявляются повышенные требования по ударным нагрузкам, вибростойкости, устойчивости к ионизирующим излучениям, надёжности.

Переносные ЭВМ размещаются в специальных чемоданах, в качестве одной из разновидностей переносных ЭВМ являются карманные ЭВМ.

Встраиваемые ЭВМ не оформляются в виде самостоятельных приборов, а встраиваются непосредственно в оборудование (станки, приборы, узлы, агрегаты), образуют с ними конструктивное целое.

Одноплатные ЭВМ имеют сравнительно небольшой объем памяти. Расширение их возможности идёт за счёт добавления модулей памяти, а также устройств, обеспечивающих связь с внешними устройствами (контроллеров), источников питания. Названные устройства выполняются, как правило, в виде отдельных плат. Таким образом одноплатные ЭВМ превращаются в *многоплатные.*

3.2. Структура и принцип работы ЭВМ

Типичная ЭВМ состоит из четырёх основных частей: процессора, памяти, устройств ввода и вывода информации. Все составные части ЭВМ связаны между собой шинами адреса, данных и управления. Обобщённая структурная схема фон-неймановской ЭВМ приведена на рис. 3.2.1.

Рис.3.2.1. Упрощенная структура компьютера

Процессор

Процессор является ядром вычислительной системы и предназначен для обработки информации и управления работой всей системы. Основными его узлами являются арифметикологическое устройство (АЛУ), устройство микропрограммного управления (УУ) и регистры различного назначения (РОН), КЭШ-память.

Тактовый генератор (ТГ) формирует тактовые импульсы, синхронизирующие работу всех устройств компьютера. Ввиду того, что быстродействие различных устройств ЭВМ различно, генератор тактовых импульсов формирует сигналы на нескольких частотах.

Арифметикологическое устройство предназначено для выполнения простейших операций: сложения, сдвига, логических операций И, ИЛИ, исключающего ИЛИ. Составив соответствующую программу, на основе этих операций можно выполнить и более сложные операции, такие как умножение, возведение в степень и т. д. АЛУ связано с регистрами, аккумулятором (специальный однобайтовый регистр) и устройством управления.

Устройство управления является одним из важнейших узлов процессора. Совместно с генератором тактовых импульсов оно обеспечивает правильную последовательность выполнения всех операций в ЭВМ. После извлечения команды из памяти и её дешифрации устройство управления генерирует последовательность сигналов, необходимую для выполнения команды. Он выполняет и другие функции, например, прерывание программы.

Регистры имеют разное назначение, но, независимо от этого, их общая функция состоит во временном хранении информации: команд, адресов, данных или признаков состояния АЛУ.

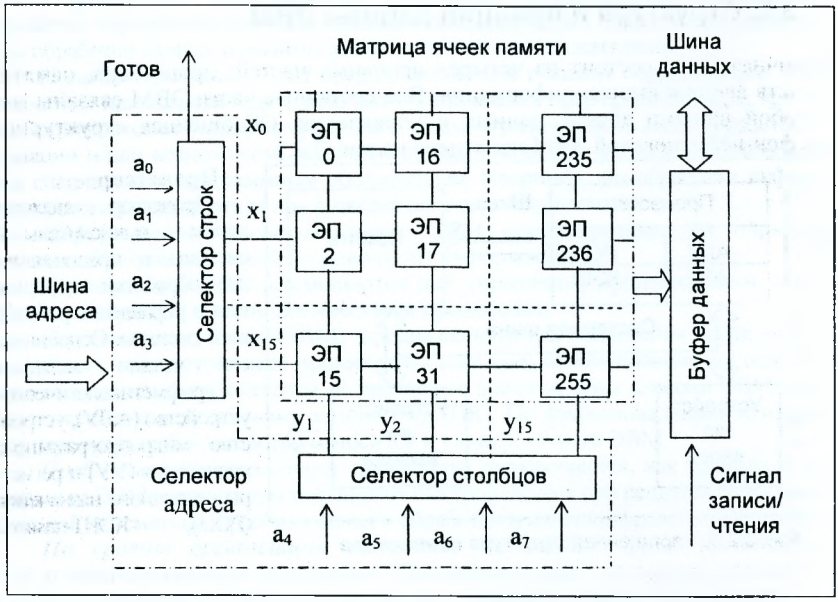


Рис. 3.2.2. Структура блока памяти

КЭШ - память – оперативная память компьютера первого уровня. Служит для хранения наиболее часто используемых программ и данных.

Память

Память компьютера служит для хранения программ и данных. Она состоит из ячеек, связанных с шинами данных и адреса (рис. 3.2.2.).

Ячейки памяти (элементы памяти) служат для хранения информации. Элементом памяти в компьютерах является обычно байт. Число ячеек в блоках памяти зависит от разрядности шины адреса. Например, при 8-и разрядной шине адреса блок памяти может содержать 256 ячеек, а при 16-и разрядной шине адреса число ячеек в блоке равно $2^{16} = 65536$, или 64 Кбайт. Ячейки нумеруются номерами от 0 до 65535. Ячейки памяти объединяются, как правило, в матрицы.

Селектор адреса состоит из селекторов строк и столбцов и предназначен для выбора ячейки с заданным адресом с целью передачи содержимого ячейки в шину данных или наоборот, записи данных в ячейку. По окончании селекции ячейки с заданным адресом он выдаёт в процессор по шине управления сигнал "Готов". При записи информации в память этот сигнал указывает на то, что данные могут вводиться в память. При чтении данных сигнал "Готов" разрешает процессору начать приём данных.

Буфер данных служит для временного хранения принимаемой или выдаваемой из памяти информации. После получения от селектора адреса сигнала "Готов" процессор выдает по шине управления команду на запись или чтение данных.

Шина

Шина представляет собой стандартный комплект токопроводящих линий с устройствами согласования и служит для передачи данных и адреса для выбора ячеек, а также сигналов управления процессора.

Шина адреса имеет 16 или 32 разряда, однонаправленная служит для передачи адреса выбираемого элемента памяти или адреса порта ввода-вывода к соответствующим устройствам.

Шина данных может быть двунаправленной, и может иметь 8, 16, 32 или 64 разряда, в зависимости от разрядности процессора. Она служит для передачи данных от микропроцессора к устройствам ввода-вывода и памяти или от устройств ввода-вывода и памяти к процессору.

Шина управления двунаправленная. Она служит для передачи сигналов записи, чтения, тактовых сигналов, сигналов синхронизации от процессора к блокам памяти, устройствам ввода-вывода, а также для передачи сигналов готовности от указанных устройств к процессору.

Устройства ввода-вывода

Устройства ввода обеспечивают связь процессора с источниками информации, согласуют входные сигналы по уровню напряжения и формату с уровнем сигналов микросхем, используемых в процессоре, обеспечивают гальваническую развязку электрических цепей источников информации и процессора, а также источников информации и устройств ввода. В качестве источников информации могут быть машинные носители программ (перфокарты, перфоленты, магнитные диски и др.), клавиши устройств ввода информации, датчики состояния систем управления (температуры, давления, магнитных полей и др.).

Устройства вывода обеспечивают связь процессора с исполнительными устройствами и выполняют практически те же функции, что и устройства ввода. Отличие состоит в направлении передачи информации. В устройствах вывода информация передаётся от процессора к внешним устройствам. Кроме того, выходные устройства обеспечивают усиление выходных сигналов по напряжению и мощности до величины, необходимой для управления исполнительными устройствами. В качестве исполнительных устройств могут быть устройства вывода информации на машинные носители (магнитные диски, перфокарты, перфоленты и др.), дисплеи, графопостроители, индикаторы, исполнительные устройства систем управления техническими системами и технологическим оборудованием (реле, электроприводы, пусковая аппаратура систем электроснабжения), устройства формирования звуковых сигналов и др.

Принцип работы ЭВМ

В ЭВМ используется *принцип программного управления*. Один из способов его реализации был предложен в 1945 г. американским математиком Д. Нейманом, и с тех пор неймановский принцип программного управления используется в качестве основного принципа построения персональных ЭВМ. Этот принцип состоит в следующем:

- информация кодируется в двоичной форме и разделяется на единицы информации - слова;
- разнотипные слова информации различаются по способу использования но не по способам кодирования;
- слова информации размещаются в памяти ЭВМ и идентифицируются номерами ячеек, которые называются номерами слов;
- *алгоритм* представляется в виде последовательности управляющих слов - команд, которые определяют наименование операции и слова информации, участвующие в операциях. Алгоритм, представленный в терминах машинных команд, называется *программой*;
- выполнение вычислений, предписанных алгоритмом, сводится к последовательному выполнению команд в порядке, однозначно определяемом программой. Первой выполняется команда, заданная пусковым адресом программы. Обычно это адрес первой команды программы. Адрес следующей команды однозначно определяется в процессе выполнения текущей команды и может быть либо адресом следующей по порядку команды, либо адресом любой другой команды. Процесс вычислений продолжается до тех пор, пока не будет выполнена команда, предписывающая прекращение вычислений.

Основные характеристики ЭВМ

Характеристики ЭВМ определяют её назначение, область применения и потребительские качества. К ним относятся следующие показатели:

1. Состав и типы подключаемых внешних устройств.
2. Тип процессора. Наибольшее распространение в персональных ЭВМ в настоящее время имеют процессоры Pentium III, Pentium 4, Celeron фирмы Intel, K5, K6, K7 (Athlon), Duron фирмы AMD.

3. Разрядность. Разрядность ЭВМ определяется разрядностью процессора и характеризует точность вычислений и производительность машины. Различают 8-, 16-, 32- разрядные ЭВМ.

4. Быстродействие - число элементарных операций, выполняемых в единицу времени (оп/с). Быстродействие определяется тактовой частотой задающего генератора. Первые ПК имели тактовую частоту 4, 8, 16 МГц. В настоящее время частота тактового генератора достигает 2 ГГц и, и будет повышаться далее в связи с освоением новых технологий. Например, в ноябре 2000 г. был выпущен процессор Pentium 4 с тактовой частотой 1,5 ГГц, изготовленный по 0,18 микронной технологии (под технологией процессора понимается наименьший размер одного элемента, например транзистора, диода, конденсатора). А в настоящее время уже выпускаются процессоры данного типа с тактовой частотой 2 ГГц.

5. Ёмкость памяти определяет возможности ЭВМ по использованию современных пакетов прикладных программ. Установленная оперативная память достигает 256 Мбайт, КЭШ – память первого и второго уровней составляет 128 - 256 Кбайт.

6. Ёмкость внешних запоминающих устройств (ВЗУ) определяет объём хранимой и используемой информации. Емкость накопителей на жестком диске достигает 100 Гбайт.

7. Программное обеспечение: операционная система, системы программирования, пакеты прикладных программ.

8. Массогабаритные характеристики.

9. Стоимость.

3.3. Устройство персонального компьютера

3.3.1. Состав блоков персонального компьютера

Персональные компьютеры независимо от типа, фирмы изготовителя имеют много общего. Отечественной промышленностью в 90 годах прошедшего столетия выпускалось несколько моделей персональных компьютеров раз-

Рис. 3.3.1. Минимальная конфигурация ПК

ной производительности, в том числе ПК серии ЕС: ЕС 1840, ЕС 1841, ЕС 1842, ЕС 1849, ЕС 1851, ЕС 1861. После развала СССР в 1992 году, и связанных с этим изменением экономических условий, в страну хлынул поток зарубежной вычислительной техники, которая по многим показателям превосходила отечественную. Это привело практически к полной гибели отечественной промышленности по производству вычислительной техники.

Независимо от фирмы производителя в состав современного ПК входят системный блок, видеомонитор, клавиатура, манипулятор графической информации "Мышь" (рис. 3.3.1). Перечисленные блоки составляют базовую (минимальную) конфигурацию ПК. Для получения твердой копии документов необходимо также печатающее устройство (принтер) индивидуального или коллективного пользования.

Кроме перечисленных блоков к ПК могут подключаться дополнительные внешние устройства и модули, обеспечивающие профессиональную ориентацию компьютера.

Системный блок предназначен для размещения электронных плат, источников питания, накопителей на магнитных дисках, элементов управления. На лицевой панели блока (рис. 3.3.1) размещены:

- накопитель для гибких магнитных дисков размером 140 мм (5,25 дюйма). На этом месте в настоящее время чаще размещают накопитель для чтения компакт-дисков (CD-ROM – сидиром), имеющий тот же размер;
- накопитель для гибких магнитных дисков размером 89 мм (3,5 дюйма);
- кнопки Power - включения питания и Reset - горячего перезапуска компьютера в случае его зависания (программа не реагирует на нажатие клавиш на клавиатуре);
- на старых моделях компьютеров может быть также клавиша Turbo, предназначенная для переключения процессора на ускоренный режим работы;
- индикаторы включения питания, ускоренного режима работы процессора и работы накопителя на жестком диске.

Внутри корпуса системного блока размещены системная плата, платы адаптеров дисководов, печатающего устройства, видеомонитора, накопитель на жестком диске, блок питания.

Системная плата (материнская плата) представляет собой одноплатную микроЭВМ и может, при необходимости, функционировать самостоятельно. На ней размещены: микропроцессор, КЭШ – память второго уровня, ПЗУ и ОЗУ, блок управления, программируемый системный таймер, 4-х канальный блок прямого доступа к памяти, адаптеры связи с динамиком и клавиатурой, системная шина.

Системная плата через разъёмы системной шины соединяется с выполненными на отдельных печатных платах модулями дополнительного оборудования: блоком оперативной памяти, адаптерами видеомонитора, принтера, накопителей на магнитных дисках.

На заднюю панель системного блока выведены разъёмы, через которые подключаются внешние устройства (ВУ): дисплей, принтер, клавиатура, манипулятор графической информации мышь (разъемы принято называть портами).

Персональные компьютеры могут работать с периферийными устройствами, имеющими связь по параллельному интерфейсу *(Centronies)* или стыку последовательного интерфейса *(RS-232C, USB*). ПК может иметь один или два стыка последовательного интерфейса.

К системной шине может подключаться также модуль расширения или модуль профессиональной ориентации. Эти модули обеспечивают усиление и ретрансляцию сигналов системной шины.

Адаптеры (или контроллеры) являются сложными электронными устройствами, в состав которых могут входить также и микропроцессоры. Адаптеры обеспечивают связь внешних устройств с центральным процессором.

Блок питания служит для обеспечения элементов электронных схем и дисководов стабилизированным и нестабилизированным напряжением требуемых номиналов.

3.3.2. Микропроцессор

Процессор является ядром вычислительной системы и предназначен для обработки информации и управления работой всей системы. Процессоры выпускаются многими фирмами. Однако наибольшее распространение для бытовых компьютеров имеют процессоры американских фирм Intel и AMD. Основные характеристики процессоров приведены в табл. 3.3.1

Из анализа данной таблицы можно сделать вывод о том, какими темпами развивалась микропроцессорная техника с момента появления первого компьютера.

На момент подготовки данного пособия промышленно выпускались и реализовывались через торговую сеть процессоры Celeron, Pentium 4, Duron, Athlon с частотой до 2000 МГц.

Быстродействие компьютера определяется числом элементарных операций, выполняемых в единицу времени (оп/с). Быстродействие зависит от тактовой частоты задающего генератора. Первые ПК имели тактовую частоту 4, 8, 16 МГц. В настоящее время частота тактового генератора достигает 2 ГГц и будет повышаться далее в связи с освоением новых технологий. Кроме повышения тактовой частоты, увеличение производительности процессоров достигается путем разработки новых архитектур и алгоритмов обработки информации. К основным приемам повышения производительности компьютера относятся:

- суперскалярная архитектура;
- конвейерная обработка информации;
- наличие разделенных КЭШ-памяти для команд и данных;
- использование блока предсказания адреса;
- использование блока вычислений с плавающей точкой;
- поддержка многопроцессорного режима работы;
- наличие средств обработки ошибок.

Термин "*суперскалярная архитектура*" означает, что процессор имеет более одного вычислительного блока. Эти вычислительные блоки называются ***конвейерами.*** Наличие в процессоре двух конвейеров позволяет ему выполнять

одновременно две команды. Каждый конвейер разделяет процесс выполнения команды на несколько этапов:

Таблица 3.3.1

Характеристики процессоров

Модель МП	*Разрядность, бит*		*Тактовая частота, МГц*	*Число команд*	*Технология, мк*	*Число транзисторов, тысяч*	*Год выпуска*
	шины данных	*шины адреса*					
Процессоры фирмы Intel							
4004	4	4	0,75	45		2,3	1971
8080	8	8	4,77			10	1974
8086	16	16	4,77 и 8	134		29	1978
80286	16	24	6 ... 12		1,5	130	1982
80386	32	32	16 ... 33	240	1,5- 1	275	1985
80486	32	32	33 ... 100	240	1-0,8	1200	1989
Pentium	64	32	75 ... 200	240	0,8	3100	1993
Pentium II	64	32	233-300		0,35	7500	1997
Pentium III	64	32	450-600		0,18	9500	1999
Pentium 4	64	32	1500		0,18	42000	2000
Процессоры фирмы AMD							
K5	64	32	75-166		0,6-0,35		-
K6	64	32	166-233		0,35-,25		1997
K7 (Athlon)	64	32	266-500				-
Sledge-Hammer	64	32	1500 и выше				-

- выборка (считывание) команды из ОЗУ или КЭШ - памяти;
- декодирование (дешифрация) команды;
- выполнение команды;
- обращение к памяти;
- запоминание полученных результатов в памяти.

При конвейерной обработке информации на выполнение каждого этапа отводится один такт синхронизирующей частоты. В каждом новом такте заканчивается выполнение одной команды и начинается выполнение другой команды. Длительность выполнения команды определяется при этом временем на выполнение самого продолжительного этапа, а не временем выполнения всей команды.

Разделение КЭШ - памяти на две части позволяет избежать структурных конфликтов, когда две команды одновременно обращаются к одному и тому же блоку памяти.

Развитие вычислительной техники идет непрерывно. В процессорах 80386 и 80486 для ускорения вычислений с плавающей точкой встраивался специальный математический сопроцессор. В последующих моделях, в связи с увеличением плотности размещения элементов на кристалле, математический со-

процессор реализован непосредственно в процессоре. В процессоре Pentium имеется два пятиступенчатых конвейера для выполнения операций с фиксированной точкой и один восьмиступенчатый конвейер для выполнения операций с плавающей точкой. Кроме выполнения математических расчетов этот конвейер используется также для быстрой обработки динамических трехмерных изображений.

Одним из путей повышения производительности является использование блока предсказания адреса. При выполнении разветвляющихся процессов и циклов можно заранее предсказать направление перехода и записать адрес перехода в специальный буфер предсказания адреса. Если результат выполнения текущей операции не совпал с содержанием буфера, то адрес считывается из памяти. То есть процессор практически ничего не теряет. Если же адрес перехода записан правильно, то он выбирается из буфера, имеющего малое время доступа, а не из памяти, имеющей значительное большее время доступа. Для реализации этой идеи в процессоре используется два буфера предварительной выборки. Один из буферов выбирает команды последовательно, а другой – согласно предсказаниям блока предсказания адреса.

3.3.3. Память ЭВМ

Классификация памяти персонального компьютера

Различают несколько видов памяти. Во первых, различают внутреннюю и внешнюю память. *Внутренняя память* размещается на системной плате, вблизи от процессора. *Внешняя память* размещается, как правило, внутри системного блока или в дополнительных внешних устройствах.

Внутренняя память делится на: *постоянную* - постоянное запоминающее устройство (ПЗУ), *оперативную* - оперативное запоминающее устройство (ОЗУ), *перепрограммируемую* - перепрограммируемое запоминающее устройство (ППЗУ), другое ее название – КМОП-память, такое название она получила по технологии изготовления, *КЭШ-память*.

Внешняя память энергонезависимая. Она предназначена для длительного хранения программ и данных. В персональных компьютерах находит применение память на магнитных дисках, магнитных лентах и оптических дисках, а также электронные внешние запоминающие устройства.

ПЗУ и ВЗУ

Основными характеристиками памяти являются время доступа или максимальная частота, с которой может работать эта память, емкость. *Время доступа* представляет собой промежуток времени между формированием запроса на чтение информации из памяти и моментом поступления из памяти запрошенного машинного слова (операнда). *Длительность цикла* – величина обратная максимальной частоте определяется минимально допустимым временем между двумя последовательными обращениями к памяти. Время доступа достигает 3 наносекунд (нс), а максимальная частота - 333 МГц. *Объем* (или емкость) одного модуля памяти может составлять от 1 до 128 Мбайт. На практике чаще используются модули памяти 32, 64, 128 Мбайт.

Постоянная память (ROM) обычно имеет объём 8 - 64 Кбайт. Микросхемы постоянной памяти устанавливаются на системной плате. Данные записываются в

ПЗУ при изготовлении компьютера и хранятся в нём без изменений. В ПЗУ хранятся вспомогательные программы, используемые микропроцессором при выполнении функций управления компьютером, а также программы инициализации и проверки оборудования, которые автоматически запускаются при включении электропитания системного блока компьютера. В функции этих программ входит проверка работоспособности ОЗУ, клавиатуры, подключенных внешних устройств и загрузка операционной системы (ОС). Кроме того, в ПЗУ имеется программа Setup, которая позволяет изменять параметры, определяющие конфигурацию компьютерной системы и необходимые для работы программ ПЗУ. ПЗУ выпускается в виде отдельной микросхемы, например AMI BIOS.

Оперативная память (RAM). Микросхемы ОЗУ расположены на системной и дополнительной платах компьютера. Объём памяти ОЗУ составляет 8, 16, 32, 64, 128, 256 Мбайт. В ОЗУ информация загружается с началом работы ПК. Сначала туда записываются файлы операционной системы (IO.SYS, MSDOS. SYS, а также сервисные (обслуживающие) программы, указанные пользователем и являющиеся резидентными. Затем по командам пользователя загружаются трансляторы и другие пользовательские программы. Память регенерации экрана монитора (видеопамять) размещается в адаптере видеомонитора.

Различают два вида оперативной памяти: статическую и динамическую. Эти два вида памяти отличаются быстродействием и удельной плотностью (емкостью) хранимой информации. *Статическая память* (Static RAM, SRAM) в качестве элементарной ячейки использует, так называемый, статический триггер, схема которого состоит из 4 - 6 транзисторов. Состояние статического триггера может сохраняться как угодно долго. Данный тип памяти имеет высокое быстродействие, но низкую плотность размещения хранимых данных. *Динамическая память* (Dinamic RAM, DRAM) представляет собой набор конденсаторов, выполненных в структуре полупроводникового кристалла. Для построения элемента динамической памяти требуется 1 – 2 транзистора, используемых для подключения и отключения конденсатора. Эта память обладает меньшим быстродействием по сравнению со статической памятью, требует периодической регенерации (это связано с разрядом конденсаторов), но позволяет создавать блоки памяти большой емкости.

В компьютерах в настоящее время применяется чаще всего динамическая память. Известно довольно много типов микросхем, отличающихся способами синхронизации, выборки информации и допустимой частотой (EDRAM, FRM DRAM, EDORAM, SDRAM, SLDRAM, RDRAM и др.). Преобладает на рынке микросхем памяти микросхема SDRAM – динамическая память с синхронным доступом. Она отвечает спецификации PC100, то есть поддерживает частоту 100 Мгц, время доступа 10 нс. Разработаны новые чипы памяти VCM133, которые отвечают спецификации 133 МГц и обеспечивают время доступа 7-8 нс, а также разработана скоростная DDR SDRAM, которая обеспечивает практически удвоенную скорость передачи данных по сравнению с SDRAM. VCM133 может быть встроена в DDR SDRAM. Однако, как показывает практика и эти сведения уже данные вчерашнего дня. В настоящее время на компьютерном рынке предлагаются чипы памяти с частотой 266 МГц и 333 МГц (DDR DIMM)

Для повышения производительности компьютера на нем устанавливается также КЭШ-память второго уровня, емкостью до 512 Кбайт. КЭШ – память

статическая. Эта память устанавливается вблизи процессора на системной плате и связана с ним скоростной шиной, работающей на более высокой частоте, чем остальная память.

Накопители на магнитных дисках

В настоящее время наиболее широкое распространение в качестве устройств хранения информации имеют внешние запоминающие устройства на магнитных носителях (дисках, лентах). Принцип действия их основан на использовании явлений электромагнитной индукции. В основе процесса магнитной записи лежит взаимодействие магнитного носителя информации (запоминающей среды) и магнитных головок - миниатюрных электромагнитов, располагаемых у поверхности движущегося магнитного носителя с небольшим зазором, при бесконтактной записи, или без зазора, при контактной записи.

Составными частями устройств записи/чтения являются носитель информации – лента или диск, покрытый порошком из ферромагнитного сплава, имеющего узкую петлю гистерезиса, и электромагнит, расположенный у поверхности движущегося носителя или на небольшом расстоянии от него (рис.3.3.2). При записи информации электрический ток, проходящий по обмотке электромагнита записывающей головки, создает в зазоре электромагнита магнитный поток, который взаимодействует с магнитным покрытием движущейся ленты или диска. Под действием этого магнитного потока, в зависимости от направления тока, происходит намагничивание или размагничивание магнитного покрытия носителя. Разные направления намагниченности соответствуют логическому нулю или логической единице информации. При чтении информации с магнитного носителя движущиеся элементарные магнитики наводят в обмотке считывающей головки электродвижущую силу, направление которой зависит от намагниченности носителя информации.

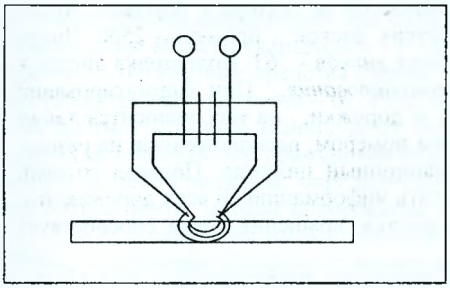


Рис.3.3.2. Принцип записи информации на магнитный носитель

На современных компьютерах устанавливаются, как правило, накопитель для гибких магнитных дисков размером 89 мм и накопитель на жестком диске.

Дискеты размером 89 мм (3,5 дюйма) имеют емкость 1,44 Мбайта. Дискета представляет собой пластмассовую или алюминиевую пластину, покрытую тонким слоем ферромагнитного материала. Для обеспечения сохранности эта пластина помещается в пластмассовый конверт. Конверт имеет окно чтения/записи информации. В нерабочем состоянии это окно закрывается металлической крышкой (заслонкой). На корпусе имеются технологические вырезы, обеспечивающие безошибочную установку дискеты в дисковод и два небольших квадратных технологических окна, одно из которых служит для защиты дискеты от несанкционированного изменения хранимой информации. Если это

окно закрыто, то дискета доступна для записи и чтения информации, а если открыто, то возможно только чтение информации.

Рис. 3.3.3. Принцип устройства жесткого диска

Жесткий диск представляет собой пакет из 4 – 9 пластин, выполненных из алюминия, латуни, керамики или стекла, толщиной примерно 2 мм, покрытых ферромагнитным материалом. Емкость жесткого диска достигает 60 – 100 Гбайт. Скорость вращения диска составляет обычно 3600, 4500, 5400, 7200, 10000, 12000 оборотов в минуту. Для чтения информации имеется блок головок чтения-записи информации (рис. 3.3.3). Вращение дисков и перемещение головок осуществляется с помощью двух электродвигателей.

Для обеспечения работы диск разбивается на сектора и дорожки. Число дорожек для дискет равно 80, а для жестких дисков – примерно 2500. Число секторов для дискет равно 18, а для жестких дисков - 63. Подготовка дисков к работе осуществляется в процессе *форматирования*. При форматировании дисков, кроме разбиения их на сектора и дорожки, на них наносится также служебная информация. Дорожки с одним номером, расположенные на разных дисках образуют своеобразный информационный цилиндр. Подведя головки чтения к нужной дорожке, можно прочитать информацию со всех дорожек этого цилиндра, что наряду с высокой скоростью вращения диска способствует уменьшению времени доступа.

Накопители на жестких дисках выпускаются различных размеров (форм-факторов) 1,8", 2,5", 3,5", 5,25". Емкость каждого сектора, независимо от номера дорожки постоянна. Поэтому на внешних дорожках плотность записи ниже, а на внутренних выше. На жестких дисках форм-факторов 3,5" и 5,25", с целью повышения емкости, диск разбивается на зоны, в пределах которых число секторов постоянно, чем дальше зона от центра, тем больше она содержит секторов.

Среди основных параметров дисков, определяющих их производительность, выделяют следующие: среднее время доступа, которое определяется временем позиционирования магнитных головок на дорожке и временем ожидания сектора, и скорость обмена данными, которая зависит от используемого интерфейса. Время доступа складывается из среднего времени перемещения между двумя случайно выбранными дорожками (8 – 20 мс) и времени ожидания нужного сектора (4 – 8 мс). Максимальное значение скорости передачи данных достигает 80 Мбайт/с.

Переносные накопители на магнитных дисках

В настоящее время, кроме перечисленных традиционных средств хранения информации, используя современные технологии, такие как эффект Бернулли для бесконтактной записи, применение лазерного луча для позиционирования головок чтения- записи, выпускаются переносные диски большой емко-

сти. Благодаря этому на дискетах размером 3,5" можно хранить от 100 Мбайт до 2 Гбайт информации.

Компания Iomega выпускает мобильный *дисковод Iomega ZIP 250*. Дисковод работает с дискетой собственного формата объемом 100 и 250 Мбайт. Скорость чтения данных – 1-1,5 Мбайта/с.

Дисковод Iomega JAZ – представляет собой настоящий переносной жесткий диск. Он работает с дискетами собственного формата емкостью 1 и 2 Гбайт. Скорость чтения – от 3,5 до 6,6 Мбайта/с. Выпускается с интерфейсом SCSI.

Компания Castlewood в конце 1997 года представила новый *дисковод ORB* емкостью 2 Гбайта, скорость чтения – до 12 Мбайт/с.

RAID массив – устройство, состоящее из нескольких винчестеров и RAID-контроллера. Это устройство обладает большим объемом дискового пространства, повышенной скоростью обмена данными, значительной надежностью хранения информации.

Накопители на магнитных лентах

Накопители на магнитных лентах – *стримеры* позволяют сохранять большие объемы информации при создании архивных копий. Они используют специальные кассеты (картриджи) с магнитной лентой. Стримеры имеют, как правило, собственные средства сжатия информации при записи на ленту. Емкость картриджей различна – от сотен Мбайт до десятков Гбайт. Для повышения плотности записи используется технология спирального сканирования (в других источниках эту технологию называют – поперечная запись). Стримеры имеют разные стандарты, определяющие интерфейс с пользователем, формат магнитной ленты, методы кодирования и сжатия .

QUC – стримеры имеют толщину картриджа 1/4", емкость до 1,3 Гбайт. *Travan* – стримеры используют магнитные ленты шириной 0,315" и имеют емкость до 4 Гбайт. В *DAT* – стримерах используется технология спирального сканирования. Емкость достигает 8 Гбайт. *DLT* – стримеры имеют высокую надежность в эксплуатации и большую емкость – до 35 Гбайт.

Накопители на оптических и магнитооптических дисках

Устройства для чтения компакт-дисков получили название CD-ROM. В качестве накопителей информации для них используются в настоящее время оптические (или лазерные) и магнитооптические диски. Они имеют размер 133 мм и емкость до 700 Мбайт.

По способу организации чтения/записи информации эти диски можно разделить на три группы: только для чтения (CD); с однократной записью и многократным считыванием (CD-R), перезапись таких дисков невозможна; перезаписываемые диски (CD-RW). Эти диски имеют разную технологию изготовления.

Первыми на свет появились оптические диски только для чтения. Носителем информации на CD-диске является подложка из поликарбоната, на которую нанесен тонкий слой отражающего металла (обычно алюминия). В основе записи информации на эти диски с помощью лазера лежит модуляция интенсивности излучения лазера дискретными значениями нуля и единицы. Вначале создается матрица. При записи логической единицы луч лазера прожигает в матрице микроскопическое отверстие. Запись начинается с внутренних доро-

жек и ведется по спирали с большой плотностью – 630 дорожек на миллиметр. Таким способом создается первичный "мастер-диск". Затем осуществляется тиражирование этого диска методом литья под давлением или штамповкой. Выступы теперь соответствуют логической единице, а впадины – логическому нулю. Получившаяся рельефная основа металлизируется и покрывается лаком, защищающим тонкую металлическую поверхность. Чтение информации также осуществляется лучом лазера, но меньшей интенсивности. Поверхность диска по разному отражает луч при наличии на нем выступов и вмятин. От выступов получается более мощный отраженный сигнал, а от впадин - более слабый.

В оптических дисках с однократной записью и многократным считыванием основу диска составляет алюминиевый или пластмассовый диск, на который нанесен тонкий слой металла - теллура. В этом слое металла также создается матрица хранимой информации лучом лазера.

Магнитооптические диски с однократной записью и многократным считыванием выполняется из стекла, на поверхность которого наносится сплав из тербия, железа и кобальта, обладающий магнитными свойствами. Этот сплав имеет также низкую температуру Кюри – температура, при которой возможно перемагнитить сплав. При записи информации с помощью лазера производится разогрев металла на ограниченном участке диска до точки Кюри и прикладывают магнитное поле нужного направления. После остывания данный участок запоминает направление намагниченности. При нормальной температуре информация на диске не подвержена воздействию внешних магнитных полей. При считывании информации используется эффект Керра, который проявляется в изменении направления поляризации лазерного луча, отраженного от намагниченной поверхности. Такие диски читаются на любом приводе CD-ROM.

В перезаписываемых накопителях используется свойство рабочего слоя переходить под воздействием луча лазера в кристаллическое или аморфное состояние , имеющее разную отражающую способность. Эти диски могут не читаться на некоторых устаревших приводах CD-ROM.

Основными достоинствами накопителей на оптических и магнитооптических дисках являются: высокая плотность записи информации; универсальность, т. е. пригодность для хранения информации, заданной в различной форме; возможность быстрой перезаписи больших объемов информации и длительного ее хранения; высокая надежность сохранности информации.

Устройства для работы с оптическими дисками

Устройства для работы с оптическими дисками (CD-ROM) различаются по скорости считывания информации и возможности записи информации.

Дисководы имеют ту же систему обозначений, что и накопители: CD – устройства для чтения оптических дисков; CD-R – устройства для чтения и однократной записи оптических дисков; CD-RW - устройства для записи и чтения оптических и магнитооптических дисков.

За единицу скорости считывания принята скорость чтения звуковых компакт дисков -- 150 Кбайт/с. У современных CD-ROM скорость считывания значительно выше, она указывается в обозначении диска. Например, Creative 24xMX означает 24-скоростная. Однако цифра 24 это не средняя, а максимальная скорость записи. Средняя скорость записи будет раза в полтора меньше.

Универсальный цифровой диск DVD

Другой новинкой среди носителей информации является DVD-диск – универсальный цифровой диск, который по внешнему виду мало чем отличается от CD – дисков. Предназначен для хранения видео, аудио высокого качества компьютерной информации. Эти диски обладают большой емкостью. Односторонние однослойные DVD имеют емкость 4,7 Гбайт информации, двухслойные – 8,5 Гбайт, двухсторонние однослойные – 9,4 Гбайт, а двухсторонние двухслойные накопители могут вмещать до 17 Гбайт информации. Накопители DVD могут читать обычные CD-ROM-диски. Двухсторонние DVD могут читать и CD-R- и CD-RW-диски.

Накопители DVD-RAM позволяют записывать и перезаписывать информацию. На одностороннем однослойном диске можно разместить 2,58 Гбайт данных, на двухстороннем – 5,2 Гбайт, DVD-R позволяет хранить 3,95 Гбайт информации.

Электронные внешние запоминающие устройства

Запоминающие устройства на цилиндрических магнитных доменах

Для ЭВМ повышенной надёжности и портативных ЭВМ перспективны запоминающие устройства на цилиндрических магнитных доменах (ЦМД), характеризующиеся отсутствием механических узлов, высокой стойкостью к загрязнению, широким диапазоном рабочих температур, а также возможностью предварительной сортировки и логической обработки данных в самом накопителе.

Память на цилиндрических магнитных доменах занимает промежуточное положение между ОЗУ и НМЛ. Некоторые материалы обладают тем свойством, что их тонкие плёнки на подложке содержат случайно распределённые магнитные домены, одноосные поля которых перпендикулярны плоскости плёнки. При подаче внешнего магнитного поля с направлением, противоположным полю доменов, домен может сжаться в маленький цилиндр (диаметром от 1 до 10 мкм) или исчезнуть совсем. При подаче магнитного поля, направленного в плоскости плёнки, магнитные домены можно передвигать в плёнке. Если к плёнке приложен пермалоевый шаблон, домены можно заставить двигаться по направлениям, определяемым шаблоном. Если между плёнкой и шаблоном поместить проводящую сетку, то токи в сетке можно использовать для локального управления магнитным полем так, что отдельные домены можно генерировать, аннигилировать, увеличивать в размерах и расщеплять (дублировать) или заставлять перемещаться. Таким образом, появляется возможность построить память в виде длинного регистра сдвига (или несколько регистров), в котором наличие домена означает 1, а отсутствие - 0. Как и все магнитные устройства, эта память оказывается энергозависимой.

ЦМД -память выпускается в виде БИС ёмкостью от 64 Кбит до 1 Мбайт. Время доступа к информации, расположенной в памяти, построенной из таких БИС, на порядок меньше, чем у жестких дисков.

В общем случае ЗУ на ЦМД состоят из контроллера и накопителя в виде одного или нескольких идентичных модулей. Каждый модуль выполнен на отдельной печатной плате, на которой расположены от одной до четырёх магнитных интегральных схем (МИС), называемых иногда ЦМД-микросхемами. Модуль накопителя выполняется также в виде отдельной съёмной кассеты в изолированном корпусе.

Основные характеристики запоминающего устройства на ЦМД:

ёмкость МИС от 0,256 до 2 Мбит, число МИС в накопителе от 1 до 32, среднее время доступа 6 мс, скорость обмена данными 400 Кбит/с.

Наиболее целесообразно использовать в ПК и микроЭВМ повышенной надёжности для эмуляции НГМД и НМД в тех областях, где использование традиционных ВЗУ недопустимо.

Широкое использование внешних запоминающих устройств на ЦМД в персональных компьютерах сдерживается до настоящего времени относительно большой потребляемой мощностью, временем доступа и особенно высокой удельной стоимостью.

Полупроводниковые ВЗУ

Развитие полупроводниковых ВЗУ (электронных дисков) определяется быстрым ростом степени интеграции полупроводниковых БИС и, как следствие, снижением стоимости полупроводниковой памяти в расчёте на 1 бит. В то же время повышение разрядности ЭВМ (до 32) обеспечивает прямую адресацию к полупроводниковой внешней памяти как физическому расширению ОЗУ, рассматриваемому операционной системой чаще всего в качестве НГМД.

Электронный диск выступает в качестве логического диска и реализуется с помощью необходимого числа плат ОЗУ и диспетчера памяти. Конфигурация ПК с электронным диском позволяет существенно повысить эффективность использования компьютера, время жизни дискет и магнитных головок НГМД (в десятки раз), а также сократить время передачи данных в 8-10 раз. Например, модель ST868K фирмы Seagate имеет следующие параметры: информационная емкость от 67 до 402 Мбайт, время доступа не более 0,1 мс и скорость передачи данных от 2,5 до 10 Мбит/с.

Интерфейс для накопителей

Для связи накопителей информации с компьютером разработаны специальные интерфейсы. В настоящее время преобладают три вида интерфейса: IDE, SCSI и USB (табл.3.3.2).

Таблица 3.3.2

Интерфейс SCSI

Интерфейс	*Разрядность шины данных*	*Тактовая частота, МГц*	*Максимальная скорость передачи данных, Мбайт/с*	*Число подключаемых устройств*
SCSI	8	5	5	8
SCSI-2	8, 16 число информационных линий 24	10	10	8
Ultra SCSI	16	20	40	8
Ultra 2 SCSI	16	40	40	15

SCSI

Данный стандарт имеет несколько модификаций отличающихся числом используемых линий, числом подключаемых устройств, тактовой частотой (табл. 3.3.2). Данный интерфейс имеет собственный контроллер и собственную BIOS, которая управляет шиной, освобождая центральный процессор.

IDE (ATA)

Данный интерфейс использовался для подключения накопителей на жестком диске. Он позволяет подключать два дисковода. Поддерживаст два способа обмена данными: через центральный процессор; обмен данными с ОЗУ без участия центрального процессора. Максимальная скорость передачи данных 8,3 Мбайт/с.

Развитием этого стандарта стал интерфейс *EIDE*, который поддерживает устройства объемом свыше 504 Мбайт. Позволяет подключать четыре устройства. В системе можно устанавливать несколько контроллеров EIDE. Максимальная скорость передачи данных – до 16,7 Мбайт/с.

Интерфейс ATA-3 разработан для повышения надежности передачи данных и повышения ее производительности. Стандарт поддерживает технологию предупреждения отказов жестких дисков *SMART*.

Интерфейс Ultra ATA поддерживает протокол *Ultra DMA*, по которому за один такт передается в два раза больше информации, чем в предыдущих версиях.

Интерфейс USB характерен тем, что позволяет подключать последовательно несколько устройств.

Виртуальная память

В ЭВМ существует также понятие виртуальной памяти. Виртуальная память - это фиктивная память. Способностью работать с виртуальной памятью обладают все процессоры, начиная с микропроцессора *Intel 80286*. Кроме того, для этого требуется соответствующая программа операционной системы *VDISK*.

Служебная программа виртуальной памяти операционной системы *VDISK* посылает команду микропроцессору сопоставить физической памяти, предназначенной для программы пользователя, виртуальные адреса, которые будут использоваться программой. Такое свойство микропроцессора, как управлять памятью, заставляет реальную память машины иметь рабочий адрес, отличающийся от истинного, физического адреса.

Программа начинает работу с отображения некоторой части большего объёма виртуальной памяти в определённую часть меньшей по объему физической памяти компьютера. Пока программа работает лишь с частью своей виртуальной памяти, она этого не замечает. Программа на самом деле использует другие адреса памяти, но это для неё не имеет никакого значения. Когда программа пытается использовать ту, большую часть виртуальной памяти, которой не было отведено место в реальной памяти, меньшей по объёму, таблица управления памятью микропроцессора обнаруживает, что программа пытается использовать несуществующий в данный момент адрес и выдаёт команду "отсутствие страницы". Получив эту команду, специальная служебная программа виртуальной памяти начинает работу. Она временно приостанавливает действие программы, выбирает ту часть виртуальной памяти, которая находится в данный момент в физической памяти компьютера и записывает её на диск (это называется *выгрузкой* или *откачкой*). Освобожденная часть реальной памяти начинает действовать как необходимая часть виртуальной памяти. Когда же возникает необходимость в выгруженной части памяти, она подкачивается назад, то есть переписывается с диска в ОЗУ.

Как следует из описанного, диск компьютера используется в качестве склада для хранения частей виртуальной памяти, которые не используются в данный момент.

3.3.4. Клавиатура

В настоящее время в эксплуатации находится значительное число типов клавиатуры, но все они построены по одному принципу. На клавиатуре можно выделить четыре поля (рис.3.3.4):

1) поле с функциональными и управляющими клавишами;

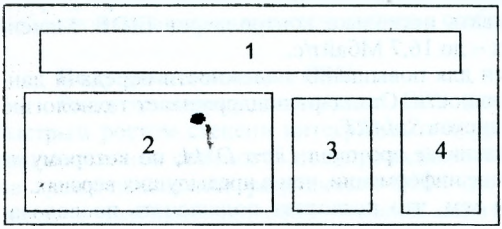


Рис 3.3.4. Поля типовой клавиатуры

2) основное (центральное) поле, где расположены клавиши с буквами русского и латинского алфавита, цифрами, специальными символами и некоторые управляющие клавиши;

3) поле с клавишами для управления курсором и редактирования текста;

4) дополнительное поле для ввода цифровой или символьной информации, а также для управления курсором.

Назначение клавиш приведено в табл. 3.3.3.

Обозначение вида [Ctrl – F1] означает: нажать первую клавишу и, не отпуская ее, нажать вторую клавишу.

Клавиши первого ряда основного поля имеют два ряда символов. Символы нижнего ряда вводятся непосредственно, символы верхнего ряда - при нажатии клавиши Shift. На клавиатуре зарубежных компьютеров переключение на ввод символов русского алфавита выполняется специальной программой-драйвером клавиатуры. Эта программа запускается, как правило, в начале работы с компьютером и затем постоянно находится в памяти ПК. Способы переключения на ввод русских символов зависят при этом от используемого драйвера, чаще всего применяются комбинации клавиш [Shift-Shift], [Ctrl-Alt], [Ctrl – Shift правый], [Ctrl - Shift левый], а также клавиша Caps Lock.

Комбинации клавиш, часто используемых при работе на ПК:

[Ctrl-Alt-Del] - перезагрузка дисковой операционной системы (горячий перезапуск);

[Ctrl-PrScr] или [Ctrl-P] - включение/выключение режима копирования на принтер информации, выводимой на экран монитора;

[Ctrl-NumLock] - приостанавливает выполнение программы, для продолжения выполнения нажать любую клавишу.

[Ctrl-S] - приостанавливает выполнение программы;

[Ctrl-Break] - завершение работы выполняемой программы или команды;

Ctrl-C] - прекращает выполнение любой команды или программы операционной системы.

[Shift-PrScr] - печать графической копии экрана.

Таблица 3.3.3

Назначение клавиш на клавиатуре IBM PC/AT

Клавиатура IBM PC/AT	*Назначение клавиш*
F1...F12	Функциональные клавиши
Caps Lock	Фиксация прописных букв
⇧ Shift	Смена регистра. Переключение на ввод прописных букв без фиксации или на ввод спецсимволов
⇄ Tab	Табуляция, перемещение курсора на фиксированное число позиций
Ctrl	Ввод дополнительного кода. Используется в комбинации с другими клавишами
Alt	Альтернативный набор. Используется в комбинации с другими клавишами, а также для ввода символов с кодами 128-255
← Backspace	Возврат на символ. Стирает последний введенный символ
Spacebar	Ввод пробела
Enter	Конец ввода команды, набора текста
Esc	Выход в вызвавшую программу, отказ от операции
Print Screen	Печать экрана
Scroll Lock	Блокировка прокрутки
Pause/Break	Пауза, приостанавливает вывод файла на экран. Для продолжения просмотра нажать любую клавишу
Num Lock	Переключение дополнительного поля на ввод цифр или управляющих символов
Ins	Включение/выключение режима вставки символов
Del	Удаление символа в позиции курсора
End	Переместить курсор в конец файла
Home	Переместить курсор в начало файла
PgUp	Переместить курсор на страницу вверх
PgDn	Переместить курсор на страницу вниз
← ↑ → ↓	Направление перемещения курсора

3.3.5. Видеомонитор

Видеомонитор, наряду с клавиатурой, является основным устройством, обеспечивающим общение пользователя с компьютером. На его экран выводится вся информация и текстовая, и графическая.

В настоящее время преобладают мониторы двух типов: мониторы на основе электроннолучевой трубки (CRT-мониторы) и жидкокристаллические мониторы (LCD-мониторы).

CRT- мониторы

Основу CRT-мониторов составляет электронно-лучевая трубка (ЭЛТ). Упрощенная структура ЭЛТ приведена на рис. 3.3.5. Она состоит из стеклянной

колбы (1), трех электронных пушек (2), системы сведения и отклонения луча (3), маски (4), экрана (5). С внутренней стороны экран монитора покрыт слоем люминофора – вещества способного светиться при бомбардировке его заряженными частицами.

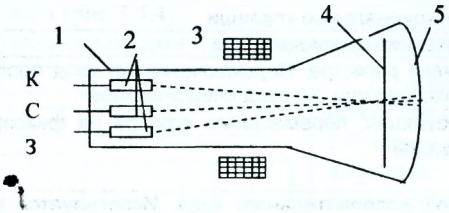


Рис. 3.3.5. Электронно-лучевая трубка

Электронные пушки предназначены для излучения и фокусировки электронов на экране видеомонитора. В цветных мониторах имеется три электронных пушки. Отклоняющая система служит для управления электронным лучом. В состав отклоняющей системы входит генератор строчной развертки, перемещающий луч в горизонтальном направлении, и генератор кадровой развертки, перемещающий луч в вертикальном направлении. В процессе работы луч пробегает по экрану по строкам, начиная с верхнего левого края до правого нижнего края, и создает изображение на экране. Во время обратного хода изображение на экране не меняется, а луч перемещается снова в левый верхний угол экрана. Экран в цветных мониторах состоит из точек трех цветов, расположенных группами по три (триады): красного, синего и зеленого. Каждая из этих точек освещается своим лучом. От интенсивности луча зависит яркость ее свечения.

Эта триада зерен люминофора образует точку на экране - *пиксел*. При воспроизведении изображения точки могут группироваться как по вертикали, так и по горизонтали, образуя более крупные точки.

Наш глаз воспринимает эти три точки как одну точку с определенным цветом. Для обеспечения точного попадания лучей на одну триаду перед экраном устанавливают маски. От технологии изготовления маски, а также от качества электронной схемы управления сведением лучей зависит качество ЭЛТ.

Основными характеристиками мониторов, определяющих их потребительские качества, являются четкость, частота строчной и кадровой развертки, размер экрана, разрешающая способность, емкость видеопамяти, глубина цвета.

Четкость изображения зависит от *шага точки* – максимального расстояние между люминофорными точками одинакового цвета. Эта величина находится в интервале 0,28 – 0,24 мм. На практике этот параметр называют часто размером зерна. Чем меньше шаг точки, тем выше качество изображения при прочих равных условиях.

Размер экрана. Размер экрана – это размер экрана монитора по диагонали, его принято измерять в дюймах. Установилось четыре основных стандарта размеров экрана для бытовых компьютеров: 15", 17", 19", 21".

Разрешающая способность определяется числом точек на экране по горизонтали и по вертикали. Современные мониторы позволяют получать различные значения разрешающей способности: 320x200; 640x480, 800x600, 1024x768, 1280x1024, 1600X1200 и др. При работе в операционной системе Windows предпочтительное значение разрешающей способности для комфортной работы зависит от размера экрана монитора (табл. 3.3.4).

Частота строчной развертки и частота кадровой развертки взаимосвязаны. Они зависят также от размера монитора и установленного разрешения экрана. Частота строчной развертки должна обеспечить воспроизведение требуемого количества линий на экране за время прямого хода кадровой развертки. Например, при разрешении экрана 1280x1024 частота строчной развертки должна быть не менее 70 КГц. Частота кадровой развертки непосредственно влияет на качество изображения, комфортность работы, утомляемость пользователя. При низкой частоте кадровой развертки становится заметным мерцание экрана. Рекомендуемые значения кадровой развертки – 75-100 Гц (не все мониторы поддерживают частоту 100 Гц). Установлено, что при частоте 110 Гц глаз человека не может заметить никакого мерцания. Установлено также, что при больших углах обзора мерцание заметнее при больших частотах кадров.

Таблица 3.3.4

Рекомендуемые значения разрешающей способности монитора

Размер экрана	***Разрешающая способность***					
	800x600	1024x768	1152x864	1280x960	1280x1024	1600x1200
15"	+	+				
17"		+	+	+		
19"			+	+	+	
21"					+	+

При воспроизведении текста экран разбит условно на строки и столбцы. В обычном режиме на экране размещается 25 строк и 80 столбцов. Нумерация строк и столбцов начинается с нуля. Для каждого символа отводится одно знакоместо. Имеется возможность программно изменять число строк и столбцов на экране.

Глубина цвета или количество воспроизводимых одновременно цветов.

Современные мониторы типа SVGA способны воспроизводить практически неограниченное число цветов. Чаще всего устанавливаются палитры из 64, 256 цветов, а также 16-и и 24-х битные палитры. Теоретически, выделив для управления каждым цветом (синим, зеленым и красным) один байт, можно получить 256x256x256 различных цветов и оттенков. Число воспроизводимых цветов ограничивается практически только размером видеопамяти. Например, при разрешении 640x480 и 256 цветах для хранения информации о состоянии экрана необходимо 307200 байт видеопамяти, при разрешении 1200x1024 и 256 цветах потребуется уже 1228800 байт видеопамяти

Видеопамять. Видеопамять размещается в видеоадаптере. Она организуется в виде страниц. Видеомонитор имеет, как минимум, две страницы памяти: одна страница - видимая, отображается на экране, другая страница в это время заполняется (во время обратного хода луча). Для качественного воспроизведе-

ния мультипликации компьютер должен иметь 4 страницы видеопамяти. Емкость видеопамяти современных компьютеров составляет 4, 8, 16 Мбайт.

Существенными недостатками CRT-мониторов являются большое потребление электрической энергии и наличие излучений. Для обеспечения безопасной эксплуатации мониторов к их изготовлению предъявляют высокие требования. Эти требования изложены в трех стандартах: TCO 92, TCO 95 и TCO 99, - цифры означают год принятия стандарта. Эти стандарты устанавливают требования по максимально допустимому значению электромагнитных излучений, шума и тепла при работе мониторов, функции энергосбережения монитора, эргономике, экологии, пожарной безопасности и электробезопасности.

LCD – мониторы

Принцип работы LCD-мониторов основан на свойстве некоторых веществ, находящихся в жидком состоянии и обладающих свойствами, присущими кристаллическим телам, изменять плоскость поляризации света, проходящего через жидкость или отражающегося от нее, при воздействии на нее электрического поля.

Экран в жидкокристаллических мониторах представляет собой две прозрачные пластины, пространство между которыми заполнено жидкими кристаллами. Перед передним экраном установлены два поляризационных фильтра, чтобы глаз человека мог различить изменения в поляризации светового потока. Экран разделен на отдельные ячейки, к которым подведены электроды, создающие электрическое поле. Для вывода цветного изображения делается подсветка монитора сзади так, чтобы свет порождался в задней части LCD-монитора. Цвет получается в результате использования трех фильтров, которые выделяют из излучения источника белого цвета трех основных цветов. Комбинируя три основные цвета для каждой точки, можно воспроизвести любой цвет.

Разрешение LCD-мониторов одно, оно соответствует максимальному физическому разрешению CRT-мониторов. Несомненными достоинствами LCD-мониторов являются низкое потребление электрической энергии и отсутствие вредных излучений.

LCD-мониторы изготовляются по различным технологиям, что сказывается на их характеристиках.

Другие технологии изготовления мониторов

Плазменные мониторы PDP

Принцип работы плазменных мониторов основан на явлениях электрического разряда в газах. Плазменные экраны создаются путем заполнения пространства между двумя стеклянными поверхностями инертным газом, например аргоном или неоном. На стеклянную поверхность помещают маленькие прозрачные электроды, на которые подают высокое напряжение. Под воздействием этого напряжения в прилегающей к электроду газовой области возникает электрический разряд. Плазма газового разряда излучает свет в ультрафиолето-

вом диапазоне, который вызывает свечение частиц люминофора в диапазоне, видимом человеком. Каждый пиксел работает при этом как обычная флуоресцентная лампа (лампа дневного света). Высокая яркость и контрастность, отсутствие дрожания изображения является достоинством этих мониторов.

FED-мониторы

Принцип работы FED-мониторов похож на принцип работы CRT-мониторов – свечение люминофора при бомбардировке электронами. В FED-мониторах каждый пиксел представляет собой, по существу, миниатюрную электронную трубку. За каждым элементом экрана располагается источник электронов. Каждый источник электронов управляется отдельным электронным элементом, и каждый пиксел затем излучает свет благодаря воздействию электронов на люминофорные элементы, как в традиционных CRT-мониторах. Основное достоинство FED-мониторов – очень малая толщина.

Видеоадаптер

Видеоадаптер представляет собой, специальное устройство, сконструированное в виде отдельной платы расширения. Он управляет выводом информации на экран монитора. Характеристики видеосистемы зависят как от параметров используемого монитора, так и от установленного в компьютере видеоадаптера. В качестве примера используемых адаптеров можно привести видеоадаптер XGA – расширенный графический массив фирмы IBM, QVision 1024E EISA фирмы Compag, Tseng фирмы ATI. На плате видеоадаптера размещаются и микросхемы видеопамяти.

3.3.6. Печатающие устройства

Печатающие устройства (ПУ) или, как их ещё называют, принтеры предназначены для получения "твёрдой копии" документа, который мог бы храниться длительное время и был удобен для чтения человеком.

Все печатающие устройства могут выводить текстовую информацию, многие из них могут выводить также графики и рисунки, в том числе и цветные.

Существует множество моделей принтеров, но наибольшее практическое применение получили матричные, струйные и лазерные принтеры.

Матричные принтеры являются наиболее распространённым типом принтеров для IPM ПЭВМ. Принцип печати этих принтеров следующий: печатающая головка перемещается относительно неподвижного листа бумаги вдоль строки и наносит текст или рисунок с помощью игл, ударяющих по красящей ленте. Печатающая головка содержит 9, 24 или 48 игл. Скорость печати матричного принтера от 10 до 60 с на страницу.

Струйные принтеры обеспечивают изображение на бумаге с помощью сопел, через которые выдуваются мельчайшие капельки чернил. Этот способ печати даёт более высокое качество печати по сравнению с матричными прин-

терами и очень удобен для цветной печати. Струйные принтеры содержат 24 или 64 сопла. Скорость печати у них в несколько раз выше, чем у матричных принтеров. Основной недостаток струйных принтеров заключается в том, что при случайном попадании влаги на готовый отпечаток он размазывается.

Лазерные принтеры обеспечивают в настоящее время наилучшее (близкое к типографскому) качество печати. В них используется электрографический принцип создания изображения. Основными элементам лазерного принтера являются специальный барабан, источник лазерного излучения, электронная схема управления и сухой порошок – *тонер.* Тонер представляет собой кусочки железа, покрытые пластиком. Принцип работы лазерного принтера заключается в следующем. Изображение формируется построчно. Полупроводниковый барабан предварительно электризуется. Затем под управлением электронной схемы луч лазера пробегает по барабану. При наличии информации луч включается, при отсутствии – выключается. Там где луч лазера коснулся барабана, электрический заряд стекает с барабана. Таким образом на поверхности барабана оказывается нанесенным скрытое изображение. На следующем этапе осуществляется "проявление" изображения – частички тонера притягиваются к тем участкам барабана, которых коснулся луч лазера, под действием сил электростатического взаимодействия зарядов разного знака. Когда видимое изображение на барабане построено, то есть он покрыт тонером в соответствии с изображением оригинала, лист бумаги заряжается таким образом, что частички тонера с барабана притягиваются к нему. Прилипший порошок закрепляется на бумаге за счет нагрева частиц тонера до температуры плавления пластмассы, покрывающей частицы железа. В результате этих операций формируется несмываемый водоупорный отпечаток.

Цветные лазерные принтеры формируют изображение, последовательно накладывая голубой, пурпурный, желтый и черный тонеры на фоточувствительный барабан. Принтер работает в четырехпроходном режиме, поэтому скорость печати цветного принтера существенно меньше скорости печати черно-белого принтера.

К потребительским качествам принтеров относятся следующие показатели:

качество и скорость печати;

наличие русского шрифта в ПЗУ или возможность его загрузки из программы компьютера;

надёжность;

количество шрифтов, которые поддерживает принтер;

смена красящего элемента;

совместимость с имеющимися программами по управляющим кодам.

3.3.7. Дополнительные устройства компьютера

Дополнительные устройства обеспечивают ввод в компьютер и вывод из него текстовой и графической информации.

Мышь служит для управления перемещением курсора по экрану, выбора пунктов меню, прокрутки экрана, запуска исполняемых файлов. Она имеет шарик, встроенный в дно корпуса, две или три клавиши. При перемещении мыши по ровной поверхности программа считывает положение шарика и преобразует его

координаты в положение указателя мыши на экране. Нажатие клавиш фиксируется программой и преобразуется в команды. Некоторые прикладные программы работают только с манипулятором мышь, но большинство программ могут работать как с манипулятором, так и с клавиатурой компьютера. В графических операционных системах манипулятор мышь является основным средством управления компьютером, а клавиатура выполняет вспомогательную роль.

Джойстик - это специальное устройство для ввода информации в компьютер. Он представляет собой стержень (ручку), свободно перемещаемый в двух измерениях, и две кнопки - переключателя. Программа считывает положение стержня в виде двух координат X и У и преобразует их в положение курсора на экране. Нажатие кнопок-переключателей также может быть зафиксировано программой. Используется джойстик, чаще всего, как манипулятор в игровых программах.

Световое перо предназначено для считывания информации с экрана дисплея, а именно координат точки экрана, к которой оно прикасается. Световое перо используется, чаще всего, в системах автоматизированного проектирования. При использовании светового пера программа позволяет "рисовать" на экране, "брать" и перемещать части рисунка по экрану, производить выбор из меню, прикасаясь пером к элементу данных и так далее.

Графопостроитель (плоттер) представляет устройство для вывода чертежей на бумагу. Графопостроители бывают барабанного и планшетного типа, используются, как правило, в системах автоматизированного проектирования (САПР) для вывода графических изображений.

Графический планшет -- это устройство для ввода контурных изображений в ЭВМ. Используется в САПР для ввода чертежей в компьютер.

Сканер (читающий автомат) – устройство, предназначенное для считывания графической и текстовой информации в компьютер. Сканеры, при наличии соответствующего программного обеспечения, могут вводить в компьютер рисунки, а также распознавать символы, что позволяет быстро вводить напечатанный (а иногда и рукописный) текст в ПК. Сканеры бывают настольные и ручные. Настольные сканеры обрабатывают весь лист бумаги целиком. Ручные сканеры необходимо проводить над нужным текстом или рисунком вручную.

Дигитайзер – устройство для "оцифровки" изображений. Позволяет преобразовывать изображения в цифровую форму для обработки в компьютере. Используется в системах обработки изображений.

Сетевой адаптер - используется для подключения компьютера в локальную вычислительную сеть. При этом пользователь получает доступ к данным, находящимся на другом компьютере.

Контрольные вопросы

1. Назовите основные классификационные признаки ЭВМ.
2. Приведите структурную схему ЭВМ и поясните принцип ее работы.
3. Приведите основные характеристики персональных компьютеров.
4. Поясните назначение и основные типы микропроцессоров.

5. Какие технические приемы и технологии применяются для повышения быстродействия процессоров?

6. Приведите классификацию памяти современного компьютера и поясните ее.

7. В чем заключается принцип работы накопителей на магнитных дисках?

8. В чем заключается принцип работы накопителей на оптических дисках?

9. Назовите условные поля клавиатуры и поясните их назначение.

10. Поясните принцип работы мониторов на электронно-лучевых трубках (CRT-мониторов).

11. Поясните принцип работы матричных печатающих устройств.

12. Поясните принцип работы лазерных печатающих устройств.

4. Программное обеспечение компьютера

4.1. Классификация программного обеспечения

Программным обеспечением (ПО) называется комплекс программ, описаний и инструкций, позволяющих автоматизировать отладку программ и решение задач на ЭВМ.

В зависимости от назначения программное обеспечение ПК делится на две большие группы: базовое и прикладное.

Базовое программное обеспечение

Базовое программное обеспечение включает комплекс программ, обеспечивающих управление ресурсами вычислительной системы, подготовку программ и техническое обслуживание компьютера и делится на четыре группы:

- операционная система (ОС) и ее окружение:
- системы программирования;
- средства контроля и диагностики;
- средства телекоммуникации и сетевой поддержки.

Основными компонентами ОС являются управляющие программы, программы интерпретаторы, файловая система, утилиты.

Управляющая программа планирует ресурсы ПК, обеспечивает взаимодействие его с внешней средой, оперативно контролирует исправность технических средств и организует восстановление процессов обработки данных после появления неисправности.

Интерпретаторы обеспечивают диалог пользователей с системой, воспринимают и расшифровывают (интерпретируют) его команды, обеспечивают их выполнение.

Файловая система представляет совокупность средств ОС, обеспечивающих выполнение операции поиска и ввода/вывода данных, создания, копирования, удаления, переименования файлов, обработку прерываний.

Утилиты - специальные сервисные программы, расширяющие возможности ОС или облегчающие использование ее команд с помощью программных

оболочек. Примерами таких программ являются нортоновские утилиты, программы *Norton Commander*, *Volkov Commander*, *CHKDSK* и другие.

Системы программирования образуют языки программирования, трансляторы, компиляторы, интерпретаторы программ, представленных на этих языках, соответствующая программная документация, а также вспомогательные средства для подготовки программ к выполнению.

Средства контроля и диагностики предназначены для обеспечения процедур контроля и диагностики неисправностей, проверки и восстановления работоспособности системы. После включения ПК обычно сразу же запускается программа проверки состояния внешних устройств, оперативной памяти и микропроцессора. В случае обнаружения неисправности на экран видеомонитора выдается сообщение о характере обнаруженной неисправности, рекомендации по дальнейшей работе. Вместе с компьютером поставляется обычно на дискетах программа диагностики компьютера и соответствующее руководство.

Средства телекоммуникации и сетевой поддержки обеспечивают возможность передачи данных между компьютерами, объединение их в локальные или глобальные вычислительные сети.

Прикладное программное обеспечение

Прикладное программное обеспечение включает пакеты прикладных программ (ППП) и специальные программные продукты (оригинальные программы).

В связи с широким распространением в производстве и в быту персональных ЭВМ постоянно расширяется и класс задач, решаемых на них, следовательно, и программ. Рынок программного обеспечения для персональных ЭВМ содержит сотни наименований прикладных программ различного назначения. Существуют программы для решения задач узкого класса, например, проектирование печатных плат, или широкого класса, например, редактирование текстов. Пишутся программы, как правило, программистами высокой квалификации, но могут разрабатываться и пользователями. Несмотря на сравнительно короткую историю развития программного обеспечения, уже возникли устойчивые классы программ для персональных компьютеров, объединенные общей прикладной направленностью и единым интерфейсом с пользователем. Возможности ПК во многом связаны с наличием или отсутствием необходимого программного обеспечения. Не будет программ — дорогостоящая вычислительная машина станет украшением интерьера, будут необходимые прикладные программы — станет Вашим надежным помощником.

Ориентация ПК на конкретные области применения осуществляется с помощью прикладного программного обеспечения. В настоящее время выработана следующая классификация прикладного программного обеспечения: по способу реализации и принципам функционирования, по области применения и классам решаемых задач, по ориентации на определенный метод или процедуру обработки данных.

По способу реализации и принципам функционирования различают библиотеки прикладных программ (БПП), пакеты прикладных программ (ППП), интегрированные пакеты.

По области применения и классам решаемых задач прикладное программное обеспечение делится на программы, расширяющие возможности

операционных систем, общего назначения, специального применения, для решения инженерных и научно-технических задач, для решения экономических задач и задач автоматизированных систем управления.

По ориентации на определенный метод или процедуру обработки программы делятся на методо-ориентированные, проблемно-ориентированные и технологически ориентированные.

Библиотеки прикладных программ являются простейшей формой организации программного обеспечения и представляют собой набор программ в совокупности с системой их вызова, хранения и настройки на соответствующие параметры, предназначенные для решения задач определенного класса. Программы, входящие в библиотеку, написаны, как правило, на одном языке программирования.

Пакеты прикладных программ (ППП) — более совершенная форма организации программного обеспечения. ППП — это комплекс взаимосвязанных программ и системных средств, работающих под управлением головной (организующей) программы пакета - программы монитора. ППП ориентирован, как правило, на решение определенного класса задач, близких по содержанию или применяемым математическим методам, и в общем виде состоит их трех основных компонентов: функционального наполнения, языка заданий и системного наполнения.

Функциональное наполнение — совокупность модулей (конструктивных элементов), используемых на разных этапах работы пакета. Состав этой части отражает специфику предметной области пакета, полноту покрытия этой предметной области, объем прикладных знаний, заложенных в пакет.

Язык заданий — средство общения пользователя с пакетом. На языке заданий пользователь описывает порядок выполнения операций, обслуживающих решение задачи или постановку задачи, по которой эта последовательность строится автоматически. Предметная область и технология программирования, принятые в пакете, определяют набор операций и синтаксис языка заданий. Через язык заданий пользователь оценивает, каковы вычислительные возможности ППП и насколько удобно их использование.

Системное наполнение — комплекс программ, которые обеспечивают взаимодействие пользователя с пакетом и автоматизацию выполнения задания.

Интегрированные пакеты — это более высокий уровень организации программного обеспечения, вызванный расширением технических возможностей ПК. В отличие от наборов ППП, не связанных между собой, интегрированный пакет представляет собой единый программный комплекс с возможностью обмена данными между его компонентами. В интегрированной операционной среде вместо сложной системы различных режимов и командных строк реализуется простое и эффективное взаимодействие с ПК с использованием перекрывающихся окон и манипулятора "мышь". При работе в интегрированной среде не требуется обращение к другим программам. Но работа с интегрированными пакетами требует наличия значительного объема оперативной и внешней памяти компьютера.

К пакетам прикладных программ общего назначения относятся редакторы текстов (текстовые процессоры), электронные таблицы (табличные процессоры), системы управления базами данных (СУБД), графические редакторы, интегрированные системы.

Редакторы текста позволяют создавать на персональном компьютере различного рода документы, печатать их в заданном формате и требуемом количестве экземпляров. Редакторы текстов отличаются по сложности и объему выполняемых функций. Одни используются для подготовки небольших текстов и документов, другие позволяют обрабатывать документы больших объемов и готовить их к изданию с помощью настольных издательских систем.

Электронные таблицы позволяют решать широкий круг научно-технических, планово-экономических, учетных и других задач, для которых исходные данные и результаты могут быть представлены в табличной форме. При этом обеспечивается хранение в памяти компьютера и просмотр на экране дисплея таблиц большого размера.

Системы управления базами данных. Эта группа средств позволяет создавать на базе персонального компьютера автоматизированные информационно-справочные системы различного назначения, обеспечивающие хранение во внешней памяти компьютера на магнитных дисках большого объема информации, поиск и выборку этой информации по запросам, оформление выходных данных в виде документов и отчетов определенной формы.

Графические редакторы обеспечивают вывод на экран дисплея графических изображений при наличии в составе компьютера графического дисплея. Они позволяют наглядно отображать результаты вычислений в виде круговых, линейных, столбиковых и иных диаграмм, а также графиков функций.

Интегрированные системы объединяют функции многих из перечисленных выше систем. Характерной особенностью интегрированных систем является многооконный интерфейс, позволяющий отображать, например, в одном окне данные, выбираемые из базы данных, во втором окне — данные, записанные на электронном бланке, в третьем — графическое представление данных и так далее. Можно обмениваться данными между программами, вызванными в разные окна. Интегрированные системы предоставляют также специальный язык, с помощью которого можно запрограммировать работу прикладной системы обработки данных.

Методо-ориентированные ППП предназначены для решения различных научных и инженерных задач, реализуя определенные методы вычислительной математики, в том числе задач математической статистики, сетевого планирования и управления, многокритериальной оптимизации и так далее.

Проблемно-ориентированные ППП предназначены для решения конкретных задач в различных областях применения и создания на базе персональных компьютеров автоматизированных рабочих мест (АРМ) для различных видов профессиональной деятельности.

4.2. Операционная система

4.2.1. Классификация операционных систем

Операционная система — это совокупность программ, управляющих функционированием всех компонентов компьютера.

Операционная система обеспечивает:

– автоматический запуск в работу вычислительной системы с проверкой аппаратных и программных средств;

– распределение ресурсов системы в соответствии с решаемой задачей и управление работой процессора, памяти, устройств ввода/вывода;

– запуск на выполнение прикладных программ и их взаимодействие с аппаратными средствами;

– обработку прерываний (запросов программ пользователя);

– создание и ведение каталогов, организацию и управление файлами;

– управление аппаратными средствами;

– диалог с пользователем о ходе обработки информации и работе аппаратных средств.

Операционные системы можно классифицировать по ряду признаков: количеству одновременно работающих пользователей, числу одновременно решаемых задач, количеству используемых процессоров, типу пользовательского интерфейса, способу использования общих аппаратных средств и программных ресурсов, разрядности.

По количеству одновременно работающих пользователей ОС делятся на однопользовательские и многопользовательские:

– в *однопользовательских* ОС все ресурсы вычислительной системы находятся в распоряжении одного пользователя;

– в *многопользовательских* ОС ресурсы (в основном процессорное время) распределяются между несколькими пользователями по определенным правилам. При этом, ввиду высокого быстродействия процессора, пользователю, работающему на *рабочей станции* или *терминале,* кажется, что процессор полностью находится в его распоряжении. И только при выполнении длительных операций, связанных, например, с вводом/выводом информации, пользователь может заметить замедление в работе.

По числу задач одновременно выполняемых под управлением ОС они делятся на однозадачные и многозадачные:

– *однозадачные ОС* предоставляют пользователю виртуальную машину и включают средства управления файлами, периферийными устройствами и средствами общения с пользователями. К выполнению новой задачи ОС может приступить только после завершения выполнения текущей задачи;

– *многозадачные ОС* дополнительно управляют разделением между задачами совместно используемых ресурсов. При этом, в зависимости от способа управления распределением процессорного времени, различают ОС с невытесняющей и вытесняющей многозадачностью. *Невытесняющая* многозадачность (Windows 3.x, Windows 9x, NetWare) характеризуется тем, что активный процесс выполняется до тех пор, пока он сам, по собственной инициативе, не передаст управление операционной системе. При *вытесняющей* многозадачности (Windows NT, OS/2, UNIX) решение о переключении процессора с одного процесса на другой принимается операционной системой, а не самим процессом. Многозадачные ОС позволяют, например, редактировать текст и выводить на печать другую программу в фоновом режиме, включить для комфорта музыку и работать в графическом редакторе или в электронной таблице и т. п.

Одним из важнейших свойств ОС является возможность распараллеливания вычислений в рамках одной задачи. Это свойство получило название *многонитевидность.* Многонитевая ОС распределяет время не между задачами, а между отдельными ветвями (нитями) текущей задачи.

По количеству используемых процессоров ОС делятся на *однопроцессорные* и *многопроцессорные*. Алгоритмы работы однопроцессорных ОС соот-

ветствуют принципам работы ЭВМ фон-неймановской архитектуры. *Мультипроцессирование* является другой важной характеристикой ОС. Мультипроцессирование приводит к усложнению всех алгоритмов управления ресурсами.

По типу пользовательского интерфейса ОС делятся на *командные* (текстовые) и *объектно-ориентированные* (графические). В настоящее время в связи с ростом быстродействия процессоров и объемов оперативной и дисковой памяти преимущественное распространение имеют графические ОС, так как они предоставляют пользователю значительно более удобный интерфейс и средства управления компьютером с помощью мыши.

По способу использования общих аппаратных средств и программных ресурсов программы ОС делятся на сетевые и локальные.

Локальные ОС обеспечивают управление только данным компьютером (MS-DOS).

Сетевые ОС обеспечивают работу компьютера в локальных и глобальных вычислительных сетях. Сетевые ОС предназначены для эффективного решения задач распределенной обработки данных. Такая обработка ведется не на отдельном компьютере, а на нескольких компьютерах, объединенных сетью. Сетевые ОС поддерживают распределенное выполнение процессов, их взаимодействие, обмен данными между ЭВМ, доступ пользователей к общим ресурсам и ряд других функций. Все сетевые ОС делятся на две группы: одноранговые ОС и ОС с выделенным сервером. В *одноранговых* сетях каждая ЭВМ может выполнять как функции сервера, так и рабочей станции. В *сетях с выделенным сервером* рабочие станции не предоставляют свои ресурсы другим ЭВМ.

По разрядности ОС делятся на 8, 16, 32 и 64 разрядные.

4.2.2. Краткий обзор операционных систем

Наиболее известны и широко используются на персональных компьютерах операционные системы MS-DOS и Windows, Unix, OS/2.

Операционная система ***MS-DOS*** фирмы Microsoft и ее аналоги других фирм применялись на 16 разрядных ПК. Это локальная однопользовательская, однозадачная, командная операционная система. Известно много версий этой операционной системы. Элементы операционной системы MS-DOS входят в состав операционной системы Windows 95/98. С появлением графических операционных систем дисковые операционные системы утратили свое былое значение.

Операционная система ***Windows*** является в настоящее время лидером среди ОС персональных компьютеров. Начало этом семейству ОС положила *Windows 3.1*, позднее появились другие версии этой системы с нумерацией, соответствующей году выпуска: *Windows 95, Windows 98*. В литературе эти операционные системы часто обозначают Windows 9x. Эти операционные системы являются объектно-ориентированными, локальными, однопользовательскими, многозадачными. Одна из версий программы Windows 3.1 - *Windows for Workgroups 3.11* была предназначена для построения одноранговых сетей, не требующих установки специального компьютера в качестве сервера.

В 1999 году была выпущена ОС *Windows Millennium*, в которой получили дальнейшее развитие технологии, заложенные в операционных системах Windows 9x..

В 1993 году было положено начало выпуску ОС серии *Windows NT*, которые являются сетевыми операционными системами. Имеется две модификации ОС *Windows NT*. Одна предназначена для рабочих станций, другая ориентирована на

использование в качестве выделенного сервера. Windows NT - это многопроцессорная, многозадачная, 32 разрядная операционная система, в ней предусмотрена высокая степень защиты от несанкционированного доступа и случайного повреждения. Позже были выпущены ОС *Windows NT 3.5*, *Windows NT 4.0*.

Windows NT 5.0 или Windows 2000 – сетевая, 32-разрядная, многопроцессорная ОС с приоритетной многозадачностью, имеет средства обеспечения надежности, защиты и управления. Windows 2000 выпускается в четырех вариантах: Windows 2000 Professional (NT Workstation), Windows 2000 Server (NT Server), Windows 2000 Advanced Server (NT Server Enterprise Edition), Windows –2000 DataCenter Server. Эти варианты отличаются количеством входящих в поставку служб и программ, степенью поддержки аппаратного обеспечения. Так Windows 2000 Professional поддерживает до двух процессоров, Server – до четырех процессоров, Advanced Server – восемь процессоров, а DataCenter поддерживает шестьдесят четыре процессора.

Windows 2000 Professional является базовой ОС для настольных и мобильных компьютеров. Она поддерживает файловые системы NTFS, FAT16 и FAT32. Обеспечивает реализацию многих приложений MS-DOS, Windows 9x и части приложений OS/2, имеются средства для работы в Unix- и Novel-сетях.

Операционная система OS/2

Операционная система OS/2 версии 2.0 является одной из первых 32-разрядных ОС для персональных компьютеров. Это совместная разработка фирм IBM и Microsoft. Разработкой OS/2 предполагалась замена DOS. Планировалось, что она будет поддерживать вытесняющую многозадачность , многонитевидность, виртуальную память, графический интерфейс.

Развитием этой операционной системы стала OS/2 Wrap (1994) предназначенная для клиентских машин сетей (машин пользователей). По технологии и используемому интерфейсу эта операционная система похожа на операционные системы Windows 9x. OS/2 Wrap содержит специальную программу, позволяющую работать с любыми подключенными к сети устройствами, а также имеет специальные средства, позволяющие работать с приложениями DOC, Windows и их файловыми системами.

OC LAN Server 4.0 фирмы IBM использует объектно-ориентированный интерфейс OS/2 для создания мощного набора графических средств администратора сети. LAN Server 4.0 работает с новой версией OS/2, обеспечивающей симметричную многопроцессорную обработку, поддерживает файловые системы NTFS и FAT, а также протокол TCP/IP. Для организации доступа используются утилиты, входящие в состав Wrap 3.0. Имеется возможность выделения квот дискового пространства для пользователей . При достижении установленного предела система предупреждает как администратора сети, так и пользователя. LAN Server позволяет администратору создавать профили (группы) пользователей, регулируя их доступ к определенным системным ресурсам. Можно создавать единую процедуру входа пользователя в сеть, а также организовывать централизованное управление сетевыми ресурсами с помощью концепции доменов.

Семейство операционных систем UNIX

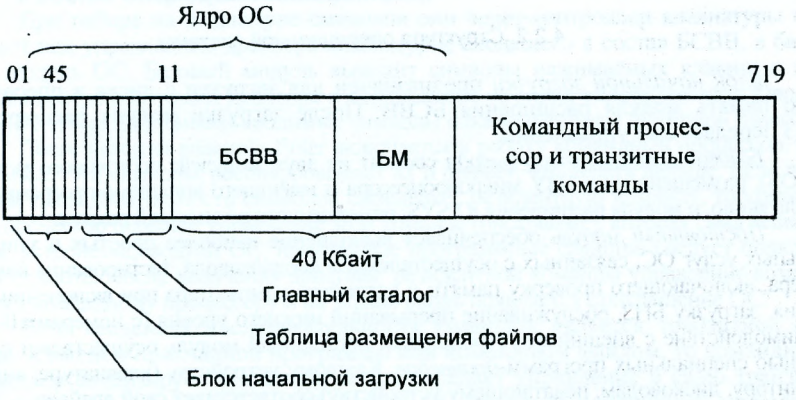
Начало разработкам этой операционной системы было положено фирмой AT&T. В настоящее время различные версии этой ОС выпускаются различными фирмами. Общими чертами этих операционных систем является то, что они

многопользовательские, многозадачные, ориентированы главным образом на большие локальные (корпоративные) и глобальные сети. Операционная система UNIX завоевала большую популярность прежде всего благодаря легкой переносимости на другие типы ЭВМ и обширному набору системных программ, которые позволяют создать благоприятную обстановку для системных программистов. Кроме того, UNIX обеспечивала работу в сети в режиме диалога в реальном масштабе времени. Наиболее перспективной операционной системой этого семейства считается ОС Linux, используемая для персональных компьютеров и малых серверов. Эта ОС развивается для многих типов процессоров.

4.2.3. Операционная система MS-DOS

Структура операционной системы

Операционная система MS-DOS продолжает еще достаточно широко использоваться на персональных компьютерах. Такая широко известная графическая операционная система как Windows 98 позволяет работать в сеансе MS-DOS. Операционная система MS-DOS размещается также на загрузочных дисках для установки ОС Windows или для ее восстановления при сбоях. Знакомство с операционной системой MS-DOS позволяет лучше понять принцип работы операционной системы. Кроме того, операционная система MS-DOS или ее элементы, хотя и в сильно измененном виде присутствует во всех операционных системах.



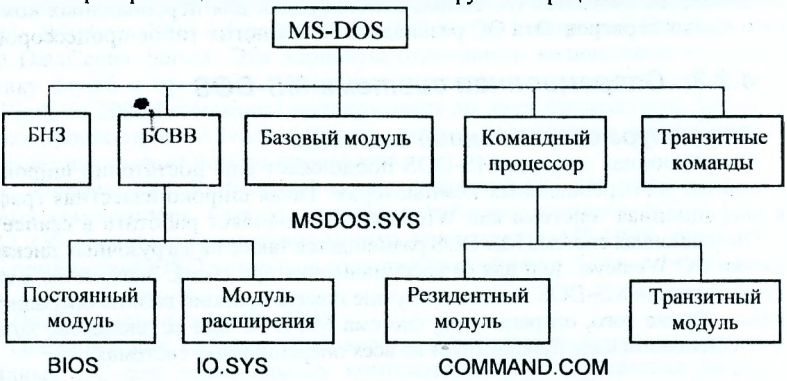
4.2.1. Структура размещения информации на системном диске

Операционная система MS-DOS хранится на одном из дисков, называемом системным. Структура системного диска приведена на рис. 4.2.1, а структура ОС MS-DOS представлена на рис. 4.2.2. В состав ОС MS-DOS входят: блок начальной загрузки (БНЗ); базовая система ввода-вывода (БСВВ); базовый модуль (БМ); командный процессор (КП); сервисные программы-утилиты (транзитные команды).

Блок начальной загрузки размещается всегда на нулевой стороне, в первом секторе нулевой дорожки любого диска. Модуль расширения БСВВ (файл IO.SYS) и базовый модуль (файл MSDOS.SYS) также размещаются в определенной области дисковой памяти. При форматировании диска эту область памяти можно зарезервировать для последующей записи модулей ОС. В даль-

нейшем эта область становится недоступной для пользовательских программ и утилит, что гарантирует сохранность ОС от случайного удаления или перемещения. Командный процессор размещается на диске в любом месте вслед за базовым модулем в файле command.com, как обычный файл.

На современных компьютерах на жестком диске может быть установлено несколько операционных систем. В этом случае на диске С создается главная загрузочная запись, где указаны адреса размещения блоков начальной загрузки операционных систем.



4.2.2. Структура операционной системы

Блок начальной загрузки предназначен для загрузки с диска в оперативную память модуля расширения БСВВ. После загрузки модуля расширения БНЗ передает ему управление.

Базовая система ввода-вывода состоит из двух модулей: постоянного модуля (BIOS), размещенного в ПЗУ микропроцессора и имеющего аппаратно-программную реализацию, и модуля расширения IO.SYS, размещенного во внешней памяти.

Постоянный модуль обеспечивает выполнение наиболее простых и универсальных услуг ОС, связанных с осуществлением ввода/вывода, тестирования компьютера, включающего проверку памяти и устройств компьютера при включении питания, загрузку БНЗ, обслуживание прерываний нижнего уровня (с номерами 0-31). Взаимодействие с внешними устройствами постоянный модуль осуществляет с помощью специальных программ-*драйверов.* Каждому устройству (клавиатуре, видеомонитору, дисководам, печатающему устройству) соответствует свой драйвер.

Модуль расширения БСВВ содержит дополнительные драйверы устройств. Он придает гибкость всей системе и позволяет создавать различные конфигурации вычислительной системы. Модуль расширения реализован в виде файла IO.SYS. Модуль расширения после загрузки его в оперативную память осуществляет необходимую подстройку прерываний и других параметров MS-DOS в соответствии с заданными в файле config.sys командами конфигурирования, обеспечивает загрузку остальной части ядра ОС — базового модуля и командного процессора и передает управление базовому модулю.

Базовый модуль обеспечивает работу файловой системы, обслуживает прерывания верхнего уровня (запросы прикладных программ и внешних устройств). Базовый модуль связан только с драйверами и не зависит от аппарат-

ной части. Обращение к нему производится либо из прикладной программы, либо из другого модуля ОС — командного процессора.

Командный процессор обрабатывает команды, вводимые пользователем. При загрузке в ОЗУ он разбивается на две части: резидентную и транзитную. Резидентная часть располагается вслед за базовым модулем. Транзитная часть размещается в конце области пользователя (часть оперативной памяти, отведенная для пользовательских программ). При недостатке объема памяти она будет стерта и освобождающая часть ОЗУ будет передана в распоряжение прикладной программы. После окончания работы прикладной программы резидентная часть командного процессора восстановит транзитную часть.

В состав резидентной части командного процессора входят программы модуля обработки прерываний, программа загрузки транзитной части командного процессора, программа инициализации для обработки файла загрузки автозапуска autoexec.bat, а также программы обработки наиболее часто используемых команд: работы с каталогами, настройки операционной системы, большинство команд работы с файлами.

Транзитная часть состоит из программ обработки редко используемых команд, например, команд работы с дисками.

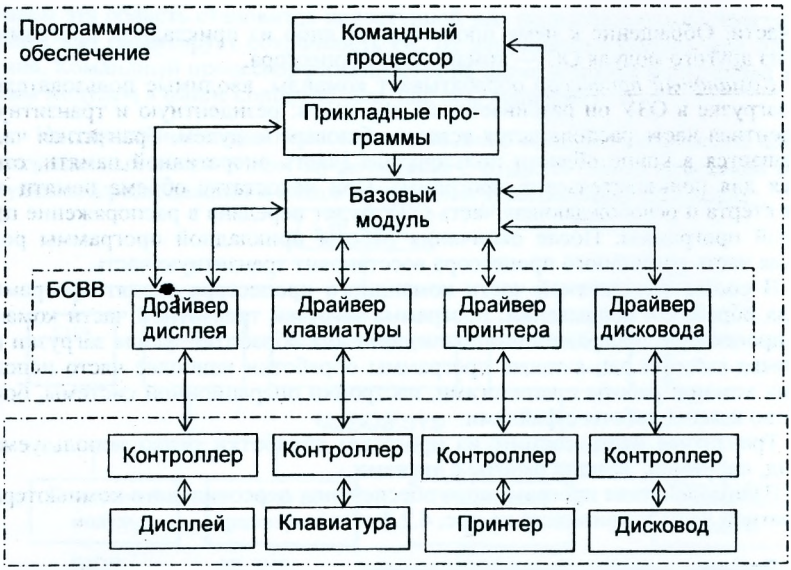
Взаимодействие программного обеспечения персонального компьютера с аппаратной частью приведено на рис. 4.2.3.

Работа операционной системы

При наборе на клавиатуре символов они через контроллер клавиатуры попадают под управлением драйвера клавиатуры, входящего в состав БСВВ, в базовый модуль ОС. Базовый модуль выводит символы нажимаемых клавиш с помощью соответствующих драйвера и контроллера на экран видеомонитора. Параллельно с этими действиями вводимые символы накапливаются в буферной памяти.

После нажатия клавиши Enter включается в работу командный процессор. Он принимает эти символы, расшифровывает их и решает, что делать дальше. Если введенная последовательность символов определяет резидентную команду, то командный процессор передает управление этой команде (программе), которая сама решает, какие устройства, модули ОС, прикладные программные средства следует подключить к работе. После окончания действия резидентной команды управление возвращается командному процессору и на экране вновь появляется приглашение ОС.

Если введенная последовательность символов является транзитной командой, именем прикладной программы или командным файлом, то командный процессор анализирует введенное имя и передает управление в свою транзитную часть, а далее — в базовый модуль ОС, который подключает драйвер и контроллер дисковода и обращается к файловой системе для поиска на диске требуемого файла. Содержимое найденного файла через контроллер дисковода под управлением драйвера дисковода и загрузчика переписывается в ОЗУ. Затем командный процессор передает управление этой программе. После выполнения команды или при выходе из среды прикладной программы управление возвращается командному процессору, инициирующему на экране видеомонитора приглашение ОС. Если введенная последовательность символов является ошибочной, то командный процессор, не найдя такой команды, выдает сообщение: BAD COMMAND OR FILENAME (неверная команда или имя файла).



4.2.3.Взаимодействие программного обеспечения персонального компьютера с аппаратной частью

Похожие действия протекают в ОС и в случае возникновения любых сигналов прерывания, логического, программного или аппаратного типа. Сигнал прерывания приостанавливает работу процессора, состояние обрабатываемого процесса запоминается и, в зависимости от кода прерывания, подключает модуль ОС, который обслуживает это прерывание. Затем операционная система начинает реализовывать функции, определяемые кодом прерывания. После окончания обработки прерывания процессор переходит к выполнению процесса, который выполнялся в момент поступления сигнала прерывания.

4.2.4. Устройства компьютера

Операционная система осуществляет ввод и вывод информации на различные устройства. Такими устройствами являются, как известно, дисководы, монитор, клавиатура, порты параллельного и последовательного ввода/вывода информации. Операционная система обращается к этим устройствам по их логическим именам.

Дисководы, как физические устройства, имеют номера. Имя диска обозначается буквой латинского алфавита с двоеточием. Например: первый диск обозначается *A*:, второй — *B*:, жесткий диск — *C*:. Ввиду того, что емкость жестких дисков в настоящее время очень большая, их с помощью программных средств разбивают на логические диски. На жестком диске в этом случае могут быть расположены логические диски *D*:, *E*:, *F*:, *G*:, *H*: и т. д.

Для других устройств установлены следующие логические имена:

PRN — печатающее устройство (принтер);

CON — консоль. При вводе информации под консолью понимается клавиатура, при выводе информации — экран видеомонитора;

NUL — пустое устройство: все операции ввода-вывода для этого устройства игнорируются. Используется при отладке программы;

LPT1—LPT3 — устройства, присоединяемые к параллельным портам 1...3 (обычно это принтеры);

COM1—COM3 — устройства, присоединяемые к асинхронному последовательному порту 1,2,3;

AUX — дополнительное устройство, присоединяемое к асинхронному последовательному порту 1.

Наиболее часто используются устройства PRN и CON. Имена устройств используются в командах ОС для обращения к ним.

На современных компьютерах число разъемов значительно больше: имеются специальные разъемы для подключения звуковых колонок, устройств мультимедиа, игровых приставок, внешней сети, телефона.

4.2.5. Файловая система

Файлы

Информация на дисках или других машинных носителях, а также в памяти компьютера хранится в файлах.

> *Файл — это поименованная область на диске или другом машинном носителе.*

В файлах могут храниться тексты программ, документы, данные.

Все файлы кодируются двоичным кодом, но кодируются по-разному. В связи с этим введено понятие *формат* файла. Признак формата файла указывается в расширении имени файла.

Программа MS-DOS различает два вида файлов: двоичные и текстовые (ASCII – файлы). Признак двоичного файла (используется программой по умолчанию) служит указанием интерпретатору, что размер файла определяется числом байтов, указанных в каталоге. Текстовые файлы заканчиваются, как правило, символом конца файла (вводится комбинацией клавиш Ctrl –Z). Если текстовый файл содержит символы латинского алфавита, то при его использовании проблем практически не возникает. Если же файл написан на русском языке, то при использовании этих файлов на импортных моделях ЭВМ могут возникнуть трудности, если они не имеют соответствующего программного обеспечения для перекодировки. На импортных ЭВМ используется чаще всего ASCII — код, на отечественных — различные модификации таблиц кодов ASCII. Наиболее часто используется альтернативная кодировка (Alt), а также новая кодировка (New). Указанные недостатки характерны для программ для ДОС.

Имя файла

Каждый файл на диске имеет обозначение, состоящее из двух частей: имени и расширения.

Имя файла в ОС MS-DOS может содержать от 1 до 8 символов. Имя должно начинаться с буквы и не должно содержать знаков пунктуации и пробелов, может содержать специальные символы: _ - . $ # & @ ! % ( ) {} `~ ^.

Расширение начинается с точки, за которой следует три символа. Расширение имени файла не является обязательным, однако оно характеризует вид хранимой в нем информации. Основные расширения имен файлов приведены в табл. 4.2.1. Примеры имен файлов: *command.com, expert1, autoexec.bat*

autoexec - имя файла

.bat – расширение имени файла

autoexec.bat – полное имя файла.

Таблиц 4.2.1

Примеры использования типовых расширений имен файлов

Расширение	*Вид файла*
.COM	Программный файлы в машинных кодах
.EXE	Программный файл в машинных кодах
.BAS	Программный файл на языке БЕЙСИК
.PAS	Программный файл на языке ПАСКАЛЬ
.FOR	Программный файл на языке ФОРТРАН
.BAK	Файл-копия
.BAT	Командный файл
.TXT	Текстовый файл, документ

Расширения имен файлов, приведенные в табл. 4.2.1, являются зарезервированными, то есть используются по умолчанию при работе с соответствующими программами.

Файлы, имеющие расширение .EXE, .COM или .BAT, считаются внешними командами ОС. При вызове внешней команды можно вводить только имя файла без расширения. Если используется несколько файлов, имеющих одинаковые имена, но разные расширения, то при вводе имени этой команды ОС выполнит только одну программу в соответствии с приоритетом: .COM, .EXE, .BAT. Файлы с указанными расширениями называются *исполняемыми.* Причем файлы типа .COM и .EXE хранятся в двоичных кодах, а файлы с расширением .BAT — в символьном виде и содержат последовательность команд, которые должны выполняться также, как и при вводе с клавиатуры.

Файл типа .COM полностью готов к выполнению, в нем привязка адресов уже выполнена и поэтому все адреса правильно записаны в файле перед его загрузкой. В EXE-файлах привязка адресов выполняется при загрузке. Не все адреса программы могут быть установлены до тех пор, пока неизвестно ее положение в памяти. EXE-файлы имеют заголовок, который содержит информацию

об этой привязке. Эти файлы загружаются по указанной причине немного дольше и требуют больше места на диске, чем файлы типа .COM.

Маска

При выполнении некоторых операций с файлами: копировании, переименовании, удалении, поиске файлов возникает необходимость выделить группу файлов имеющих, например, одинаковые имена или расширения имен. В этом случае для выделения файлов применяются маски.

Маска или шаблон - это символ, который заменяет все слово, его часть или один символ.

В качестве маски используются символы * и *?*. Символ * заменяет все имя или расширение файла, символ *?* означает любой символ в месте расположения данного знака. Например:

. — все файлы на текущем диске;

*.com — все файлы с расширением .com;

a*.sys — все файлы с расширением .sys, имя которых начинается с символа "a";

contr?.bas — все файлы с именем contr и расширением .bas, отличающиеся последним символом в имени файла (contr1.bas, contr2. bas и т.д.).

Каталог

Имена файлов регистрируются на магнитных дисках в каталогах (или папках).

Каталог — специальное место на диске, в котором хранятся сведения о файлах.

К таким сведениям относятся: имя и расширение имени файла, размер файла в байтах, дата и время его создания или последней модификации, атрибуты файла, начальный адрес файла на диске.

На диске может быть несколько каталогов. Наличие каталогов позволяет сгруппировать файлы по назначению, теме или пользователю, что облегчает их поиск на диске. Структуру каталогов на диске принято называть деревом каталогов (рис. 4.2.4).

Каталог верхнего уровня называется корневым. Он обозначается символом "\" – обратный слеш. *Имена каталогов* образуются по тем же правилам, что и имена файлов.

Каталог, с которым работает пользователь в настоящее момент, называется текущим. Если в команде указать имя файла, то он будет отыскиваться или создаваться в текущем каталоге. Для отыскания или создания файла в другом каталоге необходимо перейти в этот каталог или указать путь (маршрут) к этому каталогу.

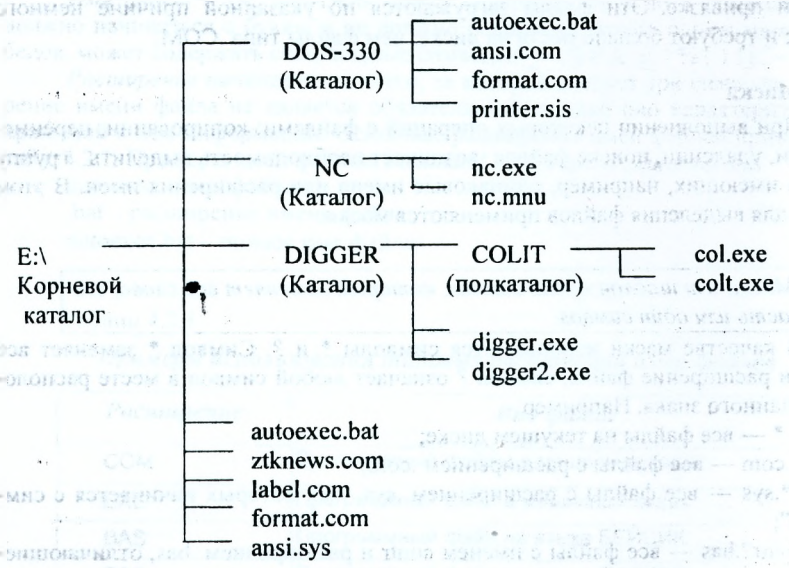


Рис.4.2.4. Дерево каталогов

Путь (маршрут) – последовательность каталогов, разделенная символом "\", ведущая к файлу.

Часто файл находится не на том диске, с которым работает пользователь. В этом случае при указании пути поиска необходимо указать еще и имя диска, например:

E:\DOS330\ansi.sys

E:\DIGGER\COLIT\col.exe

Здесь E: — имя диска; DOS330, DIGGER, COLIT - имена каталогов и подкаталогов.

Таким образом, полное имя файла включает имя диска, путь, имя и расширение имени файла:

[дисковод:] [путь] имя файла [расширение]

Обозначения, указанные в квадратных скобках, необязательны. Если дисковод не указан, то подразумевается текущий диск. Если каталог (подкаталог) является текущим, то путь также не указывается, например:

shrift.com

A: chrdsk.com

A: \UCH\ sessia.bas

Полное имя файла называют также ***спецификацией.***

Корневой каталог, таблица размещения файлов

Сведения о файлах и каталогах хранятся в специальной области диска. Сведения о каталогах хранятся в корневом каталоге, а сведения о файлах – в таблице размещения файлов – FAT. Для обеспечения надежности на диске создается две копии таблицы размещения файлов. Операционная система тщательно "следит" за тем, чтобы информация в обоих копиях была идентична.

Минимальной единицей размера файла на дискетах является сектор одной дорожки. Его размер равен 512 байтам. На жестких дисках минимальный размер файла может составлять несколько секторов. Эти группы секторов называют ***кластерами***. Таким образом, *кластер – минимальный размер адресуемого дискового пространства.* Размер кластера зависит от размера диска и от типа используемой таблицы размещения файлов. В операционной системе MS-DOS применяется файловая система FAT – 16, то есть она имеет 16 разрядов для адресации файлов и папок. Имея 16 разрядов можно задать 65536 адресов. Эта файловая система не может работать с дисками больших размеров из-за ограниченности количества адресов. Например, при емкости жесткого диска 1 Гбайт размер кластера составит 16 Кбайт, на диске размером 2 Гбайт размер кластера составит уже 32 Кбайта. Это значит, если создать файл содержащий всего один байт, то на диске он займет 32 Кбайт, то есть имеет место неэффективное использование дискового пространства. В операционной системе Windows 9x (общее обозначение семейства ОС Windows 95, Windows 98) применяется файловая система FAT – 32, которая успешно работает с дисками больших размеров. Но и в этой операционной системе при емкости диска 8 Гбайта размер кластера большой – 4 Кбайт. В связи с этими недостатками для операционной системы Windows NT и старших версий применяется 128 разрядная файловая система - NTFS. Эта файловая система обеспечивает лучшую защиту и большее быстродействие за счет кэширования файлов, а также большую эффективность использования дискового пространства.

4.2.6. Загрузка операционной системы

Включение питания

Включение современных компьютеров осуществляется предельно просто: достаточно нажать кнопку включения питания на лицевой панели системного блока и на панели управления монитора. Бытовые компьютеры имеют, как правило, одну кнопку включения питания на лицевой панели системного блока, но могут иметь также выключатель (тумблер), расположенный на задней стенке системного блока. В этом случае для включения компьютера необходимо включить вначале выключатель, а затем нажать кнопку включения питания.

Загрузка операционной системы

После включения питания идет самопроверка компьютера и проверка подключенных внешних устройств. Тестирование занимает значительное время, от нескольких секунд до нескольких минут. Основное время тратится на тестирование оперативной памяти. Объём проверенных блоков памяти (в

Кбайтах) выводится на экран монитора. Затем осуществляется проверка клавиатуры и видеомонитора, подключенных внешних устройств и определение конфигурации компьютера. Если будет обнаружена критическая ошибка: неисправность блоков памяти, неисправность клавиатуры, - то на экран выводится код ошибки и работа программы завершается. При неисправности монитора выдается последовательность звуковых сигналов. Характер неисправности определяется по руководству, прилагаемому к компьютеру - "тестовые программные средства". Если проверка прошла успешно, то BIOS пытается загрузить ОС с первого накопителя на гибком магнитном диске, а именно с диска A:. Если на первом дисководе не обнаружено дискеты, то осуществляется попытка загрузить ОС с жесткого диска или компакт-диска. Последовательность опроса дисков может быть изменена путем настройки BIOS.

Если на первом дисководе обнаружена неисправная или несистемная дискета, то выдается сообщение

Non system disk or disk error
Replace and strike any key when ready

(Несистемный диск или ошибка на диске — замените диск и нажмите любую клавишу).

При обнаружении на нулевой стороне в первом секторе нулевой дорожки блока начальной загрузки осуществляется загрузка ОС. В процессе загрузки на экран монитора выводится служебная информация (не обязательно). По окончании загрузки операционной системы MS-DOS на экране появляется приглашение системы, например, *C:\>*, где C: — имя текущего диска. Форма приглашения может быть различной и устанавливаться пользователем с помощью соответствующих команд ОС. После этого можно вводить команды ОС. После загрузки ОС Windows на экране появляется рабочий стол с элементами управления.

Если на компьютере установлено несколько операционных систем, то после обнаружения главной загрузочной записи на экран выводится меню, из которого надо выбрать ту операционную систему, которая требуется. Если операционная система не выбрана пользователем, то загружается операционная система, установленная по умолчанию.

В некоторых случаях, например, при "зависании" компьютера или желании перейти на другую операционную систему, требуется перезагрузка компьютера. Процесс перезагрузки ОС можно ускорить, если использовать "горячий" перезапуск, то есть загрузку ОС без выключения питания. При горячем перезапуске самопроверка компьютера не проводится и время загрузки ОС сокращается. Имеется несколько способов "горячего" перезапуска:

1) одновременное нажатие клавиш ***Ctrl-Alt-Del***. Этим способом можно загрузить новую ОС, если установить дискету с этой ОС на диск A:. При отсутствии дискеты на диске A: загрузка ОС будет осуществляться с жесткого диска;

2) нажатие клавиши RESET (перезапуск), установленной на лицевой панели системного блока. Эта клавиша используется обычно при "зависании" компьютера, когда командой *Ctrl-Alt-Del* не удается ввести режим перезагрузки.

Завершение работы. После окончания работы на компьютере необходимо подготовить его к выключению питания. Эта операция осуществляется командой park. При выполнении этой команды головки чтения/записи на жестком

диске устанавливаются в положение, при котором возможна транспортировка или перенос компьютера. После запуска программы park на экране монитора может появляться сообщение:

SYSTEM UNIT ALREADY SHUTDOWN, PLEASETURN THE POWER OFF.
(ЭВМ переведена в останов, можно выключить питание.)

Внимание! Во избежание механических повреждений поверхности жесткого диска перемещайте компьютер даже на небольшие расстояния только в запаркованном виде. Все современные компьютеры имеют программы парковки дисков, которые включаются автоматически при выходе из операционной системы и никаких сообщений на экран не выводится. При работе с ОС типа Windows выход из программы следует осуществлять только через команду главного меню ***Завершение работы***. В этом случае будут сохранены все настройки пользователя и обеспечивается сохранность данных. При выходе иным способом возможна потеря данных или даже повреждение жесткого диска.

Выключение компьютера

Выключение компьютера осуществляется в порядке, обратном порядку включения, т е. выключаются все подключенные внешние устройства: принтер, видеомонитор, системный блок. На практике это правило редко применяется.

4.2.7. Основные команды операционной системы

Команды ОС по назначению могут быть разделены на несколько групп. Рассмотрение всех команд операционной системы выходит за рамки настоящего учебного пособия. Поэтому ниже приведены основные, наиболее часто используемые группы команд:

- команды работы с дисками;
- команды работы с каталогами;
- команды работы с файлами;
- команды, образующие файл конфигурации;
- команды управления работой командного файла.

Формат команды имеет следующий вид:

<команда> [<опция>] [<опция>...] [/<опция>]

где обозначено:

<команда> — ключевое слово, определяющее характер выполняемых действий;

<опция> — содержит дополнительную информацию, уточняющую команду;

<опция...> — три точки указывают на возможность повторения опции в команде;

[.] — квадратные скобки указывают на необязательность выражения, заключенного в скобки;

/<опция> — ключ. Ключом называют опцию, начинающуюся с наклонной черты - слэш.

Команды работы с дисками

Команды работы с дисками обеспечивают: подготовку дисков к работе, то есть разметку дисков на дорожки, сектора, размещение служебной информации, перенос системных файлов; копирование и сравнение дискет; обслуживание накопителя на магнитном диске. К ним относятся следующие команды:

FORMAT — форматирование (разметка) диска;
DISKCOPY — копирование дискет;
DISKCOMP — сравнение дискет;
SYS — перенос системных файлов;
LABEL — получение/установка метки диска;
VOL — получение метки диска;
CHKDSK — проверка диска;
FDISK — обслуживание НМД.

Форматирование диска - *FORMAT*

Команда FORMAT обеспечивает подготовку дисков к работе: разбивает диск на сектора и дорожки, создает корневой каталог и таблицу размещения файлов.

Формат команды:

FORMAT D:[/v [:метка]] [/q][/u] [/1] [/8] [/n: секторов] [/t:дорожек] [/s] [/c]

или

FORMAT D:[/1] [/b] [/n: секторов] [/t: дорожек]

или

FORMAT D:[/v[:метка]] [/f:размер] [/s].
FORMAT [/?] – получение краткой справки по команде

Ключи имеют следующее значение:

/v [: метка] – здесь метка - текстовое сообщение длиной до 11 символов;

/q – безопасное форматирование. Удаляется корневой каталог и таблица размещения файлов. Данные могут быть восстановлены командой UNFORMAT;

/u – безусловное форматирование. Все данные безвозвратно теряются;

/1 — одностороннее форматирование, даже если используемый дисковод может работать и с двухсторонними дискетами;

/8 — восьмисекторное форматирование. По умолчанию дискета форматируется по 9 секторов на дорожку на дисководах емкостью до 720 Кбайт и по 15 секторов на дорожку на дисководах емкостью 1,2 Мбайт (для дискет размером 5,25 дюйма);

/b — восьмисекторное форматирование с резервированием места для системных файлов MS-DOS;

/n — указание числа секторов на дорожке. Этот ключ форматирует 3,5 — дюймовые диски на указанное число секторов (по умолчанию 18 секторов). Для 720К-дисков это значение равно 9, например: /n:9;

/t — указание числа дорожек на диске. Этот ключ форматирует 3,5 — дюймовые дискеты на указанное число дорожек. Для 720К - дисков и 1,44М дисков это значение равно 80 (/t:80);

/f — указывает размер форматируемой дискеты. Для дискет размером 3,5" возможны два размера: 1,44 Мбайта или 2,88 Мбайта. Данный ключ не может использоваться совместно с ключом /n.

Ключи /n и /t не могут быть вместе с ключом /f. Команда FORMAT индивидуальна для каждой версии DOC.

/s — копировать файлы ОС, перечисленные в файле FORMAT.TBL, с текущего диска на форматируемый диск. Новый диск должен иметь объем не менее 1,2 Мбайт (только для MS-DOS 4.0 и старше);

/c – проверка сбойных кластеров. Если в процессе форматирования диска программа обнаружила сбойный кластер, то по окончании форматирования диска эти кластеры можно повторно проверить и затем повторить форматирование диска.

/? – выдает краткую электронную справку по команде.

Приведенный формат соответствует MS-DOS 7.0. Команда работает в сети. По окончании работы она возвращает код завершения.

0 — успешное завершение;

3 — прерывание по нажатию CTRL-C;

4 — фатальная ошибка;

5 — не поступило ответа на запрос для жесткого диска: (форматировать (Д, Н) ?)

Эти коды можно использовать при пакетной обработке с помощью команды IF.

Например, команда FORMAT A: /s /v задает форматирование диска А с переносом системных файлов и заданием метки диска. После ввода команды на экран выводится запрос системы:

Insert new diskette to drive A: and strike ENTER when ready
(Установите дискету в дисковод А: Нажмите ВВОД, если готовы)

В процессе работы на экран дисплея выводится информация о стороне дискеты и номере проверяемой дорожки:

Head: 0 Cylinder: 21
(Поверхность: 0 Цилиндр: 21)

Если в процессе проверки будет обнаружена серьезная неисправность, то на экран выводится соответствующее сообщение, например:

Trac 0 bad — disk unusable
(Дорожка 0 испорчена, дискету использовать нельзя).

Это сообщение может появиться также при попытке инициализировать дискету с высокой плотностью записи (то есть емкостью 1,2 Мбайта) на дисководе для дискет емкостью 360 Кбайт.

По окончании проверки выводится сообщение:

Format complete System transferred
(Форматирование закончено, система перенесена).

Если была задана метка диска, выводится запрос:

Volume label (11 Character, ENTER for none)?

(Метка тома (11 символов, если нет — ВВОД)?)

В качестве метки может быть использована любая символьная информация. Если метка не требуется, нажмите Enter.

После ввода метки на экран выводится сводная информация о диске, например:

362 496 bytes totaldisk space	(общий объем дискеты в Кбайтах)
78 848 bytes used by system	(число Кбайт, занятых системными файлами)
5 120 bytes in bad sectors	(число Кбайт в неисправных секторах)
183 648 bytes available an disk	(объем свободной памяти в Кбайтах)

По окончании работы программа выдает запрос:

Format another (Y/N)?

(Форматирование еще (Y-да, N-нет)?)

Если есть необходимость разметить еще одну или несколько дискет, нажмите "Y" и Enter.

Копирование дискет — *DISKCOPY*

Формат команды: DISKCOPY [D:[D1:]] [/1]

Обеспечивает копирование дискеты, установленной в накопителе-источнике (D:), на дискету, установленную в накопителе — назначении D1:). Копирование осуществляется по дорожкам. Эта команда позволяет создать точную копию дискеты источника. При этом обеспечивается перенос и системных файлов. Ключ /1 задает копирование только одной стороны дискеты.

Сравнение дискет — *DISKCOMP*

Формат команды: DISKCOMP [D:] [D1:] [/1] [/8]

Команда обеспечивает сравнение дискет, установленных в дисководы D: и D1: по дорожкам.

Ключ /1 — задает режим сравнения только одной стороны дискеты, если даже дискета двухсторонняя.

Ключ /8 — означает сравнение только 8 секторов на каждой дорожке.

В процессе работы программы на экран дисплея выводится информация о результатах сравнения каждой дорожки.

При копировании или сравнении дискет на компьютере с одним накопителем на гибком магнитном диске в дисковод сначала устанавливается дискета источник (первая дискета), информация записывается в оперативную память. Затем программа просит установить дискету назначения (вторую дискету) и пересылает информацию из оперативной памяти на дискету назначения или сравнивает информацию, записанную в ОЗУ с содержанием второй дискеты.

Перенос системных файлов — *SYS*

Формат команды: SYS D:

Команда переписывает скрытые системные файлы с текущего накопителя на накопитель обозначенный D:. Каталог на диске должен быть пуст и диск должен быть размечен командой FORMAT с параметрами /S или /B.

Получение / установка метки диска — *LABEL*

Формат команды: LABEL [D:] [метка диска]

Если при разметке диска использован ключ /v, то команда LABEL позволяет записать, изменить или удалить метку. В качестве метки может быть использовано любое слово длиной до 11 символов.

Получение метки диска — *VOL*

Формат команды: VOL [D:]

В отличие от команды LABEL, команда VOL позволяет только вывести метку диска на экран дисплея.

Проверка диска — *CHKDSK*

Формат команды: CHKDSK [D:] [имя_файла] [/F] [/v]

Ключ /F устанавливает режим фиксации ошибок, обнаруженных в структуре каталога или в таблице размещения файлов, исправления записываются на диск. Ключ /v позволяет отобразить на экране дисплея местоположение всех файлов (имена и маршруты) указанного или текущего каталогов.

Программа CHKDSK выводит на экран сводную информацию о диске и оперативной памяти, вместо многоточий на экран будут выведены конкретные числовые значения:

- емкость диска, в байтах (... bytes total disk space);
- общий размер, в байтах, и количество "спрятанных" файлов (... bytes in ... hidden files);
- общий размер, в байтах, и количество каталогов на диске (... bytes m ... directories);
- общий размер, в байтах, и количество пользовательских файлов на диске ... bytes in ... user files);
- общий размер, в байтах, и количество файлов, восстановленных программой CHKDSK из потерянных участков на диске (... bytes in ... recovered files);
- количество свободного места на диске, в байтах (... available on disk);
- общий размер, в байтах, оперативной памяти компьютера (... bytes total memory);
- размер свободной (не занятой операционной системой и пользовательскими программами) оперативной памяти компьютера (... bytes free).

Если программа CHKDSK находит потерянные участки диска (т.е. участки, не принадлежащие ни одному из файлов и не числящиеся в списке свободных), то она выдает сообщение:

x lost clusters found in y chains
convert lost clusters to files Y/N)?

(Найдено xx потерянных кластеров, содержащихся в yy цепочках, преобразовать эти цепочки в файлы (Y — да, N — нет)?)

Если ответить "Y", то программа CHKDSK создаст в корневом каталоге файлы FILE0000.CHK, FILE0001.CHK и т.д. Их надо просмотреть и, если они не содержат ценной информации, уничтожить. Если ответить "Y", то потерянные участки сразу будут добавлены к списку свободных участков на диске.

Если программа CHKDSK сообщает о каких-либо ошибках на жестком диске, следует посмотреть в описании программы CHKDSK, какие действия необходимо предпринять. В случае серьезных ошибок надо немедленно сообщить об этом ответственному за компьютер или специалистам по техническому обслуживанию компьютеров. Иногда логические ошибки, которые не может исправить программа CHKDSK, исправляются программой NDD (Norton Disk Doctor).

Обслуживание накопителя на магнитном диске - *FDISK*

Формат команды: FDISK

Команда служит для подготовки к работе жесткого диска. Программа работает в режиме меню и позволяет:

- разбить диск на разделы. Создается один основной и один дополнительный раздел. В дополнительном разделе можно создать неограниченное число логических дисков. Основной раздел используется для установки операционной системы. Дополнительный раздел используется для размещения программ и данных. При необходимости установки на компьютере нескольких операционных систем вторую и последующие операционные системы следует размещать на логических дисках. На одном диске можно создать до четырех разделов, содержащих разные операционные системы, но не разные версии одной и той же ОС;
- удалить раздел. Нельзя удалить раздел принадлежащий другой ОС;
- изменить активный раздел. При загрузке ОС с жесткого диска загружается активный раздел.

Для получения раздела DOC необходимо:

- запустить программу FDISK;
- создать с ее помощью раздел DOC, определить его размеры и сделать раздел активным;
- отформатировать раздел командой FORMAT с ключом /s. Форматируется только созданный раздел;
- скопировать командой COPY с дискеты на жесткий диск файлы config.sys и system.sys.

После выполнения этих операций возможна загрузка ОС из созданного раздела.

Команды работы с каталогами

Команды работы с каталогами позволяют создавать, удалять, переименовывать каталоги, просматривать их на экране дисплея, обеспечивать быстрый и наглядный переход из одного каталога в другой, устанавливать дополнительные маршруты поиска файлов на диске. К ним относятся следующие команды:

DIR — вывод каталога файлов на экран монитора;

MD — создание нового каталога;
CD — смена каталога;
RD — удаление каталога;
TREE — быстрый переход в другой каталог;
PATH — указание дополнительного маршрута для поиска.

Просмотр каталога — *DIR*

Формат команды: DIR [D:] [путь] [имя_файла] [/p] [/w]

Команда позволяет вывести на экран оглавление диска, каталога, подкаталога или информацию о файле: объем в байтах, дату и время создания. Здесь D: — номер дисковода.

Ключ /p — выводит каталог постранично. После заполнения экрана вывод информации прекращается. Для продолжения просмотра следует нажать любую клавишу.

Ключ /w — выводит каталог в краткой форме. При этом на экран выводятся только имена файлов.

Команда DIR без параметров выводит каталог текущего диска.

Создание каталога — *MD*

Формат команды: MD [D:] [путь]

Команда создает новый каталог. Примеры:

MD UCH создает каталог UCH в текущем каталоге.

MD E:\UCH — создает каталог UCH в корневом каталоге диска E:

Смена каталога — *CD*

Формат команды: CD [D:] [путь]

Если задан дисковод, то каталог изменяется на указанном диске. Если диск не указан, каталог изменяется на текущем диске. Символ "\" в обозначении пути означает переход в главный (корневой) каталог, а символы ".." - переход в родительский каталог (надкаталог).

Примеры:

CD \ — переход в корневой каталог текущего диска;

CD .. — переход в надкаталог;

CD Lexicon — переход в каталог Lexicon текущего диска;

CD S:\QBasic— переход в каталог QBASIC диска S:

Удаление каталога — *RD*

Формат команды: RD [D:] [путь] <имя каталога>

Команда служит для удаления каталога. Удалить можно только пустой каталог, то есть каталог, в котором не записано ни одного файла. Текущий и корневой каталоги не могут быть удалены.

Пример: RD E:\DIGGER\ SHACH

удаляется подкаталог SHACH из каталога DIGGER.

Вывод на экран путей ко всем подкаталогам — *TREE*

Формат команды: TREE [D:] [/F] [/a]

Команда выводит на экран маршруты ко всем каталогам. Если указан ключ /F, то дополнительно выводятся имена файлов; ключ /a означает - использовать при выводе дерева каталогов символы псевдографики, имеющиеся во всех кодовых страницах.

Указание дополнительного маршрута, для поиска — *PATH*

Формат команды: PATH [D:] [путь] [; путь]

Задает дополнительный маршрут для поиска файлов. Пример:

PATH E:\ ;S:\DOS330;S:\VC

Задает поиск файлов в корневом каталоге диска E:, каталогах DOS330 и VC диска S:.

Команды работы с файлами

Команды работы с файлами обеспечивают копирование, удаление, сравнение, переименование, установку атрибутов и вывод файлов на стандартные устройства (дисплей, принтер).

К этой группе команд относятся следующие команды:

COPY — копирование файлов;

COMP — сравнение файлов;

REN[AME] — переименование файлов;

DEL (ERASE) — удаление файлов;

ATTRIB — установка атрибутов файлов;

TYPE — вывод файлов на экран монитора;

MORE — вывод файла на экран монитора порциями;

PRINT — вывод файлов на печатающее устройство.

Копирование файлов — *COPY*

Формат команды:

COPY [D:] [путь] <имя_файла или каталога> [/A] [/B] [/v]
[D1:] [путь] [<имя_файла или каталога>] [/A] [/B] [/v]

На первом месте в команде указывается откуда осуществляется копирование, на втором — куда. Можно указывать не полный формат, а именно в опциях *"куда?"* можно не указывать имени файла. В этом случае файл с диска [D:] будет скопирован на диск [D1:] с тем же именем.

Ключ */A* — обеспечивает копирование текстовых файлов, то есть файлов, сохраненных на диске в символьном виде.

Ключ */B* — обеспечивает копирование двоичных файлов.

Ключ */V* — задает проверку правильности копирования.

Команда COPY не обеспечивает копирования системных файлов. Системные файлы BIO.COM и DOS.COM (или IO.SYS и MSDOS.SYS) переносятся или в процессе форматирования дискеты с ключом /s или командой SYS. В по-

следнем случае при форматировании дискеты на ней должно быть зарезервировано место для системных файлов.

Команда COPY позволяет копировать файлы на консоль или на принтер:

```
COPY [D:] [путь] <имя_файла> CON
```

выводит файл на экран дисплея. Команду целесообразно применять только для текстовых файлов;

```
COPY CON [D:] [путь] <имя_файла>
```

создание файла. Данные вводятся с клавиатуры в файл на диске в указанный каталог. При этом для разделения строк выводимого файла в конце каждой строки необходимо нажимать клавишу Enter, а для окончания ввода нажать F6 а затем Enter;

```
COPY [D:] [путь] <имя_файла> PRN
```

выводит файл на печатающее устройство.

Команда COPY может использоваться также для объединения содержимого нескольких файлов в один файл:

```
COPY < имя_файла > [+ < имя_файла >] ... < имя_объединенного_файла >
```

В этом случае через знак "+" указываются имена объединяемых файлов, а в конце команды указывается имя файла, в который будет записано содержимое объединяемых файлов. Если имя объединенного файла не указано, то объединенный файл помещается в первый файл.

Сравнение файлов — *COMP*

Формат команды: COMP [D:] [путь] имя_ файла [D1:] [путь] имя_ файла

Команда позволяет сравнивать файлы. При указании имени файла можно использовать шаблон.

Переименование файлов и каталогов — *RENAME*

Формат команды: REN [AME] [D:] [путь] старое_имя новое_имя.

Команда RENAME допускает сокращение — REN. В именах файлов, так же, как и в команде COPY, можно употреблять символы "*" и "?":

```
REN S:\DOC\*.LST  *.TXT
```

Приведенная команда присваивает всем файлам каталога DOC на диске E:, имеющим расширение .LST, новое расширение .TXT.

Аналогично переименовываются и каталоги:

```
REN R:\UCH  UCH2
```

каталогу UCH на диске B: будет присвоено имя UCH2.

Удаление файлов — *DELETE*

Формат команды:

```
DELETE [D:] [путь] [имя_ файла [тип]] [/p]
```

Команда позволяет удалять указанные файлы. В имени файла обязательно должно быть указано расширение имени файла. В команде допускается использование шаблонов. Если параметры в команде опущены, то подразумеваются

все файлы. Ключ /P запрашивает подтверждение на удаление каждого файла. Для подтверждения удаления файла необходимо нажать "Y" и Enter. Для отказа — "N" и Enter или просто Enter.

Установка атрибутов файла. — *ATTRIB*

Формат команды:

ATTRIB [+/-R] [+/-A] [D:] [путь] <имя_файла> [расширение] [/s]

Команда ATTRIB обеспечивает установку и снятие защиты по записи или архивации на указанные файлы. Из защищенных файлов информация может только читаться, изменить их содержимое или удалить их средствами DOC невозможно. Атрибуты защиты задаются параметром *R*. Атрибуты архивации устанавливаются параметром *A*. Символ "+" перед параметром устанавливает защиту, а символ "-" — снимает защиту.

Ключ /s позволяет устанавливать или снимать необходимые атрибуты для всех файлов с заданным именем во всех подкаталогах, подчиненных каталогу, определенному заданным маршрутом.

Вывод файла на экран – *TYPE, MORE*

Формат команды: TYPE [D:] [путь] <имя_ файла>

Вывод файла на экран может быть осуществлен командой TYPE, выше это было сделано командой COPY.

Если файл не умещается на экране, то видна только последняя часть файла. Вывод файла на экран можно приостановить одновременным нажатием клавиш [Ctrl - S]. Для продолжения просмотра повторно нажмите эти клавиши. Для окончания вывода файла на экран следует нажать [Ctrl - C] или [Ctrl – Pause/Break].

Для вывода файла на экран порциями можно использовать команду TYPE с опцией MORE.

Формат команды: TYPE [D:] [путь] <имя_ файла> |MORE

Символ "|" в команде называют "трубопровод". Можно вывести файл на экран порциями и без использования команды TYPE:

MORE [D:] [путь] <имя_файла>

Вывод файла на печатающее устройство — *PRINT*

Формат команды: PRINT [D:] [путь] <имя_файла>

Это простейший формат команды PRINT. Команда обеспечивает печать файла в фоновом режиме. Эту команду можно использовать, если принтер оснащен непрерывной бумажной лентой. В формате команды [D:] — устройство, на которое выводится файл: PRN, LPT1, LPT2, LPT3, COM1, COM2, COM3.

По умолчанию файл выводится на принтер — PRN.

Команды настройки операционной системы. Файл автозапуска AUTOEXEC.BAT

В операционной системе MS-DOS имеется два файла, которые запускаются автоматически после загрузки операционной системы. Эти файлы позволяют настроить параметры среды пользователя в соответствии с его потребностями. Это файлы autoexec.bat и config.sys.

При написании файла autoexec.bat используются команды настройки операционной среды.

Команды настройки ОС создают удобную среду для работы пользователя. Они устанавливают текущую дату, время, форму приглашения. К командам этой группы относятся следующие команды:

DATE — выдача и установка текущей даты;

TIME — выдача и установка текущего времени;

PROMPT — установка формата приглашения ОС;

SET — задание параметров операционной среды.

Рассмотрим одну из них, команду SET.

Операционная система отводит специальную область памяти, называемую "окружением" (или средой), для хранения значений некоторых переменных, которые используются ОС и другими программами. Окружение состоит из строк вида: "переменная = значение".

Здесь переменная — любая строка, не содержащая символа "=". При этом прописные и строчные буквы латинского алфавита считаются равноценными; значение — любая строка символов.

Операционная система MS-DOS использует три переменные окружения:

PATH — устанавливается командой PATH;

PROMPT — устанавливается командой PROMPT;

COMSPEC — устанавливается командой COMMAND с параметром /P.

Пользователь может задавать переменные окружения с любыми другими именами с помощью команды SET.

Формат команды: SET [переменная = [значение]]

Если указанной переменной уже было присвоено какое-либо значение, то оно заменяется новым.

Прикладные программы могут анализировать область памяти, предназначенной для хранения переменных окружения, и выяснять, установлено ли значение той или иной переменной и каково ее значение. Наиболее часто переменные окружения используются для того, чтобы указать, где прикладные программы должны искать вспомогательные файлы. Значения переменных окружения часто используются в пакетных файлах. Команда SET без параметров выводит на экран информацию о переменных окружения.

Пример: SET ABC=E:\ABC

Этой командой вводится имя ABC с параметром E:\ABC, которое является указанием текстовому процессору ABC, откуда следует брать вспомогательные файлы – с диска E: из каталога ABC.

Параметры окружения задаются обычно в *файле автозапуска autoexec.bat* и устанавливаются при загрузке операционной системы.

Пример файла автозапуска:

```
echo=OFF
break=ON
path E:\; E:\NC
prompt $p$G
SET ABC=E:\ABC
MOUSE
DATE
TIME
```

В данном примере параметр *echo = OFF* запрещает вывод на экран приглашения DOC и последующих командных строк до появления команды *Echo— ON;* команда *break = ON* устанавливает режим прерывания программ при нажа-

тии клавиш *Ctrl - Break*; параметр *prompt pG* задает стандартное приглашение ОС: выдает имя текущего каталога и символ-разделитель ">"; параметр *path E:\; E:\NC* задает маршрут поиска файлов в корневом каталоге диска E: и в каталоге NC; параметр SET ABC=E:\ABC определяет маршрут поиска вспомогательных файлов текстовым редактором ABC; команда MOUSE загружает драйвер мыши; команды DATE и TIME запрашивают у пользователя текущую дату и время.

Файл автозапуска может редактироваться любым текстовым редактором. В файл autoexec.bat могут включаться программы перекодировки символов – ZTKNEWS, настройки принтера – PRINTER, печати графической копии экрана – GRAPHICS, программы для облегчения редактирования командной строки – DOSEDIT, проверки наличия компьютерного вируса, например: AIDSTEST, переопределения значений клавиш на клавиатуре – ANSI.SYS, программы оболочки, например Volcov Commander и другие команды, создающие удобную среду для работы пользователя.

Команды, образующие файл конфигурации. Файл CONFIG.SYS

Некоторые параметры операционной системы, а также используемые драйверы внешних устройств могут задаваться в специальном *файле конфигурирования системы config.sys*. В файле конфигурирования могут указываться также программы, расширяющие возможности ОС, которые необходимо загрузить в оперативную память (эти программы называют драйверами устройств, хотя они не управляют работой каких-либо устройств). Если файл конфигурации отсутствует, то параметры ОС устанавливаются по умолчанию.

Файл конфигурации config.sys представляет собой текстовый файл, использующий символы латинского алфавита. Каждая строка этого файла имеет вид:

имя_команды = значение

Наиболее часто используются команды Break, Buffers, Country, Files, Shell, Device.

Break = [ON/OFF] — устанавливает режим проверки нажатия клавиш *Ctrl - Pause/Break* при операциях ввода-вывода с диском. Это позволяет прерывать выполнение программ, которые иначе выполнялись бы до своего завершения.

Buffers = число буферов — установка числа буферов для операций ввода/вывода с диском. Чем больше установлено буферов, тем быстрее осуществляются операции ввода-вывода. Число устанавливаемых буферов зависит от емкости жесткого диска. Чем больше емкость диска, тем большее число буферов может быть установлено. Например, для компьютера с жестким диском емкостью 20...40 Мбайт рекомендовалось устанавливать 30...40 буферов. Максимальное число буферов равно 255.

Country = 061 — установка удобного формата выдачи информации о дате и времени, полное имя файла country.sys.

Files = 20 — установка максимального числа одновременно открытых файлов. Многие программы требуют для своего выполнения значение этого параметра не менее 20. Максимальное число одновременно открытых файлов равно 255.

Shell = COMMAND.COM/E: число байтов /P - увеличение размера области памяти, в которой хранятся переменные окружения. Число байтов задает размер этой области. Если размер области памяти недостаточен для хранения пе-

ременных окружения, то DOC выдает сообщение: "Out of environment space" (недостаточно места для переменных окружения).

Device = имя_файла - драйвера — установка драйвера устройства:

device = E:\EXE\SYS\ansi.sys

установит драйвер ansi.sys, расширяющий возможности по выводу на экран и позволяющий переопределять значения клавиш на клавиатуре.

Пример файла config.sys:

Break=ON	Включить режим проверки нажатия клавиш *Ctrl - Pause/Break*
Buffers=20	Установить 20 буферов для операций ввода / вывода с диском.
Files=40	Установить максимальное число одновременно открытых файлов 40
Country=061	Установить удобный формат выдачи информации о дате и времени
Device=ansi.sys	Установить драйвер управления выводом информации на экран ansi.sys
Device=vdisk.sys 384/E	Установить виртуальный диск (Vdisk), размер которого равен 384 Кбайта и который должен размещаться в расширенной памяти процессора (параметр /E).
Devicehigh=C:\Windows\himem.sys	Подключить драйвер дополнительной памяти для доступа к оперативной памяти , лежащей выше 1 Мбайт
Devicehigh=C:\CDROM\vide--cdd.sys/d:MSCOOO	Подключить драйвер дисковода CD-ROM

Файлы config.sys и autoexec.bat могут создаваться любым текстовым редактором. Параметры, задаваемые этими файлами устанавливаются только в момент загрузки ОС.

Контрольные вопросы

1. Что включает в себя базовое программное обеспечение?
2. Какие средства входят в систему программирования?
3. В каком месте диска размещаются: а) блок начальной загрузки, б) файлы IO.SYS и MSDOS.SYS; в) командный процессор?
4. Для чего предназначены программы-драйверы, где они размещаются?
5. В чем отличие резидентных и транзитных программ командного процессора?
6. Что такое файл?
7. Что такое имя, расширение и спецификация файла? Приведите примеры записи спецификации файла.
8. Назовите наиболее распространенные расширения имен файлов. Что они означают?
9. Поясните, что такое маска. Приведите примеры использования масок.
10. Поясните назначение файла autoexec bat.
11. Поясните назначение файла config.sys.
12. Поясните назначение команды FORMAT.

13. Что такое каталог? Какая информация в нем содержится?
14. Как просмотреть каталог диска постранично?
15. Как создать каталог?
16. Как скопировать файл? Укажите путь (маршрут) к этому файлу.
17. Как отформатировать новую дискету и перенести на нее системные файлы?
18. Как вывести на экран текстовый файл постранично?

5. Операционная система Windows

5.1. Общие сведения

Windows - графическая, многооконная, многозадачная, однопроцессорная операционная система.

Первая версия ОС Windows появилась в 1986 году. После этого она пережила ряд модификаций:Windows 3.1 работала в среде MS-DOS, как надстройка, Windows 95, Windows 98 – самостоятельные ОС, полностью совместимые с MS-DOS, Windows NT, Windows Millennium, Windows 2000, Windows XP – сетевые операционные системы.

Отметим главные особенности этих операционных систем.

Независимость программ от аппаратной части компьютера – программная совместимость. Windows–программа может обращаться к внешним устройствам только через Windows. Поэтому любая Windows–программа не зависит от конкретных особенностей внешних устройств и может работать с внешним устройством, если с ним может работать Windows;

Единый графический интерфейс. Пользовательские программы, разработанные для работы в среде Windows принято называть *приложениями*. Пользовательский интерфейс приложений Windows имеет одинаковое исполнение, единые приемы управления. Научившись работать в одном из приложений можно достаточно быстро освоиться в другом приложении. В состав пользовательского интерфейса входят такие элементы, как рабочий стол, окна, меню, контекстные меню, панели инструментов, линейки прокрутки, средства навигации, приемы управления элементами интерфейса. Развитие графических операционных систем привело к возникновению понятия *компьютерные технологии* - это алгоритмы и конкретные приемы работы в графической операционной среде.

> *Компьютерные технологии - это алгоритмы и конкретные приемы работы в графической операционной среде.*

Многозадачность – Windows обеспечивает возможность одновременного выполнения нескольких программ, переключения с одной задачи на другую, управления приоритетами выполнения программ.

Возможность обмена данными между приложениями. Для реализации этой возможности в ОС Windows предусмотрено три механизма:

- буфер обмена(Clipboard);
- DDE (Dinamic Data Exchange) – динамический обмен данными;
- OLE (Object Linking and Embedding) – механизм связи и внедрения объектов.

Буфер обмена – это специальным образом организованное динамическое пространство оперативной памяти для временного размещения данных. Программа хранит не только данные, но и сведения о программном приложении, которому оно принадлежит. После того, как данные помещены в буфер, они

могут быть вставлены из него в текущий или в любой другой документ. При помещении в буфер новых данных старые данные уничтожаются. Буфер можно просмотреть и при необходимости очистить.

Технология DDE представляет собой набор системных процедур, позволяющих обращаться из одного приложения (DDE – клиента) в процессе его выполнения к другому, активному на тот момент, программному приложению (DDE –серверу). По запросу клиента сервер обрабатывает запрос и возвращает результаты в той или иной форме приложению-клиенту. Имеется возможность установить привязку внедренного объекта к источнику данных, благодаря этой привязке все изменения, проведенные в приложении-сервере, будут отражены во внедренном объекте.

Технология OLE – это усовершенствованная технология DDE. Она позволяет не только внедрить объект, но и обеспечивает возможность редактирования внедренного объекта средствами программы-сервера. Для этого достаточно щелкнуть по внедренному объекту дважды мышкой.

Поддержка масштабируемых шрифтов. В Windows встроен совершенный механизм поддержки масштабируемых шрифтов True Type. Эти шрифты содержат не растровые (поточечные) изображения символов, а описания контуров символов, позволяющие строить символы любого нужного размера.

Удобство поддержки устройств. Поскольку вся работа прикладных Windows-программ с внешними устройствами осуществляется через посредство Windows, для подключения к компьютеру любого нового устройства достаточно установить драйвер этого устройства, предназначенного для Windows.

В Windows используется технология Plug and Play, упрощающая работу с компьютером за счет сервисных функций, которые позволяют распознавать устройства, динамически изменять конфигурацию системы и автоматически уведомлять об этом программные приложения.

Оборудование, выполненное в соответствии со стандартом "Plug & Play", считается самоустанавливающимся. Это значит, что его достаточно подключить только физически, операционная система сама найдет его, определит, что это такое, подберет в своей базе подходящий драйвер и пропишет его в своем реестре. А в тех случаях, когда устройство не соответствует стандарту "Plug & Play" или когда операционная система не может его распознать или не имеет в своей базе данных подходящего драйвера, установку можно выполнить с помощью программного обеспечения, полученного вместе с устройством при его покупке.

Возможность работать в сети. Последние версии операционной системы Windows обеспечивают пользователя всем комплексом услуг по работе в всемирной информационной сети Интернет: поиск информации, ведение списков посещаемых Web-узлов, поддержку всех основных стандартов Интернета; подготовку Web-страниц; передачу информации по электронной почте, факс-модему и др.

5.2. Особенности файловой системы

Операционная система Windows 98 поддерживает файловую систему FAT-16 и FAT-32. Windows 2000 и Windows XP поддерживают файловую систему NTFS и FAT-32. В этих операционных системах вместо понятия каталог введено понятие ***папка***. Однако, понятие папка более общее, чем понятие каталог. Если в каталогах хранятся сведения о файлах, то в папках могут храниться сведения о любых объектах. Для обозначения имен папок и файлов в операционных системах Windows можно использовать длинные имена. Имя папки

или файла может содержать до 255 символов, включая расширение. В имени файла могут использоваться любые символы, имеющиеся на клавиатуре кроме символов \ / : * ? < > |. В имени файла допускается использование пробела и точки, при этом последняя точка считается началом расширения имени файла.

В операционной системе Windows нет файлов настройки конфигурации операционной системы config.sys и autoexec.bat. Вместо этих двух небольших файлов в ОС Windows для хранения сведений об операционной системе, установленном оборудовании и настройках используется *системный регистр*. В системном регистре хранятся сведения не только об операционной системе но и о всех установленных программах и оборудовании. Все управление занесением или удалением данных из системного регистра осуществляется операционной системой автоматически.

5.3. Основы работы в Windows

5.3.1. Рабочий стол Windows

Запуск операционной системы и выключение компьютера

Принцип работы ОС Windows при запуске мало чем отличается от принципа работы операционной системы MS-DOS. Если на компьютере установлено несколько операционных систем, то на экран выводится список установленных операционных систем и пользователю предлагается выбрать нужную операционную систему. После проверки исправности оперативной памяти и проверки подключенных внешних устройств, осуществляется загрузка операционной системы в оперативную память компьютера и на экране появляется рабочий стол программы Windows (рис. 5.3.1).

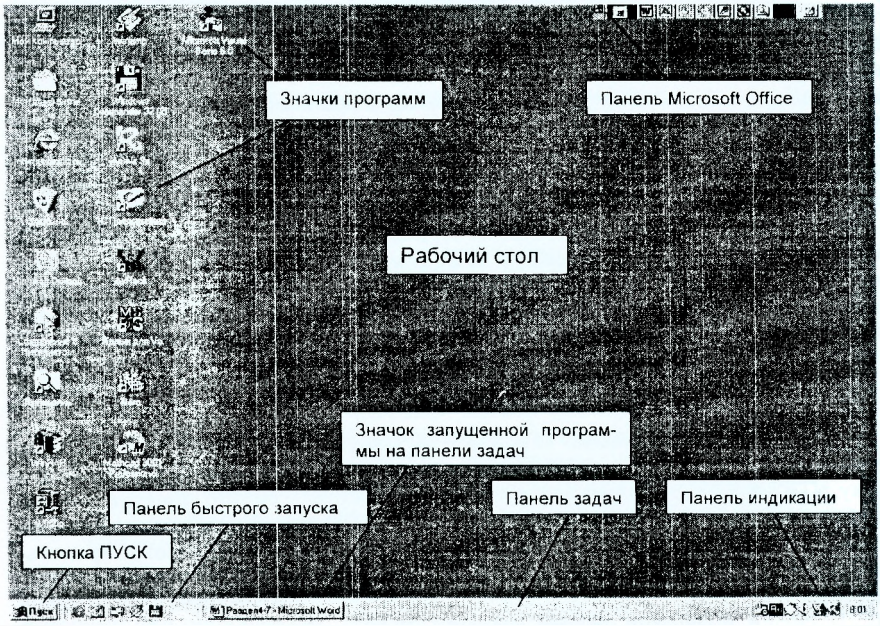


Рис.5.3.1. Рабочий стол

Если выключение компьютера было выполнено некорректно, то проводится проверка состояния дисков. Эти операции могут занять значительное время.

Для выключения компьютера следует выбрать команду *Пуск* и *Завершение* работы. Операционная система сохраняет установленную конфигурацию в системном реестре и выходит из программы и автоматически выключается питание. В операционной системе Windows XP для завершения работы на компьютере необходимо выбрать команду *Пуск, Выход из системы* и *Выключение* или сразу *Пуск, Выключение*.

При нарушении порядка выхода возможна потеря информации и даже повреждение жесткого диска.

На рабочем столе Windows размещаются объекты Windows: значки программ и папок, панель задач, панель Microsoft Office.

Среди значков отметим, прежде всего, "Мой компьютер" и "Корзину". "Мой компьютер" - это программа-навигатор, предназначенная для поиска нужных объектов на компьютере. "Корзина" – это программа, предназначенная для хранения удаленных файлов и папок. При необходимости корзину можно открыть и восстановить удаленный файл. Корзину периодически следует очищать для удаления накопившегося "мусора".

Панель задач

Панель задач размещается, обычно, внизу рабочего стола, но может быть установлена пользователем в любое положение. В левой части панели задач расположена кнопка *ПУСК*, справа – панель индикации. На панель задач могут выводиться также и другие панели. Например, в Windows 98 на панель задач могут выводиться *Панель быстрого запуска, Панель адресов, Панель каналов*.

Кнопка ПУСК открывает доступ к командам *Главного меню*. Чтобы изучить возможности Windows, достаточно просмотреть внимательно пункты меню:

Программы – позволяет найти и запустить любую программу. Запуск программ осуществляется двойным щелчком мыши по соответствующему пункту меню;

Документы - хранит список 15 документов, обрабатывавшихся на компьютере в последнее время;

Настройка - открывает доступ к программам настройки компьютера: Панель управления, Принтеры, Панель задач и меню ПУСК, свойства папки и рабочий стол;

Найти – позволяет осуществлять поиск на компьютере файлов, папок, людей;

Справка - открывает доступ к справочной системе Windows.

Выполнить – запуск программ через строку ввода.

Панель индикации - служит для отображения текущего времени, раскладки клавиатуры, регулятора громкости, настройки разрешающей способности экрана монитора и др. Если подвести указатель мыши к значку в панели индикации и задержать его на некоторое время *(зависнуть)*, то высвечивается либо дополнительная информация, либо открывается меню настройки свойств выделенного объекта.

Чтобы узнать назначение любого объекта в Windows зависните на нем мышью. Через некоторое время появится всплывающая подсказка.

Панель быстрого запуска – используется для размещения наиболее часто используемых программ.

Панель каналов – служит для размещения узлов Интернета, на которые можно осуществить "подписку" на услуги канала. Тогда в установленное время на компьютер будет сбрасываться новая информация.

Панель адресов - сюда можно помещать значки узлов Интернета, к которым приходится обращаться наиболее часто, чтобы они были всегда под рукой.

Контекстное меню

Одним из самых эффективных способов управления программами является использование контекстного меню. Для вызова контекстного меню щелкните правой кнопкой мыши по объекту. В контекстном меню отображаются все задачи, которые можно выполнить с данным объектом.

Панель Microsoft Office

Панель Microsoft Office – специальная программа, предназначенная для облегчения запуска офисных приложений Windows: редактора текста Word, электронной таблицы Excel, системы управления базой данных Access, системы подготовки презентаций Power Point и др. При запуске программы Microsoft Office на экран выводится панель с аналогичным названием. Панель может размещаться в любом месте по периметру рабочего стола. Запуск приложений осуществляется одним щелчком мыши по значку программы. На панель можно добавлять кнопки или удалять их. В принципе на панель Microsoft Office можно поместить значок любой программы.

5.3.2. Технология работы с мышью

Основным средством управления при работе в Windows является графический манипулятор мышь. Конечно, можно использовать для управления и клавиатуру, но это не эффективно. Поэтому начинающим пользователям полезно познакомиться с основными приемами работы с мышью.

Мышь имеет две или три клавиши, на последних моделях на месте средней клавиши размещают ролик. Для управления используются только две клавиши: левая и правая. По умолчанию всегда подразумевается левая клавиша. Если для управления применяется правая клавиша, это специально оговаривается. В Windows имеется возможность поменять настройку клавиш мыши для левшей. Ролик применяется для "прокрутки " документа.

При загруженном драйвере мыши на экране монитора появляется ее указатель. Форма указателя может изменяться в зависимости от того, какой объект выделен: заголовок формы, граница объекта, разделительная линия и тому подобное.

При работе с мышью используется определенная технология или, иначе, приемы работы. Этой технологии соответствуют определенные понятия.

Зависание мыши – установить указатель мыши на объект и задержать его на некоторое время.

Щелкнуть мышью – кратковременно нажать и отпустить клавишу мыши.

Дважды щелкнуть мышью – два раза кратковременно нажать и отпустить клавишу мыши. Интервал времени между нажатиями клавиши, необходимый для того, чтобы программа распознала нажатия клавиши как двойной щелчок, может настраиваться средствами Windows.

Выделить объект – это значит щелкнуть по нему мышью. Выделенный объект выделяется, обычно, синим цветом.

Зацепить объект – установить указатель мыши на объект, нажать и удерживать клавишу мыши.

Протащить мышью – зацепить объект и протянуть указатель мыши по другим объектам.

Перетащить мышью – выделить объект, зацепить его, убедиться, что возле указателя появился маленький прямоугольник с крестиком или без него и перенести объект. Перетаскивание объектов левой или правой клавишами мыши используют для перемещения или копирования объектов.

Перемещение объектов мышью. Для перемещения окон зацепите объект за заголовок и перенесите в другое место, отпустите клавишу мыши. Для перемещения других объектов зацепите их мышью и переместите в требуемое положение.

Изменение размеров объектов. У некоторых объектов, имеющих рамку, можно изменять размеры. Для изменения размера объекта зацепите его за границу так, чтобы курсор изменил форму на двунаправленную стрелку, например, ←‖→, и переместите границу в требуемом направлении.

Выделение группы объектов. Для выделения непрерывной группы объектов мышью выделите первый объект, нажмите клавишу Shift и, не отпуская ее, щелкните мышью по последнему объекту в группе. Отпустите клавишу мыши, отпустите клавишу Shift.

Выделение группы разрозненных объектов или групп объектов: выделите первый объект (или группу объектов), нажмите клавишу Ctrl и, не отпуская ее, щелкните мышью по выделяемым объектам (или выделите последующие группы объектов).

Перемещение и копирование объекта с помощью мыши. Выделите копируемый объект, зацепите его мышью и перетащите в требуемое положение. Во время перетаскивания объекта к указателю мыши прикрепляется маленький прямоугольник. При перетаскивании объектов при нажатой клавише ***Ctrl*** осуществляется копирование объектов. При перетаскивании объекта с нажатой клавишей Ctrl к указателю мыши прицепляется маленький прямоугольник с крестиком. *Перетаскивание файлов правой* клавишей отличается от перетаскивания левой клавишей мыши тем, что при перетаскивании файлов правой клавишей мыши при отпускании клавиши в новом положении объекта появляется меню, которое предлагает выбрать операцию копирования или перемещения.

5.3.3. Объекты

Операционная система Windows является объектно-ориентированной программой. Все объекты, с которыми она работает, представляются графически в виде значков: файлы, папки, диски и т. д. Многие значки в Windows стандартизированы, например, файлы текстовых, звуковых, графических и других документов.

Каждый объект обладает свойствами. К объектам в Windows относят все, что может быть различимо средствами операционной системы. С каждым объектом связано понятие свойств. Свойства – это параметры или характеристики объектов. Например, диск имеет такие параметры как: тип, имя, размер; характеристиками папки являются: место размещения, имя, размер, дата и время создания; свойствами файлов являются: имя, тип, размер, дата и время создания или последнего изменения, форма значка, который связан с файлом и адрес, где этот значок хранится, атрибуты файла, имя и адрес приложения, с помощью которого он может просматриваться и редактироваться, распечатываться, воспроизводиться и т. д.

5.3.4. Окна

Одним из важных элементов Windows являются окна – прямоугольные экранные области, внутри которых размещается графическая информация и элементы управления.

В зависимости от назначения окна делятся на окна программ, окна документов, окна диалога (запроса), окна сообщений. Все они отличаются составом элементов управления, размещаемых в них.

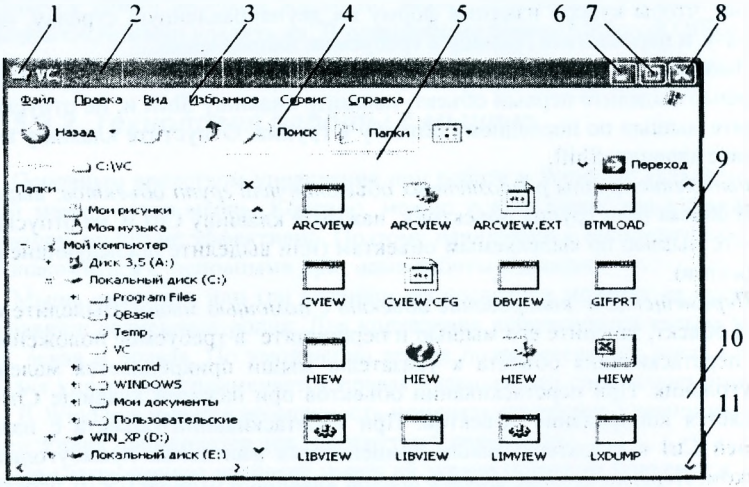


Рис. 5.3.2. Окно программы Проводник

На рис.5.3.2 представлено окно программы Проводник. В верхней части окна размещается строка заголовка (2), в которую выводится наименование откры-

той папки. В левой части строки заголовка размещается кнопка системного меню (1), а справа кнопки свертывания окна (6), развертывания окна (7), закрытия окна (8). Если окно развернуто, то на месте кнопки развертывания окна появится кнопка восстановления предыдущего состояния окна. Ниже строки заголовка размещается строка главного меню (3). При выборе пункта главного меню открывается всплывающее меню второго уровня. Меню второго уровня могут иметь подменю третьего уровня. Признаком наличия подменю у меню второго уровня является наличие стрелки ▶ в правой части строки меню. Некоторые пункты меню второго уровня выделены черным цветом, а некоторые - серым. Пункты меню, выделенные черным цветом, являются активными, а серые пункты меню – пассивными, они недоступны для управления программой. Состав активных пунктов меню второго уровня зависит от состояния программы. Меню позволяет ввести любую команду, предусмотренную для управления программой.

Ниже строки главного меню располагается , как правило, панель инструментов (4), на которую выводятся основные команды управления программой.

В рабочей части окна могут располагаться строки ввода, раскрывающиеся списки (5), кнопки, линейки прокрутки (10) и другие элементы управления (рис. 5.3.2). Линейки прокрутки появляются в окне программы в том случае, когда информация не умещается в открытой области окна. Линейки прокрутки могут быть горизонтальные и вертикальные. Они содержат ряд элементов управления (рис. 5.3.2): кнопки прокрутки (11), ползунок (9). Щелчок мышью по кнопке прокрутки перемещает экран на строку вверх (вниз) или на колонку вправо (влево). Для быстрого перемещения по окну служит ползунок, который иногда называют "лифт". Зацепив мышью за ползунок и перемещая его, можно быстро перейти к нужной области просмотра. Управлять просмотром можно также щелчком мыши по линейке прокрутки ниже или выше ползунка. При каждом щелчке окно просмотра перемещается на фиксированное число строк (столбцов).

На рис. 5.3.3. приведены основные элементы управления, используемые в окнах Windows. К таким элементам относятся:

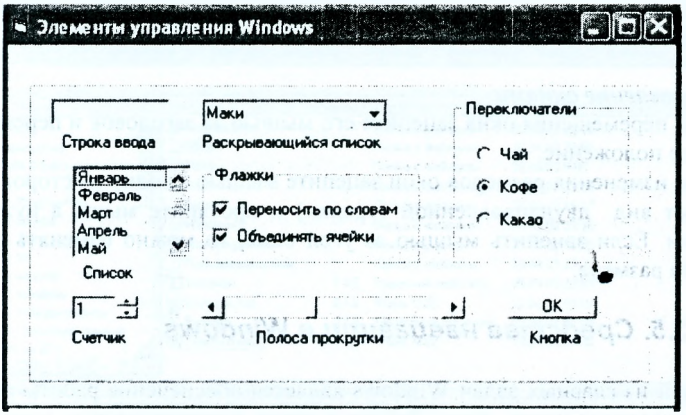


Рис. 5.3.3. Элементы управления Windows

- *строка ввода* - горизонтальная полоса белого цвета, служит для ввода буквенно-цифровой информации. Для активизации строки ввода необходимо

щелкнуть по ней мышью. При этом в строке ввода появляется курсор - вертикальная мигающая черта;

- *списки* – перечень элементов. Если весь список не умещается в окне, то справа появляется линейка прокрутки. Для выбора элемента списка необходимо щелкнуть по нему мышью;

- *раскрывающийся список* – список, который для экономии места в окне свернут. На экране присутствует лишь заголовок списка, содержащий строку ввода и кнопку раскрытия окна. Для выбора нужного элемента списка его необходимо открыть щелчком мыши по кнопке раскрытия окна, найти в списке нужный элемент и щелкнуть по нему мышью. Выбранный элемент списка записывается в строку ввода заголовка и список сворачивается;

- *флажки* – элемент управления в виде маленького прямоугольника. Флажок можно установить или снять. Если флажок установлен, то внутри прямоугольника появляется отметка в виде крестика или галочки. Снятие и установка флажка осуществляется щелчком мыши. Если флажок установлен, то действие, указанное в текстовом сообщении справа от флажка, выполняется.

В окне диалога может быть несколько флажков объединенных в группу. Все флажки независимы друг от друга. Поэтому установка или снятие какого-либо флажка не влияет на состояние других флажков.

- *переключатели* – представлены на экране кругом с пояснительной надписью справа. Переключатель может быть включен или выключен. Если переключатель включен, то он отмечается черной точкой внутри круга. Все переключатели объединяются в группу по умолчанию. При этом только один переключатель может быть включен, а остальные переключатели автоматически выключаются;

- *счетчики* – предназначены для ввода чисел. Ввод числа можно осуществлять либо вручную в строке ввода, либо установить с помощью кнопок, расположенных справа (слева, сверху или снизу) от строки ввода.

Управление окнами

Для перемещения окна зацепите его мышью за заголовок и переместите в требуемое положение.

Для изменения размеров окон зацепите мышью за одну из сторон (курсор принимает вид двунаправленной стрелки) и протяните мышь в нужном направлении. Если зацепить мышью за угол окна, то можно изменять одновременно оба размера.

5.3.5. Средства навигации в Windows

Одой из главных задач Windows является обеспечение работы файловой системы: создание папок, поиск файлов и папок, переименование, копирование, пересылка файлов и папок.

Для выполнения этих функций в Windows имеются средства навигации Internet Explore, Проводник, Мой компьютер.

Internet Explorer – программа предназначенная для навигации во всемирной информационной сети, в том числе и по Вашему компьютеру. Для навигации только по компьютеру предназначены программы Проводник и Мой компьютер. Какой из этих программ отдать предпочтение – дело вкуса и привычки.

Программа "Проводник"

Вызов программы

Вызов программы можно произвести несколькими способами. Самый простой – через команду главного меню Программы. Другой способ – через контекстное меню кнопки ПУСК. А лучше всего поместите значок программы Проводник на рабочий стол или в Панель Microsoft Office. Эту операцию выполняют следующим образом:

- найдите программу в главном меню: *Пуск, Программы, Проводник*;
- зацепите значок программы и перетащите его на Рабочий стол или непосредственно в Панель Microsoft Office.

Окно программы

Окно программы Проводник является типичным окном Windows (рис. 5.3.4). В верхней части окна выводится строка меню. Состав пунктов меню зависит от операционной системы. На рис. 5.3.2. приведена программа Проводник ОС Windows XP. Ниже строки заголовка расположена Панель инструментов, а под панелью инструментов – панель адресов. Ниже панели адресов располагаются панели программы Проводник и строка состояния. Состав элементов управления в окне программы зависит от установленных параметров в пункте меню Вид.

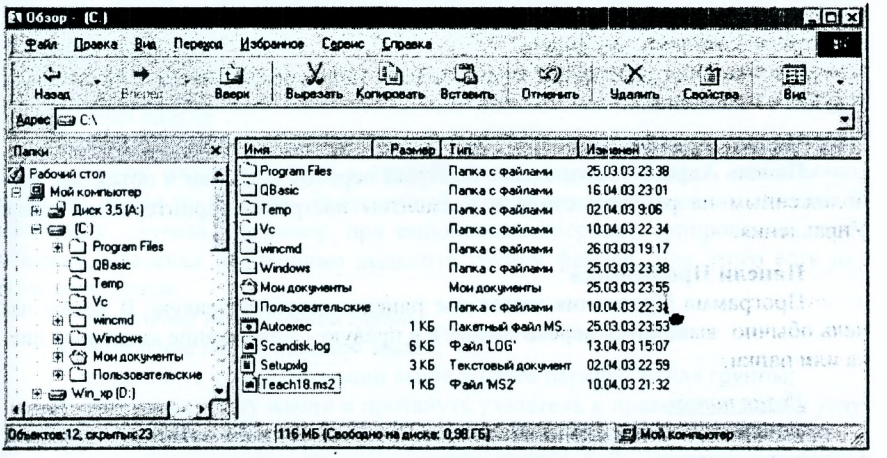


Рис. 5.3.4. Программа Проводник ОС Windows 98

Меню программы

Меню *Файл* предназначено для доступа к дискам, создания папок и ярлыков, открытия, удаления, переименования файлов и папок. Состав пунктов

меню зависит от того, какой объект выделен в панелях программы. В меню имеется также команда Свойства, которая позволяет просмотреть свойства выделенного объекта.

Меню *Правка* обеспечивает копирование, перемещение, выделение объектов.

Меню *Вид* содержит команды управления внешним видом окна программы и позволяет управлять составом элементов управления и индикации, размером значков, способом представления файлов и папок в панелях.

Меню *Сервис* содержит дополнительные средства, позволяющие наиболее эффективно выполнять некоторые операции. В частности здесь находится команда *Найти*, необходимая для поиска папок и файлов на дисках.

Пункты меню *Переход* и *Избранное* используются главным образом для работы в Интернет.

Меню *Справка* позволяет получить справку по текущему состоянию системы или найти подсказку по ключевому слову.

Такие пункты меню, как *Файл*, *Правка*, *Вид*, *Справка*, характерны для всех приложений Windows, хотя и отличаются по составу содержащихся в них опций. Для знакомства с любой программой достаточно внимательно изучить пункты меню, чтобы составить представление о ее возможностях.

Панель инструментов

Кнопки панели инструментов дублируют основные команды пунктов меню. Наибольший интерес представляют кнопки *Вперед, Назад, Вверх*. Благодаря кнопкам *Вперед и Назад* можно пройти по тому маршруту, которым мы двигались недавно по папкам и дискам. Рядом с этими кнопками имеются кнопки раскрытия списков. Открыв эти списки, можно быстро найти папку, которую мы недавно "посещали".

Кнопка *Вверх* позволяет подняться на один уровень вверх относительно текущего положения.

Панель Адреса

Панель Адреса обеспечивает быстрый переход к дискам и объектам, расположенным на рабочем столе и элементам настройки Принтеры и Панель Управления.

Панели Проводника

Программа Проводник имеет две панели: левую и правую. В левую панель обычно выводится дерево папок, а в правую – содержание активного диска или папки.

Левая панель

Содержанием информации, выводимой в левую панель можно управлять с помощью команды. *Вид, Панели обозревателя* В эту панель могут выводиться панель Папок, Поиск, Избранное, Журнал и др. Все эти панели предназначены для работы с сетью Интернет.

Панель *Папки*. Выводится, как правило, по умолчанию. Она содержит дерево папок. Корневой папкой считается Рабочий стол. В качестве объектов второго уровня выступают "Мой компьютер", "Мои документы", "Internet Explorer" и "Корзина". В моем компьютере содержатся диски, в дисках папки. У некоторых объектов слева стоит кнопка "+" или "-". Кнопка "+" свидетельствует о том, что папка свернута, и в ней содержатся папки следующего уровня. Если возле папки нет кнопки "+", значит она не содержит вложенных папок. Кнопка "-" свидетельствует о том, что папка уже развернута. Развернуть папку можно щелчком мыши по кнопке "+", а свернуть – щелчком мыши по кнопке "-".

Чтобы открыть папку, необходимо щелкнуть по ней мышью. Содержание открытой папки отображается в правой панели. Чтобы закрыть папку необходимо открыть другую папку или подняться на уровень вверх соответствующей кнопкой на панели инструментов, или воспользоваться кнопкой Назад.

Панель *Поиск* – содержит ссылки на некоторые поисковые системы Интернета, какие именно, зависит от настроек обозревателя Internet Explorer.

Панель *Избранное* – хранит адреса посещенных Web-узлов и Web-страниц, к которым предполагается обратиться еще раз. В этой панели, используя вложенные папки, можно удобно хранить адреса по темам и направлениям.

Панель *Журнал*. В этом журнале в течение заданного времени хранятся адреса Интернет, которые мы посещали.

Через панель Избранное можно перейти на панель *Каналы*. Каналы – это особая часть Web-узлов Интернета. Каналу можно "давать поручения " сбрасывать новую информацию на компьютер автоматически. Такое поручение называют "*подпиской*". При оформлении подписки задают график доставки информации. Инициатором связи в этом случае выступает не пользователь, а соответствующий Web-узел.

Правая панель

Правая панель содержит папки и файлы, имеющиеся в родительской папке. Выделение папок и файлов в правой панели осуществляется, как обычно, щелчком мыши по объекту. Выделенный объект отмечается синим цветом. В некоторых случаях, например, при выполнении операций копирования, перемещения, удаления необходимо выделить группу файлов. Для этого есть несколько приемов:

1) выделение непрерывной группы:

а) выделение протягиванием мыши;

- установить указатель мыши выше и левее первого файла группы;
- нажать клавишу мыши и протянуть указатель к правому нижнему углу группы. Выделяемая область помечается пунктирной линией.

б) выделение с использованием клавиши Shift:

- выделить первый файл группы;
- нажать и удерживать клавишу Shift;
- щелкнуть мышью по последнему файлу группы;

- отпустить клавишу Shift.

2) выделение произвольной группы файлов:

а) использование клавиши Ctrl:

- выделить первый файл группы;
- нажать и удерживать клавишу Ctrl;
- выделить щелчком мыши файлы, включаемые в группу;
- отпустить клавишу Ctrl.

б) использование клавиш Ctrl и Shift:

- выделить первый файл группы;
- нажать и удерживать клавишу Ctrl;
- выделить щелчком мыши файлы, включаемые в группу;
- при необходимости выделить непрерывную подгруппу файлов в пределах выделяемой группы, продолжая удерживать клавишу Ctrl, необходимо нажать и удерживать, на время выделения подгруппы, клавишу Shift;
- отпустить клавишу Ctrl.

Управление представлением файлов в панели

Для управления представлением файлов в панели используется меню Вид или раскрывающийся список в панели инструментов с соответствующим названием. Имеется несколько способов представления файлов в панелях: *крупные значки, мелкие значки, список* и *таблица*. Выбор того или иного способа – дело не только вкуса, но и цели просмотра. При большом числе файлов в папке целесообразно использовать для просмотра *список*. В этом случае на экран выводятся только пиктографические значки, определяющие тип файла, и их имена. При выводе файлов в панель в виде *таблицы* на экран выводятся дополнительно размер файла, дата и время его создания или последнего обновления.

Полные сведения о каждой папке и файле можно получить с помощью команды *Свойства* меню Файл или Контекстного меню.

Сортировка файлов и папок

Сортировку файлов и папок в панели можно выполнить с помощью команды *Вид, Упорядочить значки*. Сортировку можно выполнить по имени файла, типу, размеру или дате изменения. Сортировка выполняется по возрастанию кода символов в кодовой таблице компьютера: цифры 0...9, буквы латинского алфавита A...Z, a...z, буквы русского алфавита А...Я, а...я. Команда *автоматически* используется для выравнивания значков при представлении файлов в виде крупных или мелких значков.

В режиме *Таблица* сортировку можно осуществлять щелчком мыши по заголовку столбца. При этом повторный щелчок по заголовку столбца приводит к сортировке содержимого панели в обратном порядке.

Создание папок

Чтобы создать новую папку, выберите команду *Создать, Папку* в меню *Файл* или *Контекстном меню*. В панели появится папка с именем *Новая папка*. Присвойте папке другое, оригинальное имя.

Копирование и перенос файлов и папок

Копирование – двухместная операция. При копировании всегда присутствует источник информации и приемник информации. Алгоритм копирования зависит от используемого способа копирования и адреса источника и приемника:

1) копирование с помощью меню (Главного или Контекстного) или панели инструментов:

- выделить файл или группу файлов;
- выбрать команду *Правка, Копировать*. Информация помещается в буфер;
- перейти в папку – место назначения;
- ввести команду *Правка, Вставка*.

2) копирование левой клавишей мыши:

- вывести в правую панель папку-источник данных;
- вывести в левую панель диск и папку-назначение, раскрывая папки щелчком мыши по кнопкам "+"
- выделить файл в правой панели;
- зацепить файл и перенести его в папку-назначение. Во время переноса к указателю мыши прикрепляется маленький прямоугольник.

При копировании с помощью правой клавиши мыши, после отпускания клавиши мыши в месте назначения на экран выводится запрос на подтверждение выполняемой операции: *копировать, перемещать, создать ярлык*.

Перенос файлов выполняется так же, как и копирование, но после выполнения копирования *исходный файл удаляется* с диска.

При копировании файлов с помощью мыши из одной папки в другую в пределах одного диска исходный файл остается на прежнем месте, а имя файла переносится в указанную папку. Из исходной папки файл удаляется. То есть осуществляется не копирование, а перенос файла.

> *Для копирования файлов в пределах одного диска пользуйтесь правой клавишей мыши.*

Переименование файлов и папок

Для переименования файла или папки необходимо:

- выделить файл или папку;
- ввести команду *Правка, Переименовать;*
- записать в строке ввода новое имя папки или файла.

5.3.6. Настройка рабочего стола

Windows имеет средства для выполнения различных настроек, позволяющих учитывать индивидуальные особенности пользователя, создавать удобную и приятную пользовательскую среду.

Рассмотрим некоторые, наиболее важные настройки.

Настройка фонового рисунка и заставки, разрешения экрана

- щелкните правой клавишей мыши по рабочему столу – появится контекстное меню;
- введите команду *Рабочий стол Active Desktop, Настройка рабочего стола*. Появится окно диалога для настройки параметров рабочего стола (рис. 5.3.5) .

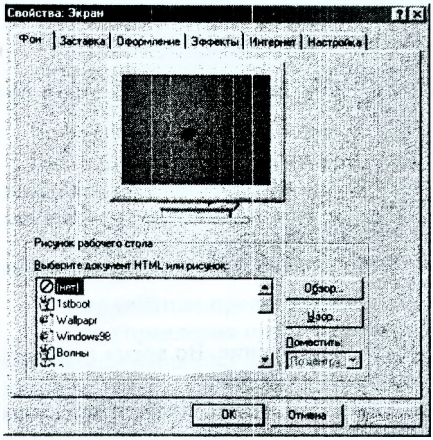


Рис. 5.3.5. Настройка параметров рабочего стола

Данное окно диалога позволяет изменить следующие параметры:

- фоновый рисунок рабочего стола. Выбирается из списка представленного на экране или осуществляется поиск растровых рисунков с расширением имени файла .bmp с помощью кнопки *Обзор*;
- выбрать заставку, которая будет появляться на экране, когда пользователь не осуществляет никаких операций на компьютере – закладка "Заставка";
- изменить оформление рабочего стола, меню, документов и т. д. (не рекомендуется) – закладка "Оформление";
- поменять значки объектов рабочего стола, размер значков и др. – закладка "Эффекты";
- показывать рабочий стол, как Web-страницу, можно изменить способ управления открытия файлов и папок на стиль Web-документов (например, открытие файлов и папок одним щелчком, выделение файлов не щелчком мыши, а установкой указателя мыши на имя файла) – закладка Интернет;
- настроить разрешение экрана, частоту обновления экрана, цветовую палитру, установить флажок для вывода в панель индикации панели задач значка настройки разрешения экрана (*Настройка, Дополнительно*, установить флажок "Вывести значок настройки на панель задач) – закладка Настройка. *Не рекомендуется* изменять настройки, установленные по умолчанию.

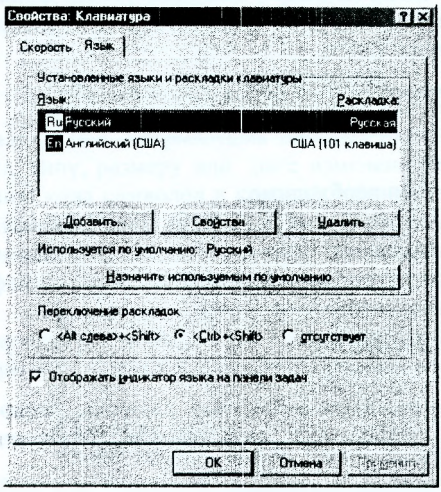


Рис. 5.3.6. Настройка клавиатуры

Настройка раскладки клавиатуры

Настройка раскладки клавиатуры позволяет установить скорость повтора символов при удержании нажатой клавиши на клавиатуре, время задержки перед повторением символов (закладка "Скорость"), язык, используемый по умолчанию, изменить способ переключения на ввод русских символов, добавить другой язык (при наличии дистрибутивной дискеты), установить режим отображения индикатора языка на панели задач. Для вывода на экран окна диалога "Свойства: Клавиатура" (рис. 5.3.6) необходимо ввести команду *Пуск, Настройка, Панель управления, Клавиатура.*

Настройка панели задач

Для настройки панели задач щелкните правой клавишей мыши по *Панели задач* и выберите команду *Свойства.*

В окне диалога на закладке *"Параметры Панели задач"* можно установить или снять флажок "Отображать часы", а также установить или снять флажок "Автоматически убирать с экрана". Если флажок "Отображать часы" установлен, то в Панели индикатора Панели задач появляется текущее время. Двойным щелчком мыши по времени можно вызвать окно диалога для настройки часов. Если установлен флажок "Автоматически убирать с экрана", то при запуске какой-либо программы панель задач убирается с экрана, освобождая место для работающего приложения. Чтобы в этом случае открыть Панель задач, необходимо подвести указатель мыши к нижней границе экрана.

5.4. Сервисные оболочки

5.4.1. Сервисная оболочка Volcov Commander

Назначение и запуск программы

Программа Volcov Commander (VC) наряду с программой Norton Commander является одной из наиболее популярных программных оболочек для работы с командами операционной системы MS-DOS. Она позволяет выполнять такие общие команды, как просмотр каталогов, копирование, переименование и удаление файлов и каталогов методом "подведи курсор и нажми", что значительно проще, чем путем ввода командной строки.

Запуск Volcov Commander осуществляется командой "VC", вводимой в командной строке.

Volcov Commander, как и многие другие сервисные оболочки, является многооконной системой. Используются окна трех видов: панели, окна диалога и окна сообщений. Панели используются для вывода списка файлов и каталогов, текстовых файлов и другой информации. Окна диалога содержат, как правило, строку ввода информации и набор команд для выбора варианта действий. Окна сообщений выводят информацию о состоянии системы и имеют только одну кнопку для закрытия окна. Кроме того, имеется встроенное меню, которое имеет набор команд для настройки среды пользователя и управления ком-

пьютером. Меню построено по иерархическому принципу. Первое меню горизонтальное. При выборе требуемого пункта меню открывается вертикальное меню второго уровня, которое может содержать меню следующего уровня.

Для выхода из VC следует нажать клавишу *F10*. В центре экрана появляется запрос на подтверждение. Чтобы выйти, необходимо нажать *Enter* или "*Y*", чтобы отменить команду — *Esc* или "*N*".

Volcov Commander позволяет получить помощь по работе с программой. Вывод помощи на экран осуществляется нажатием клавиши *F1*. Помощь выдается на русском языке.

Данная программа имеет небольшой набор функций, проста в изучении. Ее полезно изучить для освоения работы с файловой системой.

Общие принципы управления

Информация в панелях и меню представляется в виде списка. Для перемещения по списку используются клавиши управления перемещением курсора ←, →, ↑, ↓, а также клавиши *Home* — установка курсора в начало списка и *End* — установка курсора в конец списка. Для быстрого перемещения по списку можно использовать также клавиши прокрутки PgUp и PgDn. Выделить требуемый пункт меню можно также с помощью мыши.

Для ввода команды ее необходимо выделить *курсором* и нажать клавишу *Enter*. Нажатие клавиши Enter завершает ввод команды. В дальнейшем изложении материала, когда речь будет идти о вводе команд, клавиша Enter может не упоминаться.

Для завершения ввода команды нажмите клавишу Enter.

Ввод команд можно производить также с помощью *горячих клавиш* - символов, выделенных в команде цветом, жирным шрифтом или символом подчеркивания.

Работа в среде Volcov Commander

Описание экрана

После запуска VC на экране появляются два прямоугольных окна, ограниченных двойной рамкой (рис.5.4.1.) Эти окна обычно называют *панелями*. Ниже панелей размещается строка ввода команд с приглашением ОС, а в последней строке экрана — справка о назначении функциональных клавиш.

В каждой панели VC может изображаться:

– оглавление каталога на диске;

– дерево каталогов на диске;

– сводная информация о диске и каталоге на другой панели;

– содержимое файла, выделенного на другой панели.

Оглавление каталога. Если в панели VC выводится оглавление каталога, то в верхней части панели выводится имя этого каталога. Имя каталога текущей панели подсвечено. *Текущей панелью* называется панель, в которой нахо-

дится *курсор* панели — инверсная полоса на черно-белых мониторах или цветная полоса на цветных мониторах шириной в одну строку экрана. Для перевода курсора в другую панель - *не текущую* следует нажать клавишу *Tab*.

В первых строках панели выводятся имена каталогов на диске. Имена каталогов изображаются прописными буквами, справа от имени каталога изображается <SUB — DIR>. Самую верхнюю строку занимает ссылка на родительский каталог. Для корневого каталога ссылки на родительский каталог нет. В поле имени для родительского каталога изображается двоеточие – символ "..", а справа от него <UP — DIR>.

Имена файлов выводятся строчными буквами. Для файлов с атрибутами "спрятанный" или "системный" имя файла начинается с прописной буквы, а между именем файла и его расширением выводится символ "▒" или "A", в зависимости от используемой кодовой таблицы.

Каталог может выводиться в *полной* или *краткой* форме. При полной форме выводятся имя файла, его размер в байтах, дата и время создания. При краткой форме выводится только имя файла.

E:\BASIC

Name	Size	Date	Time
..	UP-Dir	1-01-80	12:11a
CONTR	SUB-DIR	1-01-80	12:40a
DJAKONOV	SUB-DIR	1-01-80	12:02a
TORM	SUB-DIR	1-01-80	12:04a
UCH	SUB-DIR	1-01-80	12:51a
algtrans bas	176	1-11-89	1:01a
difurav bas	255	1-11-89	1:11a
expert bas	10002	1-01-80	3:31a
expert bat	286	1-01-80	12:04a
expert1 bas	10274	1-01-80	3:24a
expert4 bat	2793	1-01-80	1:22a
expert5 bat	1856	1-01-80	1:08a
qwbasic exe	80592	24-07-87	12:00a
integral bas	276	1-11-89	12:53a
	UP-Dir	1-01-80	12:11a

E:\ 1:30a

Name	Name	Name
ARC	bmbio com	ztknews com
BASIC	lbmdos com	aidstes1 exe
CALC	chkdsk com	aidstest exe
DGF	command com	pct exe
DIGGER	diskcomp com	sf exe
DOS330	diskcopy com	nc mnu
FOXBASE	dm8 com	treeinfo ncd
FOXRUS	dosedit com	ansi sys
KOROVA	fdisk com	config sys
LEX	graftab1 com	config sys
NU	m86 com	config sys
SIMP	park1 com	
TURBO_B	shrift com	
autoexec bat	sys com	
ARC	SUB-DIR	1-01-80 12:02a

E:\BASIC

1Help 2Menu 3View 4Edit 5Copy 6RenMov 7Mkdir 8Delete 9PullDn 10Quit

Рис. 5.4.1. Рабочий экран Volcov Commander

В нижней части панели выводится *строка мини-статуса.* Она отделена одной горизонтальной чертой. В строке мини-статуса отображается характеристика *выделенного файла* или *каталога.* Файл или каталог считается выделенным, если на него установлен курсор панели. Перемещение курсора внутри панели осуществляется клавишами управления перемещением курсора ←, →, ↑, ↓, а также с помощью клавиш *Home* — установка курсора в начало каталога и *End* — установка курсора в конец каталога.

Чтобы быстро выделить файл в текущем каталоге, необходимо нажать клавишу *Alt* и, не отпуская ее, набрать в командной строке первые буквы имени файла. Volkov Commander выделит нужный файл, как только будет введено достаточно символов имени файла для его распознания.

В выделенный каталог можно "*войти*", нажав *Enter*. При этом в текущей панели появится оглавление выделенного каталога. Для *выхода* из текущего каталога в родительский каталог необходимо переместить курсор панели в начало каталога на символы ".." и нажать *Enter*.

Быстрый поиск файла на диске. Для быстрого поиска файла во всех каталогах на диске необходимо нажать [Alt - F7]. Открывается окно поиска файлов. В строке ввода необходимо набрать имя файла, который требуется найти, и нажать клавишу *Enter*. При указании имени файла можно использовать маску. После нажатия клавиши *Enter* состояние окна меняется. В рабочую часть окна выводятся имена найденных файлов, а в нижней части окна поиска выводится меню команд: *Stop* - остановка поиска, *New Search* - новый поиск, *Go to* - переход в каталог, Quit - выход. При поиске по маске на экран выводятся имена всех файлов, удовлетворяющих условию поиска. После окончания поиска выделите нужный файл и введите команду *Go to*. Программа переходит в тот каталог, где находится выделенный файл. Для продолжения поиска используется команда *New Search.*

Управление панелями

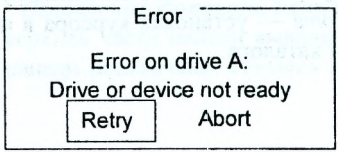
Панели Volcov Commander находятся на экране постоянно. При выполнении команд они временно убираются, а затем автоматически восстанавливаются, так что результаты выполнения команды, выведенные на экран, оказываются закрытыми панелями. Для просмотра результатов необходимо убрать одну или обе панели. Управление панелями осуществляется с помощью комбинаций клавиш:

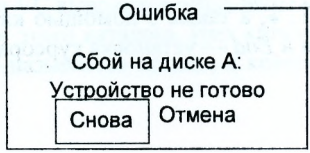
Ctrl – F1 – убрать левую панель с экрана или вывести ее на экран;
Ctrl – F2 – убрать правую панель с экрана или вывести ее на экран;
Ctrl - 0 – убрать обе панели с экрана или вывести их на экран;
Ctrl - P – убрать одну из панелей (не текущую) с экрана или вывести ее на экран;
Ctrl - U – поменять панели местами.

Переход на другой диск

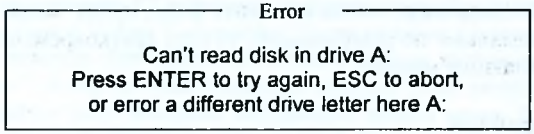
Для вывода в панель оглавления другого диска необходимо нажать *Alt – F1* для левой панели и *Alt – F2* для правой панели. После нажатия клавиш в панели появляется список доступных дисков. Затем необходимо выбрать имя нужного диска клавишами ←, → и нажать *Enter*. Для отказа от перехода на другой диск нажмите *Esc*.

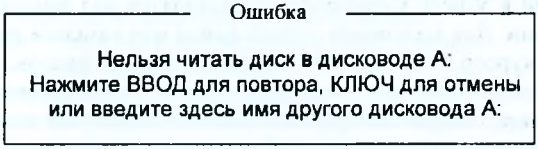
В случае, если диск не отформатирован или дефектный, появляется соответствующее сообщение.





В случае возникновения таких ситуаций можно повторить операцию, выбрав пункт меню "*Retry*" и нажав *Enter*, или отказаться. Команда *Abort* позволяет окончить ту функцию VC, при которой она возникла. Если ошибка возникла при чтении диска, то будет выдано сообщение о невозможности выполнить операцию:





В выделенное окно можно вывести номер другого диска, например D:. При отсутствии неисправностей после нажатия клавиши *Enter* в соответствующую панель выводится каталог выбранного диска.

Запуск программ и команд DOC

Если требуется выполнить программу или команду дисковой операционной системы, необходимо, как обычно, набрать эту команду в командной строке с помощью клавиатуры и нажать *Enter*. Во время выполнения команды панели убираются с экрана, а по окончании выполнения программы восстанавливаются.

Для корректировки содержимого командной строки используются клавиши: →, ←, Home, End — перемещение по командной строке (клавиши → и ← не действуют, если в текущей панели выведено оглавление диска в краткой форме).

Для ускорения набора имени файла в командной строке можно использовать средства VC: установите курсор строки ввода команд на свободное место, выберите в одной из панелей нужный файл и нажмите *Ctrl - Enter*. Имя выбранного файла помещается при этом в командную строку.

VC позволяет выводить в командную строку ранее введенные команды:

Ctrl – E — выводит предыдущую введенную команду;

Ctrl – X — выводит команду, которая была выполнена после той, что находится в командной строке.

Alt – F8 – выводит список всех команд (хранится информация о 10...15 последних командах). В этом списке можно выделить и выполнить нужную команду.

Команду, выведенную в командную строку, можно отредактировать и затем исполнить.

Для запуска исполняемых и командных файлов (файлов, имеющих расширение .EXE, .COM или .BAT) необходимо выделить нужный файл и нажать *Enter*.

Если при запуске исполняемых файлов требуется указать другие опции, то следует поступить следующим образом:

– снести выделенный файл в командную строку командой *Ctrl – Enter*;

– дополнить команду необходимой опцией и нажать *Enter*.

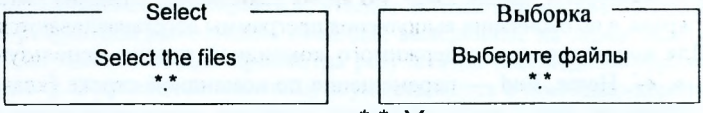
Использование мыши

При работе с манипулятором мышь VC выводит на экран красный прямоугольник — указатель мыши. Его можно перемещать по экрану, двигая мышь по гладкой поверхности, например, по столу. Работа с мышью аналогична работе с клавиатурой. Например, чтобы выделить файл, пункт меню или вариант ответа на запрос, *щелкните* по нему *мышью*, то есть кратковременно нажмите и отпустите левую клавишу мыши.

Выделение файлов

Все операции в Volcov Commander производятся над выделенными файлами или каталогами. Для выделения одного файла или каталога достаточно установить на него курсор. VC позволяет выбрать группу файлов, над которой можно выполнить некоторые действия: копирование, перемещение в другой каталог, удаление и т.д. Выбранные файлы выделяются желтым цветом на цветном дисплее и повышенной яркостью на монохромном дисплее.

Выбор файлов можно осуществлять двумя способами: с помощью клавиши Insert (Ins) или по маске. В первом случае курсор панели устанавливается на имя требуемого файла и нажимается клавиша *Insert*. Повторное нажатие этой клавиши отменяет выделение. Во втором случае поступают следующим образом: нажимают клавишу "+" на дополнительном поле клавиатуры. На экране появляется окно для ввода маски:



По умолчанию предлагается маска *.*. Можно указать свою маску, вводя соответствующие символы с клавиатуры. В маске можно использовать символы "*" и "?". Их смысл тот же, что и в командах MS-DOS.

Чтобы отменить выбор по маске, нажмите "-" на дополнительном поле и задайте маску файлов, выбор которых хотите отменить.

Использование функциональных клавиш

В Volcov Commander каждой функциональной клавише поставлена в соответствие определенная команда. Справка о назначении функциональных клавиш приведена в нижней строке экрана.

Ф1 - Help — получение справки о VC.

Ф2 - Menu — вызов меню пользователя для запуск программ, указанных в списке.

Ф3 - View — просмотр файла.

Ф4 - Edit — редактирование файла.

Ф5 - Copy — копирование файлов.

Ф6 - Remov — переименование файлов (каталогов) или пересылка файлов в другой каталог.

Ф7 - MkDir — создание каталогов.

Ф8 - Delete — удаление файлов и каталогов.

Ф9 - PullDn — вызов встроенного меню VC.
Ф10 - Quit — выход из VC.

Получение справки о VC

Нажатие клавиши *F1* позволяет получить подробную справку о назначении клавиш при работе с Volcov Commander. После нажатия клавиши F1 на экране появляется меню помощи. Для просмотра меню и выбора требуемого пункта используются клавиши ↑, ↓, PgUp, PgDn. Выделите нужный пункт меню и нажмите *Enter*. На экране появляется текст описания выбранного пункта меню. В нижней части окна находится меню управления просмотром: Next, Previous, Index, Cancel.

Next — переход к следующему пункту меню;
Previous — переход к предыдущему пункту меню;
Index — выход в меню помощи;
Cancel — выход из режима помощи.

В многостраничных описаниях пунктов меню для просмотра текста (прокрутки) используются клавиши ↑, ↓, PgUp, PgDn.

Просмотр файлов

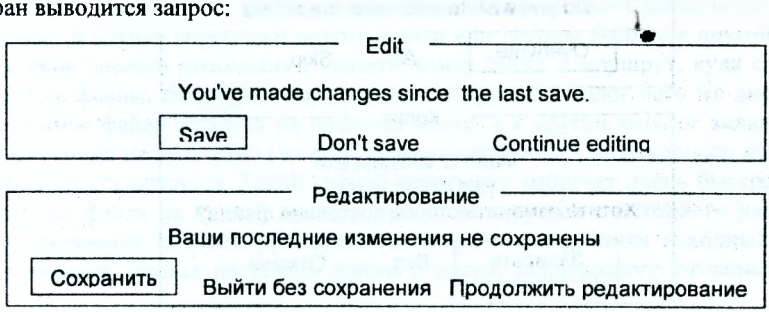
При нажатии клавиши F3 на экран выводится файл, выделенный курсором. Можно просматривать текстовые файлы, документы, созданные с помощью различных редакторов текстов, базы данных и таблицы электронных таблиц, например, файлы dBase, FoxBase+. Для перемещения по просматриваемому файлу можно использовать все клавиши управления перемещением курсора, а также клавиши PgUp, PgDn, Home, End. При просмотре баз данных можно использовать клавиши "+" и "-" на дополнительном поле для перемещения по полям записи.

Редактирование файлов

Для редактирования файла выделите его курсором и нажмите F4. После ввода команды появляется экран встроенного редактора текста. Наберите текст. В конце каждой строки набираемого текста необходимо нажимать клавишу *Enter*. Для сохранения набранного документа на диске используется команда *Save* (клавиша *F2)*. Для сохранения документа под другим именем нажмите *Shift - F2*. В открывшемся окне наберите имя диска, маршрут, имя файла и нажмите *Enter*.

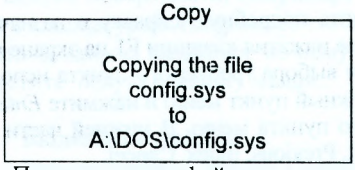
Для создания нового файла нажмите комбинацию клавиш *Shift - F4*. В открывшемся окне запишите имя файла, нажмите *Enter*. На запрос программы введите команду *New-File* (Новый Файл).

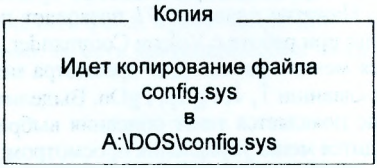
Выход из режима редактирования осуществляется командой *Quit* – клавиша *F10* или клавишей *Esc*. Если редактируемый файл не был сохранен, то на экран выводится запрос:



Копирование файлов

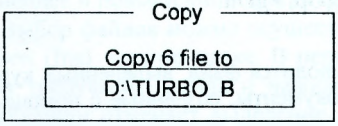
Копирование файлов осуществляется с помощью клавиши *F5*. После нажатия этой клавиши на экране появляется окно, в котором указано, что копируется и куда, например [Copy "config.sys" to].

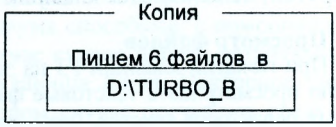




По умолчанию файл копируется в каталог, выведенный в нетекущую панель. Можно указать свой маршрут и имя файла, набрав его в строке ввода. Затем для завершения ввода команды нажать *Enter*.

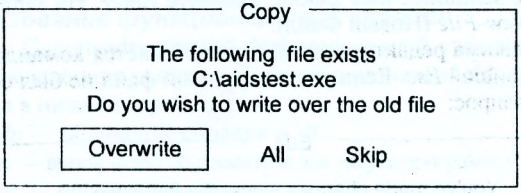
При копировании нескольких файлов в окне будет указано, сколько файлов копируется:

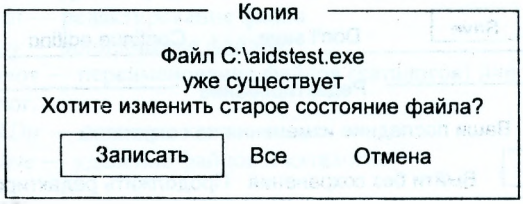




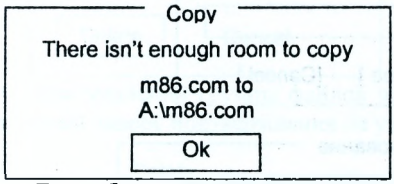
В нижней части окна диалога выведено меню команд *Copy, F10 Tree и Cancel.* При открытии окна курсор выбора меню устанавливается на команду *Copy*. Выбор нужного пункта меню осуществляется клавишами →, ←. При выборе пункта *Fl0-Tree* на экран выводится дерево каталогов текущего диска для быстрого выбора каталога, куда надо копировать файл. Команда *Cancel* служит для отказа от режима копирования. При выборе пункта *Copy* и нажатии *Enter* на экран выводится сообщение о выполняемой операции.

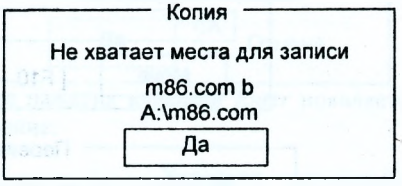
В процессе копирования на экран могут выводиться сообщения и предупреждения об ошибках. Если на диске, куда осуществляется копирование, есть файл с таким же именем, то на экран выводится соответствующее сообщение. При выборе пункта *Записать* старый файл будет уничтожен, а новый записан на его место. В случае выбора пункта меню *Все* не будет выдаваться запроса на подтверждение копирования. При выборе пункта *Отмена* программа перейдет к копированию следующего файла.





При копировании может случиться, что на диске, куда помещается копия файла, недостаточно места для его размещения. В этом случае на экран будет выведено сообщение:

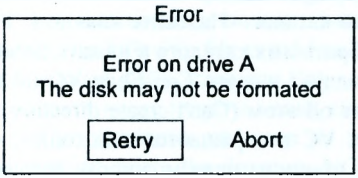


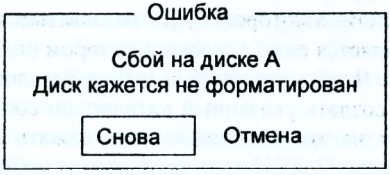


Для обеспечения наглядности и контроля результатов копирования файлов рекомендуется следующий алгоритм работы:

- выведите в нетекущую панель оглавление диска и каталога, куда необходимо копировать файлы;
- выделите в текущей панели файлы, подлежащие копированию;
- нажмите клавишу F5;
- нажмите клавишу Enter.

Когда диск назначения не готов (не установлен или не отформатирован) на экран выводится соответствующее сообщение:





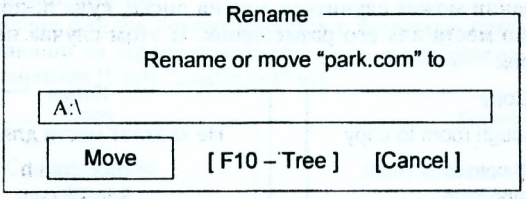
В этом случае необходимо проверить правильность установки диска и повторить операцию, выбрав пункт меню *Retry*, или отказаться от копирования, выбрав команду *Abort*.

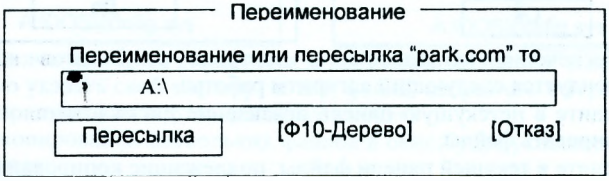
Пересылка и переименование файлов

Для *переименования* файлов необходимо выделить файл или группу файлов, нажать клавишу F6, ввести новое имя файла (или маску, при переименовании группы файлов) и нажать клавишу *Enter*. При переименовании файлов они остаются на своих местах, поэтому маршрут не указывается.

При переименовании файлов маршрут не указывается.

Процесс *пересылки* файлов с помощью VC происходит аналогично копированию. В случае пересылки одного файла или группы файлов в другой каталог в окне диалога необходимо указать номер диска и маршрут, куда следует переслать файлы. Если файл пересылается в другой каталог того же диска, то содержимое файла остается на прежнем месте, а в другой каталог включается только ссылка на этот файл (то есть элемент каталога), а из исходного каталога эта ссылка исключается. Такой способ пересылки работает очень быстро. При пересылке файла на другой диск он просто копируется с исходного диска на диск назначения, и после успешного окончания копирования исходный файл уничтожается. Нельзя переслать файлы с диска, защищенного от записи, так



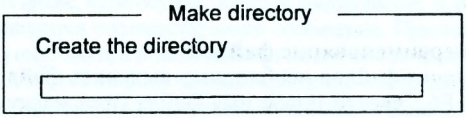


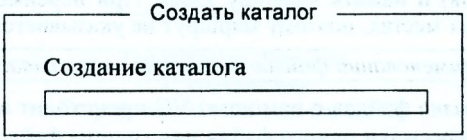
как файлы не могут быть удалены. В этом случае на экран выводится сообщение: Can't rename the file – Нельзя переслать файл.

Создание каталогов

Для создания каталога выведите в текущую панель оглавление диска и каталога, в котором будет создаваться новый каталог. Нажмите клавишу F7 - появляется окно диалога, в котором надо набрать имя каталога и нажать Enter.

Чтобы отменить создание каталога, следует нажать *Esc*. Если VC не может создать указанный каталог, он сообщает об этом (Can't create directory). В ответ на это сообщение надо нажать *Enter*. VC возвращается в исходное состояние. После этого необходимо выяснить причины невозможности создания каталога.

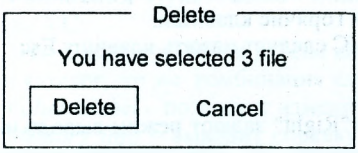


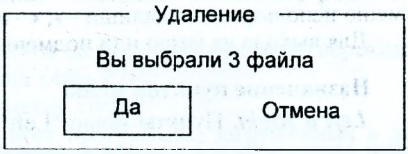


Причинами невозможности создания каталога могут быть, например: недопустимое имя каталога - файл или каталог с таким именем уже существует; в корневом каталоге диска недостаточно места; на диске недостаточно места.

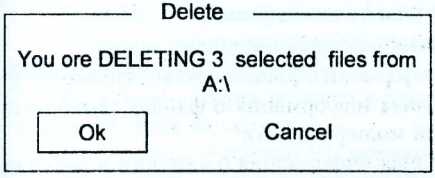
Удаление файла и каталога

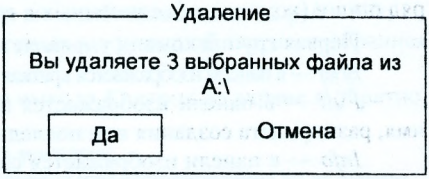
Для удаления файлов или каталогов необходимо выделить их и нажать F8. При этом на экране появляется окно, в котором указывается, что удаляется:





При удалении группы файлов после нажатия клавиши *Enter* появляется повторный запрос подтверждения на удаление:

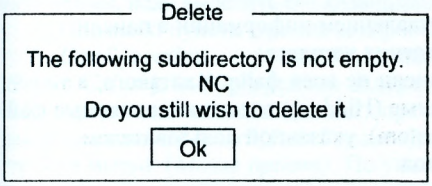


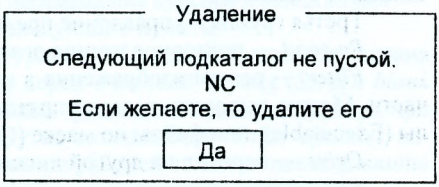


Можно нажать *Enter* для удаления файлов. Для отказа от удаления выберите команду *Cancel*. Отказаться от удаления можно также нажав клавишу *Esc*.

При удалении файла с атрибутом "только для чтения" или "спрятанный" на экран выводится дополнительный запрос на подтверждение удаления этого файла (The follaving file is markea read-only... Do still wisk...?) Если нужно удалить данный файл, нажмите Enter, иначе — Esc. Для выбора ответа можно использовать также пункты меню, как указано выше.

При удалении каталога на экран выводится сообщение, например: "Do you wish to delete the directory NC" – "Вы удаляете каталог NC". После подтверждения удаления программа проверяет, пустой ли каталог. Если удаляемый каталог не пуст, то на экран выводится предупреждение:





Для удаления каталога нажмите экранную кнопку *OK*, для отказа – клавишу *ESC*. После выполнения команды Volcov Commander возвращается к состоянию, которое было до нажатия клавиши *F8*.

Встроенное меню

Встроенное меню VC позволяет установить наиболее удобный вид представления информации на экране, изменить режим работы VC, а также выполнить некоторые другие действия. Для входа в меню следует нажать клавишу F9. Вверху экрана появится строка, содержащая пункты меню: *Left, Files, Commands, Option u Right.*

Один из пунктов меню является выделенным. Для выбора нужного пункта меню используются клавиши →, ← или горячие клавиши.

Для выхода из меню или подменю VC следует нажать клавишу Esc.

Назначение пунктов меню

Left и Right. Пункты меню "Left" и "Right" задают режим вывода информации соответственно в левой и правой панелях VC. Действующие режимы отмечены галочкой "√" или "±". Чтобы установить или отменить режим, надо выделить его и нажать *Enter*. Подменю, соответствующие Left и Right, содержат ряд опции (команд), объединенных в группы по назначению.

Первая группа команд управляет выводом информации на экран:

Brief — в панели изображается краткая информация о файлах (выводится только имя);

Full — в панели изображается полная информация о файлах (выводится имя, размер, дата создания или последней модификации);

Info — в панели изображается сводная информация о каталоге и диске на другой панели. Эту операцию можно выполнить комбинацией клавиш *Ctrl – L*, повторный ввод команды убирает информацию с экрана;

Tree — в панели изображается дерево каталогов на диске;

On/Off — выводится или не выводится на экран панель.

Вторая группа управляет сортировкой файлов и каталогов:

Name — файлы выводятся в алфавитном порядке;

eXtension — файлы выводятся так, что расширения имен файлов оказываются в алфавитном порядке;

tiMe — файлы выводятся в порядке убывания даты последней модификации, более новые файлы выводятся первыми;

Size — файлы выводятся в порядке убывания их размера;

Unsorted — файлы и каталоги выводятся в том порядке, в котором они записаны в каталоге.

Третья группа – управление представлением информации в панели:

Re-read — повторное чтение оглавления каталога;

Filter — режим изображения в панели не всех файлов каталога, а только части. Можно выводить только спрятанные (Hidden), только исполняемые файлы (Executable) или файлы по маске (Custom), указанной пользователем;

Drive — переход на другой дисковод.

Files. Пункт меню Files дает возможность производить различные операции с файлами.

Многие из этих операций закреплены за функциональными клавишами *F1 – F10*. Другие команды:

file Attributes — установка атрибутов файлов;

Send files — пересылка файла или группы файлов с помощью электронной почты;

select Group — выделение группы файлов по маске;

Unselect group — отмена выделения группы файлов по маске;

Invert group – инвертирование выделения группы файлов;

resTore selection - восстановление выделения группы файлов.

Commands. Пункт меню "Commands" позволяет выполнять различные команды VC:

Tree— вывод на экран дерева каталогов на диске для быстрого перехода в другой каталог, то же комбинация клавиш *(Alt – F10)*;

Volume label – позволяет изменить метку диска;

Memory Info – выводит справку о всех программах, загруженных в данный момент в ОЗУ;

Directory Sizes - подсчитывает размеры всех каталогов в текущей панели. Этот режим установлен по умолчанию;

Find file — поиск файла на диске, то же комбинация клавиш (*Alt – F7)*;

History — просмотр команд, выведенных в командной строке, выбранная команда может быть исполнена;

EGA lines — переключение в режим вывода 43 строк на экране и обратно, в режим 25 строк на экране;

Swap panels — панели VC меняются местами;

Panels on/off — удаление панелей VC с экрана или восстановление их на экране;

Compare directories — сравнение каталогов, изображенных на панелях VC. В каждом каталоге выделяются файлы, отсутствующие в другом каталоге или имеющие в другом каталоге иной размер или более раннюю дату последнего обновления;

send/Receive mail — позволяет переслать по электронной почте сообщения, указанные в выходном каталоге почты, и принять сообщения во входной каталог почты;

Commander mail — позволяет создавать новые сообщения для отправки по электронной почте и просмотреть сообщения, полученные с помощью электронной почты;

Menu file edit — редактирование списка команд меню пользователя;

Options. Пункт меню *Options* позволяет задавать конфигурацию VC, режим работы VC и указывает, какой редактор будет использоваться при редактировании файлов. Включенные режимы отмечены галочкой √. Чтобы установить или отменить режим, надо выделить его клавишами перемещения курсора и нажать *Enter*.

Подменю пункта Options содержит следующие пункты:

Configuration — установка конфигурации VC: установка цветов экрана (*Screen Colors*); выбор времени задержки перед очисткой экрана (*Screen blank delay*); установка способа вывода спрятанных файлов и смещения курсора при нажатии клавиши Insert (*Panel options*); другие возможности (*Other options*); настройка мыши (*Mouse options)*. По умолчанию рабочей является левая клавиша мыши. Для "левшей" можно сделать рабочей правую клавишу мыши.

advanced Options – дополнительная настройка параметров конфигурации;

Editor — указание редактора, вызываемого при нажатии клавиши *F4*.

eXtension file edit — редактирование файла *VC.EXT*, задающего действие VC при нажатии пользователем клавиши *Enter* в зависимости от расширения имени выделенного файла.

Auto menus — после выполнения одной из команд пользовательского меню, выводимого при нажатии клавиши *F2*, на экран опять выводится пользовательское меню;

Path prompt – установка формы приглашения операционной системы;

Key bar — вывод информации о назначении функциональных клавиш в последней строке экрана. Включить и выключить этот режим можно также нажав *Ctrl – B*.

Full screen — вывод панели VC размером в полный экран или в половину экрана;

auto Directory sizes – режим автоматического подсчета размера каталогов в панели;

Mini status — автоматический вывод в нижней части каждой панели строки мини-статуса;

Clock — вывод текущего времени в правом верхнем углу экрана;

Memory allocation - установка режима загрузки файла Command.com. Если режим установлен, то при загрузке программ в ОЗУ остается только резидентная часть (внутренние команды) командного процессора, в противном случае командный процессор сохраняется в ОЗУ полностью;

Save setup (Shift – F9) — автоматическое сохранение установленных режимов работы VC. При следующем запуске все его режимы будут такими же, как в момент выполнения команды Save setup.

Для получения подробной справки по каждой команде выделите ее в меню и нажмите *F1*.

Установка атрибутов файла

Volcov Commander позволяет изменить атрибуты файлов.

Для изменения атрибутов файла необходимо:

- выделить файл или группу файлов;
- ввести команды *F9*, *Files, file Attributes*;
- установить необходимые флажки с помощью клавиши пробел или мыши;
- введите команду Set (установить).

При изменении атрибутов у одного файла на экран будет выведено сообщение о текущих атрибутах файла. Знаком "x" будут помечены установленные атрибуты: "*Read only*" — только для чтения, "*Archive*" — файл не архивирован, "*Hidden*" — спрятанный файл, "*System*" — системный файл.

Для отказа от изменения атрибутов файла введите команду *Cancel*.

При изменении атрибутов у нескольких файлов на экран будет выведен запрос об атрибутах файлов. Если необходимо изменить атрибуты у выбранной группы файлов, то установите флажки требуемых атрибутов в колонке *Set*, а для снятия атрибутов – в колонке *Clear*.

Информационная панель

В панели VC можно вывести сводную информацию о диске и каталоге на другой панели. В верхней части информационной панели выводится строка "Info".

В панели изображаются следующие сведения:

- емкость оперативной памяти компьютера в байтах (...bytes Memory);
- количество свободной оперативной памяти в байтах (...bytes Free);
- емкость текущего диска в байтах (...bytes on drive);
- количество свободного места на текущем диске (...bytes free on drive...);
- количество файлов в каталоге, выделенном на другой панели VC, и их общий размер в байтах (...files use ...bytes in ...).

Ниже в информационной панели выводится содержание файла с именем DIRINFO.

Вывод на экран информационной панели осуществляется нажатием Ctrl - L. Для удаления информационной панели следует повторно нажать *Ctrl - L*.

5.4.2. Сервисная оболочка Windows Commander

Программа Volcov Commander имеет существенный недостаток: она не

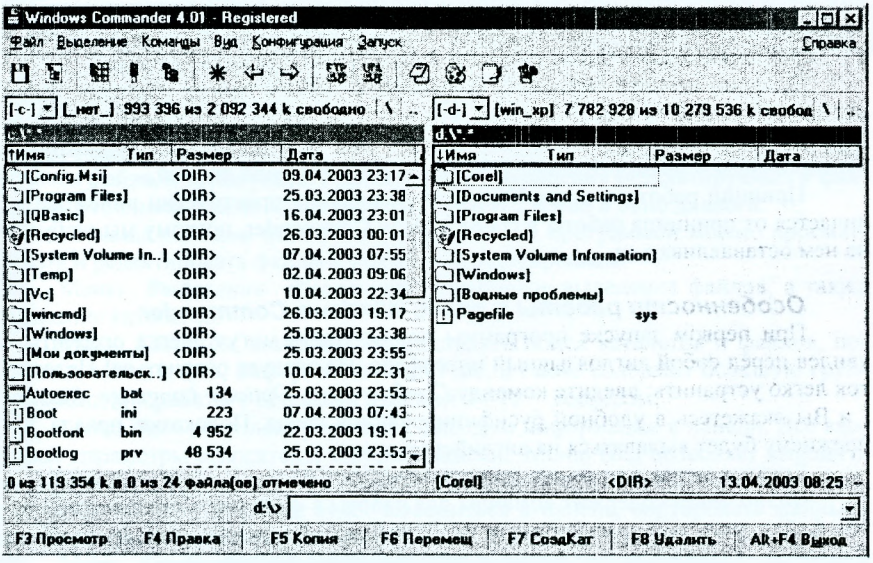


Рис. 5.4.2. Сервисная оболочка Windows Commander

может работать с длинными именами файлов. Для работы с длинными именами файлов была разработана программа Far. А вскоре появилась сервисная оболочка Windows Commander, которая разработана специально для работы с досовскими программами в среде Windows. Другой причиной разработки этой программы было, очевидно, то, что не все досовские программы запускаются из среды Windows 98 и с помощью программы Far. К достоинствам Windows Commander относится во-первых, то, что ее интерфейс похож на интерфейс уже упоминавшихся программ Volcov Commander и Far. Во-вторых, программа позволяет работать с длинными именами файлов. В третьих, она позволяет легко работать с архивными файлами, чего не позволяют делать упомянутые выше сервисные программы для работы с командами MS-DOS, а также программа Проводник. В архивы можно входить как в каталоги. Для управления в Windows Commander одинаково удобно можно пользоваться как мышью, так и клавиатурой.

Рабочее окно программы Windows Commander приведено на рис. 5.4.2. Окно программы оформлено в соответствии с принципами Windows: строка заголовка, главное меню, панель инструментов. Ниже панели инструментов расположены два открывающихся списка для выбора диска, выводимого в соответствующую панель, рядом со списком выведена метка диска, его размер и

размер свободного пространства на диске. Под списками расположены две панели: левая и правая. Ниже панелей имеется строка ввода команд и далее строка помощи.

Панели

Панели предназначены для вывода списков папок и файлов на дисках. В заголовке панели указано имя диска и маска, в соответствии с которой выводится информация в панель. В нижней части панели имеется строка состояния, в которой приведены полные сведения о выделенном файле или папке.

Одна из панелей активная, заголовок активной панели выделен синим цветом. Переход из одной панели в другую осуществляется либо щелчком мыши по нетекущей панели, либо клавишей Tab. В активной панели находится курсор панели – прямоугольник, отмеченный пунктирной линией.

Принцип работы в среде Windows Commander практически ничем не отличается от принципа работы в среде Volcov Commander, поэтому мы не будем на нем останавливаться.

Особенности работы в среде Windows Commander

При первом запуске программы пользователи могут слегка огорчиться, увидев перед собой англоязычный интерфейс. Не следует огорчаться! Недостаток легко устранить: введите команду *Configuration, Options, Language, Russian,* - и Вы окажетесь в удобной русифицированной среде. Подсказка, правда, попрежнему будет выдаваться на английском языке.

Главное меню постоянно присутствует на экране.

Меню ***Файл*** содержит команды для работы с файлами: установка атрибутов файла, упаковка, распаковка и проверка архива, печать файлов, разбивка файлов и их сборка, кодирование и декодирование файлов.

Команда *Упаковать* позволяет упаковать выделенный файл с помощью одного из архиваторов (рис. 5.4.3). Архиваторы должны быть на диске и маршруты их поиска должны быть прописаны в файле autoexec.bat. Аналогично выполняется и распаковка. При этом, по возможности, используется внутренний распаковщик.

Команда *Разбить* позволяет разбить длинный файл на части (рис. 5.4.4).

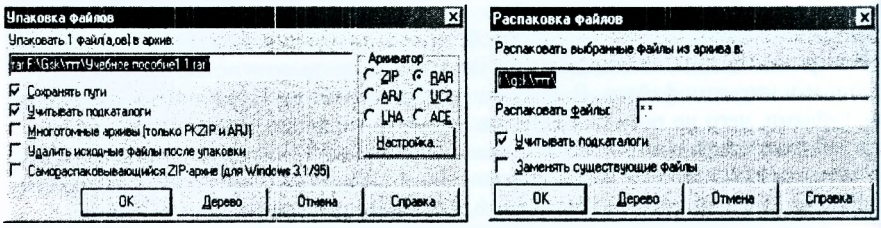


Рис. 5.4.3. Упаковка и распаковка файлов

При этом создается заголовочный файл с расширением .CRC и его части с размером, указанном в окне диалога. Части разбитого файла имеют то же имя, что

и заголовочный файл, и расширение .001, .002 и т. д. *Части разбитого файла не редактируются.*

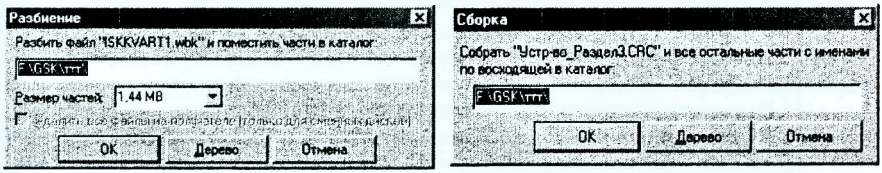


Рис. 5.4.4. Разбивка и сборка файла

Для сборки файла необходимо выделить заголовочный файл и ввести команду *Файл, Собрать файл* или просто щелкнуть дважды по имени заголовочного файла, а затем указать маршрут для сохранения результирующего файла. Для этой цели можно использовать команду *Дерево* в окне диалога.

Команда *Связать* позволяет указать, какой программой можно просматривать и редактировать файлы с указанным расширением.

Меню ***Выделение*** обеспечивает групповое выделение файлов, а также сравнение каталогов.

Меню ***Команды*** позволяет осуществлять поиск каталогов и файлов, получение метки диска, информации о системе, а также содержит большую группу команд для работы с сетевыми дисками и FTP – сервером.

Меню ***Вид*** позволяет управлять выводом информации в панели. Все вводимые параметры относятся к текущей панели. Меню Вид позволяет выводить информацию о файлах в краткой или полной форме, устанавливать типы файлов, информация о которых будет выводиться в панель, сортировать файлы и изменять порядок сортировки.

Меню ***Конфигурация*** позволяет осуществлять разнообразные настройки рабочей среды. Отметим лишь одну из них – управление выводом скрытых или системных файлов: ввести команду *Конфигурация, Настройка, Экран* и в группе *Отображение файлов* установить или снять флажок *Показывать скрытые / системные файлы*. Закладка *Правка/Просмотр* позволяет указать редактор, с помощью которого будут просматриваться или редактироваться выделенные файлы.

Меню ***Запуск*** позволяет производить настройку меню запуска: добавлять или удалять команды. Опция *Изменить меню запуск* позволяет добавлять или удалять команды из меню запуска. Опция *Изменить главное меню* содержит команды для перевода описания интерфейса на национальные языки.

Контрольные вопросы

1. Поясните, чем отличается отображение каталогов и файлов в панелях VC.
2. Какая информация о файле выводится при полной и краткой форме представления каталога?
3. Какая панель VC является текущей?
4. Как перевести курсор в другую панель?
5. Что означает выражение “выделить файл”, “выделить каталог”?
6. Как "войти" в каталог? Как "выйти" из каталога?
7. Поясните порядок выполнения команд из среды VC.
8. Поясните порядок запуска исполняемых и командных файлов.

9. Выполните команду форматирования диска A:
10. Выведите для редактирования файл autoexec.bat.
11. Выведите в правую панель оглавление диска A:.
12. Что такое встроенное меню и пользовательское меню?
13. Выведите в левую панель каталог в полной форме по алфавиту, по ту, по расширению имени файла, по дате изменения.
14. Выведите в правую панель каталог в краткой форме.
15. В чем отличие Windows Commander от Volcov Commander?
16. Как осуществляется архивация файлов в Windows Commander?
17. Как установить атрибуты файлов в Windows Commander?
18. Как просмотреть спрятанные или системные файлы в Volcov Com- Commander и в Windows Commander?

6. Дополнительные сведения о работе на персональном компьютере

6.1. Архивация файлов

6.1.1. Понятие об архивации файлов

При эксплуатации компьютера по самым разным причинам возможна порча или потеря информации на магнитных дисках: неправильная корректировка или случайное уничтожение файлов, разрушение информации компьютерным вирусом и тому подобное. Для полного или частичного восстановления потерянной информации необходимо иметь копии используемых файлов и систематически обновлять копии изменяемых файлов. Копии файлов создаются на дискетах или магнитных лентах, при этом они могут храниться в обычном или сжатом виде.

Для создания копий файлов или дисков используются команды ОС Copy, XCopy и др. Для хранения копий файлов в сжатом виде используются специальные программы архивации файлов.

Файлы операционной системы MS-DOS IO.SYS и MSDOS.SYS, а также файлы пакетов программ, защищенных от копирования, не могут быть заархивированы стандартным образом. Операционную систему: файлы IO.SYS, MSDOS.SYS, COMMAND.COM, а также файлы с расширением .SYS целесообразно хранить на системной дискете в обычном виде.

Архивный файл представляет собой набор из одного или нескольких файлов, помещенных в сжатом виде в единый файл, из которого их можно извлечь, при необходимости, в первоначальном виде. Архивный файл содержит оглавление, позволяющее узнать, какие файлы содержатся в архиве, а также код циклического контроля для каждого файла, позволяющий проверить целостность архива.

Процесс создания архива называется *архивированием*, процесс восстановления файла в первоначальном виде – *разархивированием*. Упакованный (сжатый) файл принято называть *архивом*.

Архивация (упаковка) информации – это такое преобразование информации, при котором объем информации в файле уменьшается при неизменном количестве информации.

Степень сжатия информации зависит от типа архивируемого файла и от используемого метода сжатия. Степень сжатия характеризуется коэффициентом сжатия K_c, который определяется как отношение объема сжатого файла к объему исходного файла, выраженного в процентах.

Все используемые методы сжатия можно поделить на две группы: методы сжатия без потери информации и методы сжатия с потерей информации. При использовании первой группы методов информация восстанавливается полностью, при использовании второй группы методов при восстановлении информации они не могут быть восстановлены в первоначальном виде. Очевидно, что для сжатия текстовых файлов можно использовать только методы сжатия без потери информации. При архивации графических файлов можно использовать любые методы, так как даже в случае потери части информации это может отразиться только на качестве воспроизведенного рисунка.

Для архивации без потери качества используют две метода: метод Хаффмана и RLE-кодирование путем учета повторений.

Метод Хаффмана основан на том, что частота повторения символов в тексте различная. Поэтому можно заменить символы вспомогательным кодом. Чем больше частота повторения символов, тем меньше должна быть длина кода. Кодовая таблица, образовавшаяся при архивации, хранится вместе с архивным файлом.

Второй метод основан на замене повторяющихся последовательностей байтов одной последовательностью с указанием числа повторений.

6.1.2. Программы для архивации файлов

Существует много программ для архивации файлов. Как правило, эти программы позволяют помещать копии файлов на диск в сжатом виде в архивный файл, извлекать файлы из архива, просматривать оглавление архива и так далее. Разные программы отличаются форматом архивных файлов, скоростью работы, степенью сжатия файлов при помещении в архив, удобством использования. Широко распространены программы PKZIP, PKUNZIP, ARJ, ICE, LHA, RAR.

Программа архивации ARJ

Программа ARJ (версия 2.42), известна как одна из лучших по набору функций, предоставляемых пользователю, степени сжатия и скорости работы. Особенно эффективна программа ARJ при работе с текстовыми файлами и файлами баз данных.

Формат команды:

ARJ <команда> [ключ] <имя-архива> [<список файлов>]]

При вводе команды *ARJ* без параметров на экран выводится список команд и ключей программы.

Команды:

a - добавление файлов в архив;

d - удаление файлов из архива;

e - извлечение файлов из архива в текущий каталог;

f - добавление в архив только новых файлов;

l - просмотр списка содержания архива без указания пути к файлам;
m - перенос файлов в архив;
t - проверка целостности архива;
u - изменить файлы в архиве;
v - просмотр списка архива с указанием пути к файлам;
w - найти текстовую строку в архиве;
x - извлечение файлов с полным именем (восстанавливает структуру каталога);
y - копирование архива с новыми параметрами.

Ключи (опции) (приведена лишь часть ключей):
c — извлечение файлов с указанным временем создания;
d — извлечение с удалением;
e — исключить маршрут из имени файла;
f — обновить существующие файлы;
g — защита создаваемого архива паролем:
g <пароль> - пароль вводится в командной строке;
g? - ввод невидимого пароля;
je – создание самораспаковывающегося архива;
jp – пауза при просмотре содержимого архива после заполнения экрана;
m — указание метода сжатия (0, 1, 2, 3, 4):
m0 – без сжатия; m1 – нормальное сжатие (по умолчанию); m2 – наибольшее сжатие; m3 - быстрое сжатие меньшая степень сжатия (компрессия); m4 – самое быстрое сжатие и наименьшая компрессия;
n — только новый файл;
r — добавление файлов из текущего каталога и всех вложенных каталогов и подкаталогов;
u — изменить файл (новые и обновленные);
v — создание многотомного архива;
w — указать каталог, в который будут помещены временные файлы во время работы;
x — добавление/замена файлов, за исключением файлов, имена которых указаны за ключом;
y — с подтверждением запроса на извлечение, предполагается ответ Yes.

Примеры команд

Так как файлы архивации используются достаточно часто, целесообразно путь к файлам программ архивации указать в файле автозапуска.

Архивация файлов:

C:\>arj a D:\ARCHIV\new.arj E:\NEW

Производится упаковка каталога NEW с диска E: с использованием полных имен файлов. Архивный файл помещается на диск D: в каталог ARCHIV, имя архивного файла new, а расширение имени файла - arj;

C:\>ARJ a D:\ARCHIVL\lex.arj E:\LEX*.*

архивируются все файлы из каталога LEX диска E: и помещаются в файл lex.arj каталога ARCHIVL диска D:;

C:\>ARJ a -r -v A:sc5.arj E:\SC5*.*

архивируются все файлы из каталога SC5 и всех его подкаталогов (режим -r) диска E:, создается многотомный архив на диске A: (режим -v). После заполнения одной дискеты выдается запрос на установку следующей дискеты.

Разархивация файлов:

ARJ <команда> [-режимы] <имя-архива> [каталог\] [имена-файлов]

Здесь [каталог\] - каталог, куда будут помещаться файлы, извлекаемые из архива, [имена-файлов] – файлы, извлекаемые из архива.

C:\>ARJ e D:\ARCHIVL\lex E:\LEX\

извлекается архивный файл lex.arj из каталога ARCHIVL диска D:. Извлекаемые файлы помещаются в каталог LEX диска E:

C:\>ARJ e –v A:\lex.arj E:\LEX\

Производится разархивация (распаковка) многотомного архива lex.arj, расположенного на дискетах, в каталог LEX диска E:

Для ускорения процесса разархивации целесообразно скопировать архивированный файл с дискеты в корневой или требуемый каталог диска E:, а затем разархивировать файл командой, например:

C:\>ARJ x E:\LEX\Sc5.arj E:\LEX\

извлекается файл Sc5.arj с восстановлением всех каталогов и подкаталогов (команда x). Извлекаемые файлы помещаются в каталог LEX диска E:. Если нужных подкаталогов нет на диске, то выдается запрос на их создание.

Программа архивации RAR

Другой популярной программой архивации является программа RAR. Главными особенностями архиватора RAR являются следующие:

- работает в двух режимах: полноэкранного интерактивного интерфейса и обычного интерфейса командной строки.
- работает с архивами, созданными другими архиваторами (ZIP, ARJ, LZH);
- использует высокоэффективный метод сжатия solid (на 10 – 15 % выше, чем обычно);
- позволяет добавлять файловые и архивные комментарии;
- имеется возможность частичного восстановления поврежденных архивов;
- имеется защита архива от изменения;
- позволяет просматривать архивные файлы в упакованном виде.

Режим интерфейса командной строки

Принцип работы архиватора в режиме интерфейса командной строки аналогичен принципу работы с архиватором ARJ.

Полноэкранный интерактивный режим

Запуск программы. Запуск программы в полноэкранном режиме осуществляется из командной строки командой RAR без параметров. После запуска программы на экране появляется окно (рис. 6.1.1.)

Левая панель содержит список файлов и каталогов, как в среде Volcov Commander. Правая панель содержит два раздела: память – Memory и установ-

ки – Settings. Внизу экрана размещается строка помощи. При нажатии Alt выводится подсказка о дополнительных функциях, вызываемых одновременным нажатием клавиши Alt и одной из функциональных клавиш. При нажатии F1 на экран выводится информация о назначении всех функциональных клавиш и их комбинаций (табл. 6.1.1, 6.1.2).

При выделении архивного файла и нажатии клавиши Enter программа переходит в сам архив, как в каталог, то же самое что при вводе команды

RAR <имя архивного файла>

После входа в архив работа с ним осуществляется как в обычном каталоге. В списке архивных файлов имена файлов, закодированные паролем, отмечаются звездочкой "*". На правой половине экрана выводятся сведения об архиве. Здесь могут использоваться символические обозначения:

⇒ - файл продолжается в следующем томе;

⇐ - файл начинается в предыдущем томе.

Технология работы с архивом

Просмотр архивного файла

Для просмотра архивного файла из среды MS-DOS введите команду:

RAR < имя файла>

S:\VC				
Name	Size	Date	Time	RAR archiver Version 2.04 shareware
..	<UP---DIR>			
arcview exe	36596	24-05-94	21:00	
arcview ext	609	10-12-96	20:29	——— Memory ———
arcview ini	319	24-09-96	21:25	
btmload exe	36424	21-05-93	16:05	Memory in use 388 Kb
cview cfg	23	11-03-96	20:52	Extract from archive Enabled
cview exe	261748	07-01-93	12:00	Add to archive Enabled
dbview exe	20473	15-05-92	18:26	Add to solid archive Enabled
gifprt exe	29351	12-10-92	21:1	Update Solid archive Enabled
hiew exe	79504	10-04-97	10:1	——— Settings ———
hiew hlp	21235	06-02-97	16:2	Password Absent
hiew ini	1604	07-02-97	08:4	Compression level Normal
hiew xlt	1584	29-01-97	11:0	Multimedia compression Disabled
libview bat	110	09-12-96	19:5	Add recover record Disabled
libview exe	31192	16-10-90	19:5	Add AV to archives Disabled
limview bat	109	09-12-96	19:5	Make solid archives Always
lxdump exe	53869	19-12-96	09:0	Log errors to file Disabled
me bat	38	09-12-96	19:5	Default comment file Absent
..				
1Help 2Solid 3View.. 4SFX 5SFXVol 6SolVol 7SolSVl 8Repair 9SolSFX 10Quit				

Рис.6.1.1. Полноэкранный режим работы программы RAR

После ввода команды открывается окно программы RAR, в левую панель которого выводится список архивных файлов, а в правую - сводные данные об архивном файле. Для просмотра нужного файла выделите его курсором и нажмите клавишу *F3*. Для выхода из режима просмотра нажмите *Esc*.

Создание архива

1. Выбрать диск: *Alt-D*.
2. Отсортировать файлы: *F9, Configuration, Sort Order*, установить флажок требуемого способа сортировки.
3. Выделить архивируемый файл или группу файлов.
4. Установить пароль: *Alt-P* (пароль вводится дважды).
5. Установить метод сжатия: *Alt-M*.
6. Создать архив:

   F2 – обычный;

   Alt-F2 – архив типа solid;

   Alt-F5 – самораспаковывающийся (SFX) архив, ввести предельный размер архива в килобайтах.
7. В окне диалога ввести имя архивного файла и путь к нему. Для перемещения файлов в архив установить флажок Move в окне.
8. Нажать *Enter*.

Таблица 6.1.1

Назначение функциональных клавиш при работе с файлами

Клавиши или их комбинации	*Комментарий*
F1	*Help* – помощь
F2	*Add* – добавить файлы в архив
F3	*View* – просмотр
F4	*Fresh* – обновить файлы
F5	*Volume* – создать архивные тома
F6	*Move* – переместить файлы в архив
F7	*Update* – добавить новые файлы и обновить старые
F8	*Repair*- восстановить испорченный архив
F9	*Option* – конфигурация/сохранение конфигурации
F10	*Quit* – выход
Alt – F2	*Solid* – создать непрерывный (solid) архив
Alt – F3	*View* – просмотр файла
Alt – F5	*SFXVol* – создать архив, разбитый на SFX-тома
Alt – F6	*SolVol* – создать *solid* -архив, разбитый на тома
Alt – F7	*SolVl* – создать *solid* -архив, разбитый на SFX-тома
Alt – C	Цветной/черно-белый режим
Alt – D	Выбор текущего диска
Alt – J	Временный выход в DOS
Alt – M	Выбор метода сжатия
Alt – P	Ввод пароля
Alt – S	Запись текущих опций
Alt – W	Назначение рабочего каталога для временных файлов

Таблица 6.1.2

Назначение функциональных клавиш при работе с архивом

Клавиши или их комбинации	*Комментарий*
F1	*Help* – помощь
F2	*Test* – проверка сохранности архива
F3	*View* – просмотр
F4	*Extr* – извлечение файлов из архива с полным путем
F5	*Comment* – добавить комментарии к архиву
F6	*ExCurD* – извлечь файлы в текущий каталог
F7	*SFX* – преобразовать в SFX-архив (самораспаковывающийся)
F8	*Delete* – удаление файлов из архива
F9	*Option* – конфигурация/сохранение конфигурации
F10	*Quit* – выход из архива
Alt – F3	*View* – просмотр файлов встроенной или внешней программой
Alt – F4	*Extr To* – извлечь файлы в указанный каталог
Alt – F5	*FilCmt* – добавить комментарии к файлам
Alt – F7	*Lock* – заблокировать архив от изменений

Внимание! Архивация с паролем-опасная операция. Если пароль забыт, то восстановить файл не удастся.

Извлечение файлов из архива

1. Выбрать диск: *Alt-D.*
2. Войти в требуемый каталог.
3. Выделить архивный файл и нажать Enter – выводится список файлов архива.
4. Выделить файлы, подлежащие извлечению.
5. Нажать:

F4 – для извлечения файлов с восстановлением структуры каталогов и подкаталогов;

F6 – для извлечения в текущий каталог;

Alt – F4 – для извлечения файлов в заданный каталог.

Если файлы были защищены паролем, открывается окно диалога для ввода пароля.

Программа архивации WinRAR

Для архивации файлов в среде Windows разработаны две программы

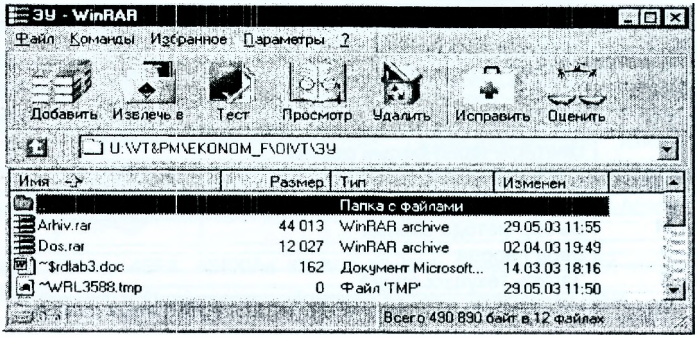


Рис. 6.1.2. Архиватор WinRar

WinZip и WinRAR. Программа WinRAR имеет более удобный интерфейс (рис. 6.1.2). Алгоритм работы при архивации и разархивации файлов также предельно прост.

Алгоритм архивации файлов:

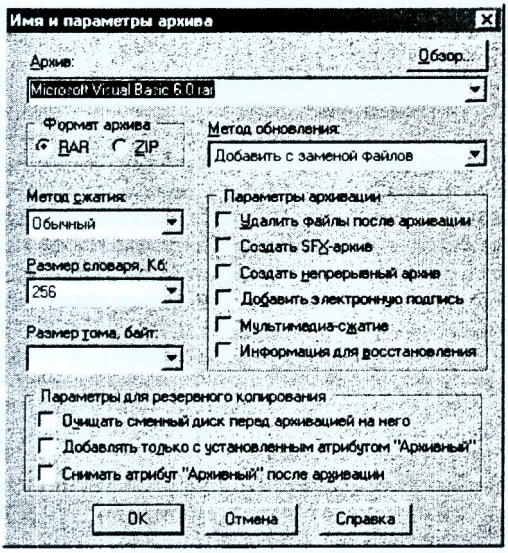


Рис.6.1.3. Настройка параметров архива

- выделить в окне архивируемый файл (папку), используя раскрывающийся список адресов;
- щелкнуть по кнопке *Добавить* – открывается окно диалога (рис.6.1.3);
- установить требуемые атрибуты архивации и указать в раскрывающемся списке маршрут, куда следует поместить архивный файл, используя кнопку *Обзор*. Имя архивного файла можно принять по умолчанию;
- после настройки параметров архива и маршрута щелкнуть по клавише *ОК*.

Алгоритм извлечения файлов

Алгоритм извлечения файлов из архива также прост:

- выделить в окне диалога (рис. 6.1.2) архивный файл;
- щелкнуть мышью по кнопке *Извлечь в ...;*
- в открывшемся окне диалога "Путь и параметры извлечения" укажите маршрут, куда следует извлечь файлы из архива, и настройте параметры для извлечения файла из архива;
- щелкните по кнопке ОК.

6.2. Защита от компьютерного вируса

6.2.1. Понятие о компьютерном вирусе

Компьютерный вирус — это специально написанная небольшая по размерам программа, которая может "приписывать" себя к другим программам (то есть "заражать" их), создавать свои копии и внедрять их в файлы, системные области компьютера и в вычислительные сети, а также выполнять различные нежелательные действия на компьютере.

Основные пути проникновения вируса в компьютер:

- загрузка компьютера с дискеты, содержащей вирус;
- из сети, при записи файлов с других компьютеров.

Заражение дискеты может произойти:

- при записи на нее зараженного файла;
- при перезагрузке компьютера при нахождении в дисководе А: дискеты.

Зараженный диск – это диск, в загрузочном секторе которого находится программа – вирус.

Зараженная программа – это программа, содержащая внедренную в нее программу-вирус. Когда такая программа начинает работу, то сначала управление получает вирус. Вирус находит и "заражает" другие программы, а также выполняет какие-нибудь вредные действия, например, портит файлы или таблицу размещения файлов на диске, "засоряет" оперативную память и так далее. Для маскировки вируса действия по заражению других программ и нанесению вреда могут выполняться не всегда, а при определенных условиях. После того как вирус выполнит свои действия, он передает управление той программе, в которой находится, и она работает так же, как обычно. Тем самым внешне работа зараженной программы выглядит так же, как и незараженной. Последствия заражения программ вирусом могут быть очень серьезными, вплоть до разрушения баз данных (баз знаний), нарушения работы сложных вычислительных систем. Так, например, в начале 1989 г. вирусом, написанным американским студентом Моррисом, были заражены и выведены из строя тысячи компьютеров, в том числе принадлежащих Министерству обороны США.

Внешними проявлениями зараженности программы вирусом могут быть: замедление времени работы, сообщения о недостатках памяти, программа не находит нужный файл, могут быть и другие непонятные явления.

Заражение компьютера вирусом происходит, как правило, при использовании "чужих" дискет или через компьютерную сеть. Каждая разновидность вируса может заражать только один или два типа файлов. Чаще всего встречаются вирусы, заражающие COM-файлы, на втором месте по распространенности вирусы, заражающие EXE-файлы, и вирусы, заражающие COM-файлы и EXE-файлы. Иногда компьютеры заражаются вирусами, распространяющимися через загрузочные сектора дискет.

Классификация программ-вирусов

Программы- вирусы классифицируют по следующим признакам:

- *по "среде обитания"*: сетевые, файловые, макровирусы, загрузочные, файловозагрузочные;
- *по способу заражения* программ: резидентные и нерезидентные;
- *по степени воздействия*: неопасные, опасные, очень опасные;
- *по особенностям алгоритмов*: паразитические, репликаторы, невидимки, мутанты, троянские.

Сетевые вирусы распространяются по различным компьютерным сетям.

Файловые вирусы внедряются главным образом в исполняемые модули, т. е. файлы, имеющие расширение *COM* и *EXE*. Файловые вирусы могут внедряться и в

другие типы файлов, но, как правило, записанные в таких файлах, они никогда не получают управление и, следовательно, теряют способность к размножению.

Макровирусы – это файловые вирусы, использующие особенности документов подготовленных редакторами текстов, электронных таблиц, систем управления базами данных. В этих файлах могут содержаться программы на макроязыках, которые позволяют создавать программы-вирусы.

Загрузочные вирусы внедряются в загрузочный сектор дисков или в сектор, содержащий программу загрузки системного диска.

Файловозагрузочные вирусы заражают как файлы, так и загрузочные сектора дисков.

Резидентные вирусы при заражении (инфицировании) компьютера оставляют в ОЗУ свою резидентную часть, которая потом перехватывает обращения операционной системы к объектам заражения (файлам, загрузочным секторам дисков и т. п.) и внедряется в них. Резидентные вирусы находятся постоянно в оперативной памяти и остаются активными до момента выключения питания компьютера.

Нерезидентные вирусы не заражают оперативную память компьютера и являются активными ограниченное время, время работы зараженной программы.

Неопасные вирусы не мешают нормальной работе компьютера, но уменьшают объем свободной оперативной памяти на дисках. Действие таких вирусов проявляется в появлении каких-нибудь графических или визуальных эффектов.

Опасные вирусы могут приводить к различным нарушениям в работе компьютера.

Очень опасные вирусы приводят к порче программы, уничтожению данных, стиранию информации в системных областях дисков.

По особенностям алгоритма вирусы трудно классифицировать из-за большого разнообразия.

Паразитические вирусы – простейшие. Они изменяют содержимое файлов и секторов дисков. Эти вирусы легко обнаруживаются и уничтожаются.

Вирусы-репликаторы (черви) распространяются по компьютерным сетям. Они вычисляют адреса сетевых компьютеров и записывают по этим адресам свои копии.

Вирусы-невидимки (стелс-вирусы) очень трудно обнаружить и уничтожить, так как они перехватывают обращения операционной системы к пораженным файлам и секторам дисков и подставляют вместо своего тела незараженные участки диска.

Вирусы-мутанты содержат алгоритмы шифровки-расшифровки, благодаря которым копии одного и того же вируса не имеют ни одной повторяющейся цепочки байтов. Из-за этого свойства они очень трудно обнаруживаются.

Троянские программы-вирусы очень опасны, хотя и не способны к саморазмножению. Такие вирусы, маскируясь под полезную программу, разрушают загрузочные сектора дисков и файловую систему дисков.

6.2.2. Профилактика против заражения вирусом

Для предотвращения заражения ПЭВМ компьютерным вирусом, а также облегчения процедуры "лечения" больного компьютера необходимо выполнять некоторые профилактические мероприятия:

- создание архивных копий информации и дискет с программными продуктами. Необходимо периодически архивировать те файлы, которые Вы создали или изменили. Перед архивацией файлов целесообразно выполнять программу для ранней диагностики наличия вируса, чтобы убедиться в отсутствии вируса в компьютере и избежать помещения испорченных или зараженных файлов в архив;
- разграничение доступа к данным. На жестком диске целесообразно создать логический диск, защищенный от записи, и поместить в него программы и данные, которые надо только использовать, но не изменять;
- защита дискет от записи. Все архивные дискеты с программами и данными необходимо защищать от записи;
- использование для перезагрузки компьютера с дискеты только защищенной от записи эталонной дискетой с операционной системой. Во избежание заражения вирусом, распространяющимся через загрузочные секторы дискет, перед перезагрузкой компьютера с жесткого диска убедитесь в отсутствии дискеты в дисководе A:;
- использование резидентных программ-фильтров для защиты от вируса либо постоянно, либо тогда, когда это возможно;
- проверка целостности программ и данных каждый раз в начале работы с компьютером, что позволит выявить наличие вируса на раннем этапе. Для этого следует включать в файл автозапуска AUTOEXEC.BAT программы-ревизоры;
- проверка чужой дискеты на отсутствие вируса перед ее загрузкой в компьютер.

6.2.3. Программы для борьбы с компьютерным вирусом

Число вирусов постоянно растет. Появление самих вирусов и их рост (как количественный, так и качественный) связан с несовершенством как человеческой личности, так и самого общества, общественно-экономических отношений, продуктом которых является человек как личность.

Наличие яда, как известно, всегда вызывает появление противоядия, в данном случае – программ для борьбы с вирусом, число которых также растет. Имеется несколько групп программ для борьбы с компьютерным вирусом: программы-детекторы, программы-доктора, программы-ревизоры, доктора-ревизоры, программы-фильтры, программы-вакцины (иммунизаторы).

Программы-детекторы позволяют обнаруживать файлы, зараженные каким-либо одним или несколькими известными вирусами. Эти программы проверяют, имеется ли в ОЗУ и файлах на указанном пользователем диске спе-

цифическая для данного вируса комбинация битов (сигнатуры вирусов). При ее обнаружении на экран выводится соответствующее сообщение. Многие программы-детекторы могут не только обнаруживать, но и лечить зараженные файлы, настраиваться на новые типы вирусов, если указана комбинация битов, присущая данному вирусу. Наиболее известны программы-детекторы Aidstest, Scan, Norton AntiVirus, Droctor Web, AVSP. Такие программы, как Aidstest и AntiVirus, могут обнаруживать более 1000 вирусов.

Программы-ревизоры самое надежное средство защиты от вирусов. Они сначала запоминают сведения о состоянии программ и системных областей дисков тогда, когда компьютер не заражен вирусом, а затем периодически или по запросу пользователя сравнивают текущее состояние с исходным. При выявлении несоответствия выдается сообщение пользователю. Программы-ревизоры запускаются при каждом включении компьютера и позволяют выявить наличие вируса на ранней стадии, когда он не нанес еще большого вреда. Проверка осуществляется путем подсчета контрольной суммы файла и сравнения ее с контрольной суммой исходного файла. Эта проверка занимает много времени, поэтому такой проверке подвергаются наиболее важные файлы. Для остальных программ проверяется их размер, указанный в каталоге. Программы-ревизоры могут обнаруживать вирусы-невидимки.

К программам-ревизорам относятся такие программы, как ADinf фирмы "Диалог-Наука", CRCLIST, CRCTEST,.

Программы-доктора «лечат» зараженные программы или диски, «выкусывая» из зараженных программ тело вируса, т.е. восстанавливая программу в том состоянии, в котором она находилась до заражения вирусом. Основной недостаток программ-докторов – их узкая специализация. Доктор, ориентированный на некоторый фиксированный набор вирусов, не в состоянии будет «вылечить» файлы, зараженные другими типами вирусов, кроме того «доктора» лечат не всегда правильно. Некоторые программы, например, AVSP фирмы "Диалог-МГУ", могут обучаться не только способам обнаружения, но и способам лечения новых вирусов.

Программы доктора-ревизоры - это гибрид программ ревизоров и докторов, т.е. программы, которые не только обнаруживают изменения в системных файлах, но и могут, в случае изменений, автоматически вернуть их в исходное состояние. Доктора-ревизоры обнаруживают "нападение" вируса и лечат зараженные программы, причем, в отличие от программ-докторов, лечат правильно. Доктора-ревизоры обеспечивают защиту от 90% 95% вирусов. К докторам-ревизорам относятся программы ADinf+ADinfExt фирмы "Диалог-Наука" и комплексная антивирусная система AVSP фирмы "Диалог-МГУ".

Программы-фильтры - это резидентные программы для защиты от вирусов. Они перехватывают те обращения к операционной системе, которые используются вирусами для размножения и нанесения вреда, и сообщают об этом пользователю. Пользователь может разрешить или запретить выполнение соответствующей операции. К таким подозрительным операциям относятся, например:

- изменения .COM и .EXE файлов;
- снятие атрибутов только для чтения;

- прямая запись на диск по абсолютному адресу;
- запись в загрузочный сектор диска;
- загрузка резидентной программы.

Эти программы также не обеспечивают стопроцентной защиты, так как некоторые вирусы используют для своего размножения непосредственное обращение к программам операционной системы, а не используют стандартные прерывания. Кроме того, программы-фильтры не обнаруживают вирусы, которые распространяются через загрузочные сектора.

К программам-фильтрам относится, например, программа VSave, входящая в состав пакета утилит ОС MS-DOS.

Программы-вакцины, или иммунизаторы – это резидентные программы. Они модифицируют программы и диски таким образом, что это не отражается на работе программы, но тот вирус, от которого производится вакцинация, считает эти программы или диски уже зараженными. Очевидно, что нельзя провести вакцинацию программ сразу от нескольких типов вирусов, поэтому такие программы неэффективны и не получили широкого распространения.

Среди популярных и эффективных антивирусных программ, работающих под управлением операционных систем Windows, выделяют: антивирус Касперского – AntiViral Toolkit Pro (AVP), F-Secure Anti-Virus, McAfee VirusScan, Norton AntiVirus, Panda Antivirus, Sophos Anti-Virus; Trend Micro PC-Cillin.

6.2.4. Действия при заражении компьютера вирусом

При заражении компьютера вирусом или подозрении на заражение вирусом рекомендуется следующий порядок действий:

1. Выключить компьютер, чтобы приостановить дальнейшее распространение вируса.

2. Перезагрузить ОС с защищенной дискеты, содержащей программу фильтр, если такая имеется. Это позволит ликвидировать последствия заражения вирусом с наименьшими потерями.

3. Выполнить программу SETUP, если она имеется в ОС, для проверки правильности установки параметров конфигурации компьютера. Если параметры установлены неправильно, то переустановить их.

4. Если сохранялись сведения о длинах и контрольных суммах файлов для программы-ревизора, то запустить эту программу для диагностики изменений в файлах. Это позволит установить, какие файлы были заражены или испорчены вирусом.

5. Определить, имеются ли для того вируса, которым заражен компьютер, программы-детекторы (доктора) для его обнаружения и уничтожения. Чтобы найти нужную программу, необходимо поочередно запускать имеющиеся программы-детекторы для проверки того диска, на котором находятся зараженные вирусом файлы. Сначала имеет смысл запускать программы, обнаруживающие сразу несколько вирусов, например, AIDSTEST или SCAN.

6. После обнаружения вируса следует последовательно обезвредить все диски, которые могли подвергнуться заражению вирусом. Если на диске нет нужных файлов, копий которых нет в архиве, то рекомендуется заново отформатировать диск, а затем восстановить их с архивных дисков.

7. В случае, если на диске есть нужные файлы, для которых нет копий в архиве, необходимо выполнить следующие действия:

а) проверить целостность файловой системы на диске с помощью программы chkdsk с ключом /f, например, chkdsk A: /f.

Если повреждения файловой системы значительные, то целесообразно скопировать нужные файлы, копий которых нет в архиве, на дискеты и заново отформатировать диск. Если диск имеет сложную файловую структуру, то можно попробовать откорректировать ее с помощью программы NDD из комплекта Norton Utilities;

б) удалить с диска все ненужные файлы, а также файлы, копии которых имеются в архиве. Те файлы, которые не были изменены вирусом (это можно установить с помощью программы-ревизора), удалять не обязательно. Файлы типа .COM и .EXE следует оставлять на диске только в случае крайней необходимости, при условии, что не было сообщения программы-ревизора об их изменении;

в) уничтожить вирус на данной дискете программой-доктором или доктором-детектором. Зараженные файлы, которые программа-доктор не смогла восстановить, необходимо уничтожить;

г) если обрабатываемый диск является системным, то на него следует заново записать загрузочный сектор и файлы операционной системы командой SYS;

д) с помощью архивных копий восстановить файлы, размещавшиеся на дискетах.

Такой обработке следует подвергнуть все диски, которые могли быть заражены или испорчены вирусом.

Контрольные вопросы

1. Что понимается под архивацией файлов?
2. Что такое архивный файл?
3. Назовите основные команды для архивации/разархивации файлов.
4. Назовите основные ключи, используемые при архивации/разархивации файлов.
5. Напишите команду архивации файлов.
6. Напишите команду для разархивации файлов.
7. Что такое "компьютерный вирус"?
8. В чем состоит опасность заражения компьютера вирусом?
9. Перечислите основные меры профилактики для защиты от компьютерного вируса.
10. Каково назначение программ-детекторов?
11. Какие первоочередные меры необходимо принять при подозрении на заражение компьютера вирусом?

на

7. Математические системы

7.1. Решение типовых задач в математических системах

7.1.1. Алфавит. Кодовая таблица

Во всех языках программирования для записи программ и данных используются символы латинского алфавита и специальные знаки. Для кодирования информации на IBM – совместимых ПК используется кодовая таблица ASCII. Она содержит 256 символов, которые пронумерованы от 0 до 255. Первая часть кодовой таблицы с кодами 0-127 – постоянная на всех компьютерах. Она содержит управляющие символы (коды от 0 до 31), часть из которых не отображается на экране, и символы латинского алфавита, цифры. Вторая часть таблицы с кодами от 128 до 255 содержит коды символов национального алфавита и символы псевдографики. Символы латинского и русского алфавита, цифры и спецзнаки вводятся непосредственно с клавиатуры. Для ввода символов псевдографики используется комбинация клавиш Alt - <числовой код> . Числовой код набирается на дополнительном поле клавиатуры.

7.1.2. Команды, операторы, константы, переменные

Команды и ***операторы*** – это синтаксические конструкции языка программирования, им соответствуют определенные подпрограммы обработки информации. Команды оперируют всей программой или ее частью. Операторы служат для реализации алгоритма программы. Команды и операторы – зарезервированные слова языка программирования, они не могут быть использованы в качестве переменных или меток. В командах и операторах импортных программ, которые составляют, к сожалению, подавляющее большинство, используются только символы латинского алфавита.

При обработке информации программы оперируют с данными. В зависимости от способа представления данных они делятся на константы и переменные.

Константы – данные, значения которых не изменяются при выполнении программы. Константы могут быть числовые и символьные. Некоторые языки программирования допускают использование именованных констант.

Переменные – данные, значения которых могут изменяться в процессе выполнения программы. Переменным присваиваются имена (идентификаторы). Имя переменной может содержать один или более символов, в зависимости от используемой программы. Например, в системе программирования Qbasic имя переменной может содержать до 40 символов. В имени переменной можно использовать латинские символы, цифры и некоторые спецсимволы, не допускаются пробелы.

Классификация переменных приведена на рис. 7.1.1.[2]

---

[2] В объектно-ориентированных языках программирования, например C, Delphy, а также Visual Basic число используемых типов переменных значительно больше. Имеются переменные логические, типа дата, время, объектные переменные и др. В настоящем пособии они не рассматриваются.

Переменные могут быть строковые и числовые.

Для представления символов используется один байт информации, поэтому можно закодировать не более 256 символов. Длина строковой переменной может быть не более 255 символов.

Числовые переменные могут быть целые и вещественные. Для хранения информации может использоваться разное число байтов, от этого зависит диапазон изменения переменной. Целые числа могут быть одинарной длины (два байта) и удвоенной длины (4 байта). Значения целых одинарной длины находятся в интервале от –32768 до 32767. Значения целых чисел двойной длины находятся в интервале от – 2147483648 до +2147883647. Вещественные числа одинарной точности занимают в памяти компьютера 4 байта и позволяют представлять числа в интервале от $-1{,}2 \times 10^{-38}$ до $3{,}4 \times 10^{+38}$. На экран монитора выводится 8 знаков, включая десятичную точку.

Вещественные числа удвоенной точности занимают в памяти компьютера

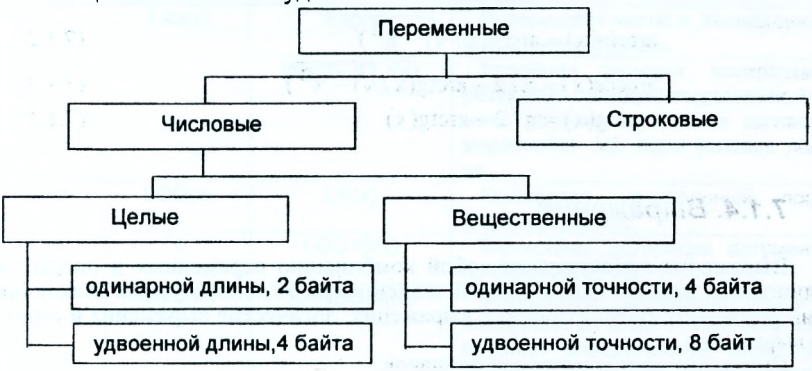


Рис. 7.1.1. Классификация переменных

8 байтов и позволяют отображать числа в интервале от $4{,}19 \times 10^{-307}$ до $1{,}67 \times 10^{+308}$. При выводе на экран чисел с двойной точностью они содержат 16 знаков, включая точку. Математические системы позволяют пользователю самостоятельно устанавливать число десятичных знаков, выводимых на экран. В этом случае числа округляются по правилам математики, то есть, если отбрасываемая часть больше 5, то к последнему разряду прибавляется единица. Системы программирования также позволяют управлять числом десятичных знаков, выводимых на экран, но программным путем.

7.1.3. Функции

При решении математических задач и обработке строковых данных часто приходится выполнять одинаковые вычислительные операции. Поэтому в системах программирования и математических системах имеется большое число ***встроенных подпрограмм***, выполняющих эти операции. Эти подпрограммы называют ***функциями***. Кроме того, вычислительные системы предоставляют пользователю возможность создавать свои ***пользовательские функции*** для реализации часто используемых вычислительных операций.

Синтаксис функции:

Имя_функции(аргумент [, аргумент])

В функциях аргументы заключаются в круглые скобки.

В табл. 7.1.1. приведены основные математические функции языка программирования QBasic и математических систем Derive и Mercury.

В языке программирования QBasic нет логарифма по основанию 10, а также нет обратных тригонометрических функций. Поэтому полезно запомнить формулы, по которым можно вычислять логарифм по произвольному основанию и обратные тригонометрические функции:

$$\mathrm{Log_a}(x) = \frac{\mathrm{Log_b}(x)}{\mathrm{Log_b}(a)} \qquad (7.1.1)$$

$$\mathrm{arcsin}(x) = \mathrm{arctg}(x / \sqrt{1 - x^2}) \qquad (7.1.2)$$

$$\mathrm{arccos}(x) = \pi / 2 - \mathrm{arctg}(x / \sqrt{1 - x^2}) \qquad (7.1.3)$$

$$\mathrm{arcctg}(x) = \pi / 2 - \mathrm{arctg}(x) \qquad (7.1.4)$$

7.1.4. Выражения

Выражения представляют собой комбинацию переменных и операндов, соединенных знаками отношения. В зависимости от используемых знаков операций различают арифметические выражения, логические выражения и строковые выражения.

Синтаксис всех выражений одинаков:

<операнд> знак_операции <операнд>

В качестве операндов в выражениях могут использоваться константы, переменные, функции и другие выражения.

Арифметические выражения

Знаки арифметических операций приведены в таблице 7.1.2.

Пример 7.1. Вычислить значение y= f(x,a,b,c)

$$y = \frac{e^{(-5x^2 + a)} + Sin^3(x^2)}{\arcsin^3(2x^3 + b) * \sqrt{2x^3}} - \frac{ctg^3 x^5 + \lg(8x^3 + c)}{\ln(2x^3 + c) + \sqrt[3]{3x^3 + 8x - 1}}$$

Таблица 7.1.1

Математические функции

Функция			*Описание*
Математика	*QBasic*	*Математические системы*	
Арифметические функции			
| x |	ABS(x)	ABS(x)	Вычисляет абсолютное значение числа x
] x [	CEIL(x)	CEIL(x)	Возвращает целое число, большее или равное *x* для положительных чисел и меньшее или равное *x* для отрицательных чисел
[x]	INT(x)	INT(x)	Возвращает целое число, меньшее или равное *x* как для положительных, так и для отрицательных чисел
e^x	Exp(x)	Exp(x)	Возвращает число e, возведенное в указанную степень
		iif(усл.,V1,V2)	Условная функция, возвращает результат согласно выражению V1, если условие истинно и согласно выражения V2, если условие ложно
Ln x	LOG(x)	LN(x)	Вычисляет натуральный логарифм аргумента
Lg x	-	LOG10(x)	Вычисляет логарифм аргумента по основанию 10
	RND(N)	RANDOM(N)	Возвращает случайное число в интервале от 0 до 1. При N<0 возвращает определенное число, зависящее от N; при N=0 – псевдослучайное число; при N>0 – новое случайное число
Округление чисел	ROUND(x,n)		Возвращает число, округленное к заданному числу десятичных знаков *n*
Знак числа	SGN(x)	-	Возвращает знак числа: 1 , если x>0; 0 , если x=0; -1, если x<0
$\sqrt{x}$	SQR(x)	SQRT(x)	Вычисляет корень квадратный из аргумента
Тригонометрические и обратные тригонометрические функции			
Cos x	COS(x)	COS(x)	Вычисляет косинус угла
Sin x	SIN(x)	SIN(x)	Вычисляет синус угла
Tg x	TAN(x)	TAN(x)	Вычисляет тангенс угла
Ctg x	-	COT(x)	Вычисляет котангенс от *x*
arcsin x	-	ASIN(x)	Вычисляет арксинус числа *x*.[3 4]
arcos x	-	ACOS(x)	Вычисляет арккосинус числа *x*
arctg x	ATN(x)	ATAN(x)	Вычисляет арктангенс числа *x*
arctg x	-	ACOT(x)	
Число π	-	PI	Константа

Таблица 7.1.2

Знаки арифметических операций

Знак операции	*Примеры*	*Описание*
-	-a	Изменение знака числа
^	y=x^a	Возведение числа в степень
*, /	y=a*b/c	Умножение и деление чисел
Mod	y = a Mod b	Возвращает остаток при целом делении двух чисел
+, -	y=a+b-c	Сложение и вычитание чисел
Порядок выполнения операций может быть изменен путем использования скобок. Выражения в скобках выполняются в первую очередь		
( )	y=(a+ b)/c	Скобки

Порядок решения

Запишем выражение с использованием функций языка Qbasic. Для решения задачи разобьем ее на части: вычислим отдельно числители и знаменатели. Кроме того, аргументы у арксинуса и десятичного логарифма сложные, поэтому их можно заменить вспомогательными переменными.

Введем вспомогательные переменные:

$u1 = 2x^3 + b$

$u2 = 8x^3 + c$

Вычислим числители и знаменатели:

Y1 = exp(-5*x^2 +a) + Sin(x^2)^3

Y2 = ATN(u1/SQR(1-u1^2))^3 * SQR(2*x^3)

Y3 = (1/(TAN(x^5))^3 + LOG(u2)/LOG(10)

Y4 = LOG(2*x^3+c) + (3*x^3 + 8*x – 1)^(1/3)

Сформируем окончательный результат:

Y= Y1/Y2 – Y3/Y4

Применение метода подстановки при вычислении сложных выражений позволяет уменьшить или вообще избежать ошибок при написании программ, сводя их к простым выражениям. Кроме того, при этом уменьшается длина записи, упрощается контроль правильности написания программы и поиск ошибок.

При выполнении арифметических операций в системах программирования необходимо помнить о тех ограничениях, которые накладываются на значения аргументов:

- при делении знаменатель не должен быть равен нулю;
- при вычислении логарифмов аргумент должен быть больше нуля;
- при извлечении квадратного корня подкоренное выражение не должно быть отрицательным (при работе с действительными числами);
- в системе программирования QBasic нельзя возводить в дробную степень отрицательные числа. Это связано с особенностью выполнения данной операции в языке программирования QBasic - операция возведения в степень заменяется вычислением по формуле: $a^x = \exp(x * \mathrm{Log}(a))$.

Необходимо помнить также, что не существует корня степени больше двух, поэтому выражение $\sqrt[n]{x}$ при программировании следует заменять степенью числа x: а именно $\sqrt[n]{x} = x^{(1/n)}$ при n>2.

Типичные ошибки при записи арифметических выражений:

Выражение	*Правильно*	*Неправильно*
e^x	Exp(x)	Exp^(x)
$Sin^3(x^2)$	(Sin(x^2))^3	Sin^3(x^2)
	или	
	Sin(x^2)^3	
$\sqrt[3]{x}$	x^(1/3)	x^1/3
$\frac{a+b}{u}$	(a+b)/u	a+b/u
$\frac{a+b}{uq}$	(a+b)/(u*q)	(a+b)/u*q

Логические выражения

Знаки логических операций приведены в табл. 7.1.3.

Наиболее часто используются логические операции И – логическое умножение, ИЛИ – логическое сложение, НЕТ – логическое отрицание.

Логические выражения используются при проверке каких-либо условий. Например, в условиях примера 7.1 при вычислении значения *y* необходимо контролировать значение выражений под знаком натурального и десятичного логарифмов, значение выражения под корнем и значение аргумента арксинуса.

Для контроля этих значений можно записать четыре условия:

Если u1<-1 AND u1>1, тогда напечатать "функция не имеет решения";

Если u2<=0, тогда напечатать "функция не имеет решения";

Если x<0, тогда напечатать "функция не имеет решения";

Если 2*x^3+c<=0, тогда напечатать "функция не имеет решения".

Так как нарушение любого из условий приводит к одному и тому же результату – отсутствию решения задачи, то все три проверки можно объединить в одну, используя оператор OR:

Если x<0 OR u2<=0 OR 2*x^3+c<=0 OR (u1<-1 AND u1>1), тогда напечатать "функция не имеет решения".

При записи логических выражений могут использоваться также логические операторы (табл. 7.1.4).

Строковые выражения

В строковых выражениях могут использоваться только операции сложения (*конкатенации*). В качестве знаков операций могут использоваться символы +, - и знак & - *амперсенд*. Знаки "+" и "&" равносильны. В некоторых языках программирования для операций с строковыми переменными используется знак "-". В этом случае у правого операнда отбрасываются лидирующие пробе-

лы. В качестве операндов в строковых выражениях могут быть только строки символов. Результатом операции также является строка символов.

Таблица 7.1.3

Знаки логических операций

Знак операции	*Примеры*	*Описание*
=	a = b	равно
<	a < b	меньше
>	b > c	больше
<=	a <= d	меньше или равно
>=	c >= e	больше или равно
<>	f <> g	не равно

Таблица 7.1.4.

Логические операторы

Оператор	*Назначение*	*Пример*			
		Выражение	*A*	*B*	*Результат*
And	Логическое умножение (И)	A AND B	1	1	1
			1	0	0
			0	1	0
			0	0	0
Eqv	Проверка логической эквивалентности (=)	A EQV B	1	1	1
			1	0	0
			0	1	0
			0	0	1
Imp	Импликация	A IMP B	1	1	1
			1	0	0
			0	1	1
			0	0	1
Not	Операция логического отрицания (Нет)	NOT A	1		0
			0		1
Or	Операция логического сложения (ИЛИ)	A OR B	1	1	1
			1	0	1
			0	1	1
			0	0	0
Xor	Операция исключающего ИЛИ	A XOR B	1	1	1
			1	0	0
			0	1	0
			0	0	1

В операциях сравнения в качестве операндов могут использоваться строки символов. В этом случае сравнение выполняется посимвольно. Если все пары символов совпадают, то строки считаются равными. Если встречается пара неравных символов, то большей считается строка, содержащая символ с большим кодом в таблице кодов ПК. Если сравниваются строки разной длины, то более короткая строка считается меньшей.

7. 2. Математическая система Mercury

7.2.1. Рабочее окно

Математическая система Mercury является многооконной текстовой программой предназначенной для решения математических задач и представления полученных результатов в виде отчетов и графиков.

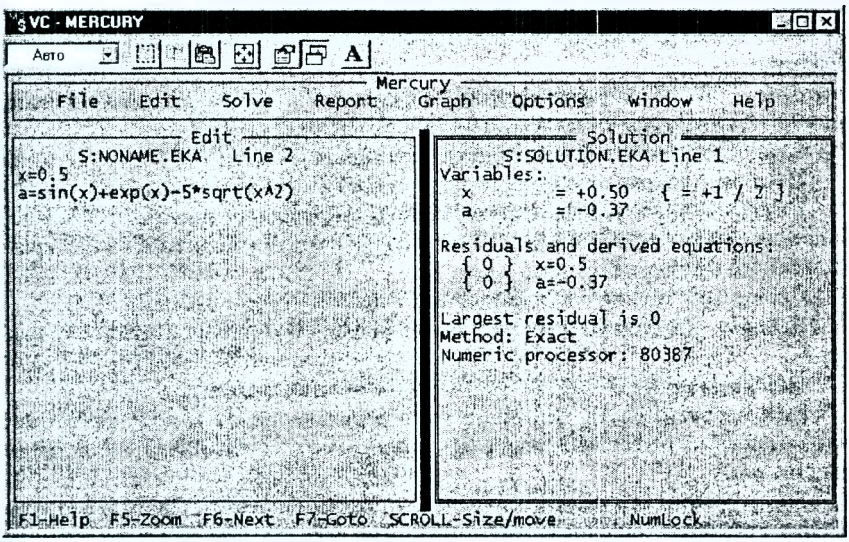


Рис. 7.2.1. Рабочее окно математической системы Mercury

Mercury позволяет вычислять математические выражения, решать алгебраические и трансцендентные уравнения, системы линейных уравнений, вычислять пределы функций, производные и интегралы, выполнять вычисления с мнимыми числами, а также строить графики функций одной переменной. Программа имеет большое число встроенных функций.

Программа имеет хорошую систему помощи и большой набор примеров решения типовых задач. Файлы примеров размещены на диске S: в подкаталогах EKA и EKAR каталога Mercury.

Программа Mercury работает под управлением операционной системы MS-DOS. Запуск программы осуществляется командой S:\Mercury\mercury.exe или с помощью меню пользователя.

После запуска программы на экране появляется рабочее окно (рис.7.2.1)

В верхней части экрана расположено главное меню программы. Ниже него расположены окно редактирования *Edit* и окно результатов решения задачи *Solution*. В нижней части рабочего окна расположена строка состояния. В эту

строку выводится назначение функциональных клавиш, доступных в текущем состоянии окна, а также состояние клавиши NumLock.

7.2.2. Меню

Меню турбооболочки является удобным средством использования команд управления программой.

File - служит для открытия файлов, сохранения их на диске, выхода из программы. Он содержит следующие опции:

Load – загрузка файла с диска в окно редактирования, имя файла запрашивается;

New – открытие нового файла. По умолчанию вновь созданному файлу присваивается имя NONAME.EKA;

Save – сохранение файла на диске со старым именем;

Write to – сохранение файла на диске. Имя файла запрашивается;

Directory – просмотр каталогов дисков, запрашивается маска для поиска. Например: S:\Mercury\EKA – будет выведено оглавление подкаталога EKA расположенного в каталоге Mercury диска S;

Change dir – смена текущего каталога диска;

Rename – переименование рабочего файла в оперативной памяти компьютера;

Quit – выход из среды Mercury в операционную систему.

Edit – обеспечивает переход в окно редактирования текста программы. Выход из режима редактирования в меню программы осуществляется нажатием клавиши *Esc*.

Solve – переход в режим вычисления задач, записанных в окне редактирования. Результат решения появляется в окне *Solution*. Для перехода в режим редактирования можно использовать также комбинацию клавиш *Alt-S*.

Report – формирование отчета. Данный пункт меню содержит ряд опций:

Go – сформировать отчет и поместить его в место, указанное в опции Output;

Output – позволяет указать направление вывода отчета: в файл или на принтер;

Name – запрашивает имя файла для сохранения отчета. По умолчанию предлагает имя рабочего файла (NONAME.RPT);

Character set – обеспечивает выбор кодовой таблицы (ASCII или IBM);

Keep log – указать содержание файла (Нет/Да);

Log file name – запрашивает имя файла для сохранения содержания, по умолчанию предлагает сохранить в файле Mercury.log на текущем диске.

Graph – построение графика. Данный пункт меню содержит большое число опций:

View – просмотр графика;

Print – печать графика;

Write to – сохранить описание графика на диске;

Bounds – запрашивает левую и правую границы интервала, на котором будет строится график функции;

Legend – легенда, обозначение графиков. Применяется в том случае, когда на график выводится одновременно несколько функций;

Mode – выбор типа монитора. По умолчанию выбирается автоматически.

GPoints – устанавливает число точек на 1см. От этого параметра зависит качество графика.

Style – стиль линии: толстая (Thick) или тонкая (Thin);

Title – заголовки графика. Позволяет ввести заголовок, подзаголовок, название осей X и Y.

Options – открывает доступ к настройкам параметров среды:

Solver – позволяет настроить параметры решения задач такие, например, как число десятичных знаков, точность, число итераций, время решения задачи и др.;

Colors – настройка цвета рабочего окна, меню, системных окон;

Directories – устанавливает рабочие маршруты для системных файлов и файлов пользователя;

Printer – выбор типа принтера и порта подключения принтера;

Load setup – загрузка сохраненных настроек рабочей среды;

Write setup – сохранение установок рабочей среды.

Window – данный пункт меню предназначен для управления окнами:

Open – открыть окно;

Close – закрыть окно;

Next – перейти к следующему окну;

Zoom – распахнуть окно;

Tile – вывести окна мозаикой, то есть расположить их один возле другого;

Stack – вывести окна каскадом. В этом случае текущее окно выводится полностью, а нетекущие окна лишь немного выступают из-за текущего окна;

Goto – перейти к указанному окну.

Help – вызов подсказки. Подсказка выводится на английском языке.

7.2.3. Решение типовых математических задач

Запись математических выражений производится в окне редактирования обычным способом, например (рис. 7.2.1):

```
a= Sin(x)+ exp(x) –5*Sqrt(x^2)
```

В одной строке разрешается размещать несколько операторов, разделяя их двоеточием. Для ввода комментариев к программе используется символ ";". Для перехода на новую строку установите курсор в конец строки и нажмите клавишу Enter. Данный режим работает только в режиме вставки символов. Переход в режим вставки или замещения символов осуществляется нажатием клавиши *Insert (Ins)*.

Математическая система Mercury имеет большое число встроенных функций для решения типовых задач: арифметических, тригонометрических, обратных тригонометрических, вычисления производной, определенных интегралов, пределов, функций комплексного переменного. Для ознакомления с функциями программы вызовите подсказку следующим образом: выделите в меню команду *Help*, нажмите клавишу *Enter*, выберите в открывшемся меню команду *Functions*. Далее выберите интересующую вас группу функций. Для просмотра списка всех функций выберите опцию *Summary*.

Имеется возможность создать функцию пользователя с помощью операции присваивания:

Sec(x):=1/cos(x)

Примеры решения задач в математической системе Mercury

1. Вычисление выражений

Вычислить выражение $y=(\arcsin x + a\,Ctg^2 x + \ln^3 x^2) / \lg(x)$ при $x=0{,}123$; $a=1{,}546*10^{-5}$

Решение:

Введите в окне редактирования текст программы

x= 0.123 : a=1.546E-5

y=(ASIN(x) + a* (1/TAN(x))^2 + LN(x^2)^3)/LOG10(x)

Для вычисления нажмите комбинацию клавиш *Alt-S*.

Ответ y= 14.339

2. Решение системы линейных уравнений:

Решить систему линейных алгебраических уравнений:

$$\begin{cases} 5x + 2y + z = 7 \\ 3x - 5y - 7z = -1 \\ x + y + z = 5 \end{cases}$$

Решение:

Запишите систему уравнений по правилам языка программирования и введите команду *Alt-S*.

Ответ: x=-0.26316, y=3.0526, z=2.2105

3. Вычисление производной от функции одной переменной

Для вычисления производной используется функция DERIV. Формат функции: Deriv(F(x), x)

Здесь F(x) – произвольная функция аргумента x. Функция может быть записана либо непосредственно в операторе, либо описана предварительно как функция пользователя. X – значение аргумента, при котором вычисляется значение производной. Если значение аргумента неизвестно, то результат выдается в общем виде.

Найти производную от функции $y=e^{x*Sin(x)}$ в символьном виде.

Решение:

F(x):=Exp(x*Sin(x)) ; объявление функции пользователя

```
A=deriv(f(x),x)        ; вычисление производной
```

Ответ: (SIN(x)+x*COS(x))*EXP(x*SIN(x))

Вычислить производную от функции $z=e^{y*Sin(y)}$ при y=1.

Решение:

```
F(x):=Exp(x*Sin(x))
B=deriv(f(y),y)
Y=1
```

Ответ: b=3.2054

4. Вычисление интеграла

Для вычисления интеграла используется функция INTEG. Формат функции: INTEG(f(x),x,a,b)

Здесь f(x) – функция произвольного аргумента, x – аргумент, a и b – верхний и нижний пределы интегрирования, соответственно.

Вычислить определенный интеграл $s = \int_0^{\pi} x^2 Sin(x)dx$

Решение:

```
F(x):=x^2*Sin(x)
A=INTEG(f(x),x,0,PI)
```

Ответ: a=5.8696

5. Вычисление пределов

Для вычисления пределов используется функция LIMIT. Формат функции: LIMIT(f(x),x,a,b)

Где f(x) - функция, x – аргумент, параметр a определяет, как приближается аргумент к пределу: слева a = -1 , справа a = 1; b=INF – бесконечность.

Вычислить предел функции y=x Sin(1/x) при x, стремящимся к бесконечности справа.

Решение:

```
A=LIMIT(x*Sin(1/x),x,1,Inf)
```

Ответ: A=1

6. Построение графика функции.

Построить график трехфазного тока на отрезке от –PI до +PI.

Решение:

```
y(x):=Sin(x)
u(x):=Sin(x + 2*PI/3)
g(x):=sin(x + 4*PI/3)
PLOT  y, u, g
```

Выполните команду ***Solve***.

Перейдите в меню ***Graph***, установите интервал построения графика функции, введите заголовки (заголовки вводятся только латинскими символами, описание осей можно выполнять русскими символами) и введите команду View для просмотра графиков.

Описание графика можно выполнить непосредственно в программе, например:

Gbounds –6.28, 6.28 ; устанавливается интервал построения графика;
Title "Sine and Cosine" ; главный заголовок
Subtitle "" ; подзаголовок
xLabel "радианы" ; название оси X
yLabel " значение" ; название оси Y

7. Решение алгебраических и трансцендентных уравнений

Алгебраические уравнения можно решить графически или численными методами. Графически можно решить уравнение в том случае, если его можно разложить на две независимых функции. Например, уравнение $\mathrm{Sin}(x)+0{,}01x^2=0$ можно представить в виде двух функций: $y=\mathrm{Sin}(x)$ и $y=0{,}01x^2$
Точки пересечения графиков и дадут значение корней уравнения.

Программа Mercury успешно решает такие задачи.

Решение:

Sin(x) + 0.01*x^2 = 0

Ответ: уравнение имеет пять корней -8.59; -6.76; 0; 3.25; 5.92

8. Вычисление суммы конечных и бесконечных рядов.

Вычисление сумм рядов с заданным числом повторений осуществляется с помощью функции SUM. Формат функции: Sum(f(x),x,a,n)

Здесь f(x) – функция, x – аргумент, a – начальное значение аргумента, n – число итераций.

а) вычислить сумму конечного ряда $\sum_{i=1}^{n} 1/i^2$ при i, изменяющемся от 1 до 1000.

Решение:

s= SUM(1/(n*n),n,1,1000)

Ответ: 1.643934567

б) вычислить сумму бесконечного ряда $\sum_{i=1}^{\infty} 1/i^2$

Решение:

s= SUM(1/(n*n),n,1,INF)

Ответ: 1.644934067

9. Отыскание комплексных корней уравнений

Вычислить корни уравнения $y=x^3 - x^2 - x - 1$ при $x=-0.5 - 0.5i$

Решение:

x^3 - x^2 – x – 1 = 0
x:= -0.5 - 0.5*i

Ответ:

0.33333; 0.001i

Другие примеры можно загрузить с диска S:\Mercury\Eka*.eka

Подготовка отчетов.

После выполнения расчетов сохраните отчет в файле на диске. Войдите в пункт меню *Report*. С помощью опции *Name* задайте имя файла. Расширение имени

файла по умолчанию предлагается *RPT*. С помощью опции *Output* установите режим сохранения отчета в файле (File), а затем введите команду *GO*. Отчет сохраняется в текстовой форме. Его теперь можно просмотреть средствами операционной системы MS-DOS или другим редактором, например Microsoft Word. Подготовленный отчет из редактора Microsoft Word можно вывести на печать. [5]

7.3. Математическая система Derive

7.3.1. Назначение и общая характеристика программы

Программа Derive является многооконной текстовой программой предназначенной для решения различных математических задач: вычисления функций, решения систем линейных уравнений, решения алгебраических и трансцендентных уравнений, численного и аналитического дифференцирования и интегрирования. Программа позволяет строить графики функций двух и трех переменных. Отличительной особенностью программы является возможность осуществлять преобразование функций (приведение подобных членов, разложение на сомножители) и решать задачи в аналитическом виде. Программа располагает стандартным сервисом: возможностью записывать текст задачи на языке близком к общепринятому математическому языку, редактировать его, сохранять на диске в форматах соответствующих таким языкам программирования, как Basic, Pascal, C, считывать текст программы, а также формировать текст задачи на основе нескольких файлов.

Загрузка программы осуществляется командой S:\Derive\derive.exe или через меню пользователя.

После запуска программы на экране появляется рабочее окно программы (рис.7.3.1). В верхней части окна могут размещаться окна алгебры и графики.

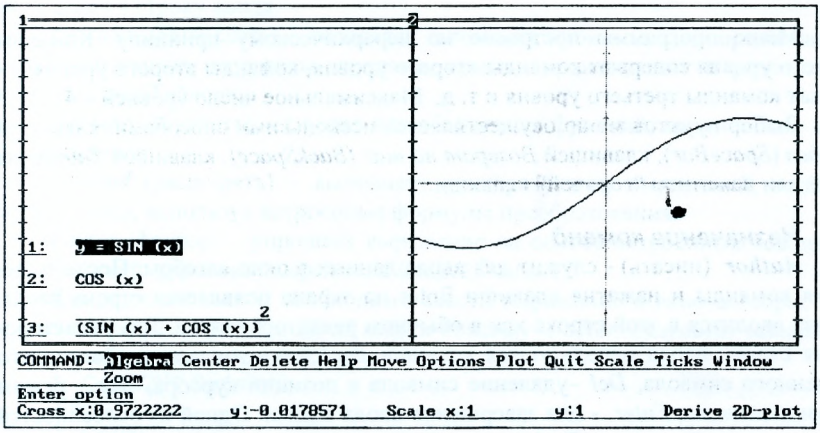


Рис.7.3.1. Рабочее окно программы Derive

При загрузке программы на экране присутствует только окно алгебры. В нижней части экрана выводится панель диалога.

Окно алгебры служит для ввода задач. Выражения в окне алгебры могут записываться как с левой частью, так и без левой части. Все задачи автоматически нумеруются. Переход от одной задачи к другой осуществляется с помощью клавиш управления перемещением курсора вверх ↑, вниз ↓. Стрелки влево ←, вправо →, совместно со стрелками вверх, вниз, служат для выделения части выражения. Новые задачи вводятся в окно алгебры с помощью команд *Author, Build, Calculus*. Кроме того задачи формируются при выполнении команд преобразования и вычисления. Поэтому число задач в окне алгебры быстро увеличивается.

Окно графики служит для вывода графиков функций одной и двух переменных.

Панель диалога содержит четыре строки (рис. 7.3.1, 7.3.2). Первая и вторая строки служат для вывода команд главного меню. Вторая строка служит также для вывода запросов программы. Третья строка предназначена для вывода сообщений программы, а четвертая строка – строка состояния. Содержание этой строки зависит от режима работы программы.

```
COMMAND: Author Build Calculus Declare Expand Factor Help Jump soLve Manage
         Options Plot Quit Remove Simplify Transfer moVe Window approX
Enter option
                                    Free:100%                Derive Algebra
```

Рис. 7.3.2. Главное меню программы Derive

7.3.2. Команды главного меню

Меню программы построено по иерархическому принципу. Команды первого уровня содержат команды второго уровня, команды второго уровня содержат команды третьего уровня и т. д. Максимальное число уровней – 4.

Выбор пунктов меню осуществляется несколькими способами: клавишей *Пробел* (*SpaceBar*), клавишей *Возврат на шаг (BackSpace)*, клавишей *Табуляция (Tab)* или нажатием "горячей" клавиши.

Назначение команд

Author (писать) - служит для ввода данных в окно алгебры. После выделения команды и нажатия клавиши Enter на экране появляется строка ввода. Задача вводится в этой строке как в обычном редакторе текста. Для редактирования строки ввода используются клавиши *BackSpace* – удаление последнего введенного символа, *Del* –удаление символа в позиции курсора, *Ins* – режим вставки символа, *Enter* - для завершения ввода задачи. Одной из особенностей математической системы Derive является то, что клавиши управления переме-

щением курсора не применяются для перемещения по строке ввода. Для этой цели используются комбинации клавиш: *Ctrl-A* – для перемещения на один символ влево, *Ctrl-S* – для перемещения на одно слово влево, *Ctrl-D* – для перемещения на один символ вправо, *Ctrl-F* – для перемещения на одно слово вправо. Для быстрого перемещения в начало или в конец строки ввода используются комбинации клавиш *Ctrl-Q-S* и *Ctrl-Q-D*, соответственно (нажать, например, клавишу *Ctrl* и, не отпуская ее, нажать последовательно клавиши *Q* и *S*).

Для ускорения ввода задач в строку ввода можно использовать ссылки на номер задачи, например, #1+#2*#4, а также клавиши *F3* и *F4*. Клавиша *F3* помещает выделенную задачу в строку ввода, а клавиша *F4* помещает выделенную задачу в строку ввода и заключает ее в скобки.

При правильном вводе функций их имена переводятся автоматически в верхний регистр. При неправильной записи имени функции программа разделяет слово на символы. Этот признак может использоваться для контроля правильности ввода задач.

Build (строить) - позволяет формировать новые задачи из задач, имеющихся в окне алгебры. После выбора команды и нажатия клавиши Enter появляется строка ввода, в которой необходимо указать номер первой задачи (операнда) и нажать клавишу *Enter*, затем выбрать из открывшегося списка операцию, нажать клавишу *Enter* и ввести в строке ввода номер второй задачи (операнда) и так далее. Для завершения формирования задачи выберите в строке операндов команду *Done*.

Строка операндов содержит следующие опции: +, -, *, /, ^, `, . , =, Minus, Recip, Ln, Exp, Tan, Sin, Cos, Atan, !, %, Done. Некоторые из этих опций нуждаются в пояснении. Символ "`" – транспонирование матриц, например A`; *Minus* – умножение на минус единицу; *Recip* – обратная величина, например, Recip A будет равно 1/A; *!* – вычисление факториала, *%* - процент, умножает число на 100.

Как показывает практика, команда *Build* не совсем удобна для работы, удобнее пользоваться командой Author с использованием приемов ускоренного ввода задач в строку ввода.

Calculus (вычислять) – позволяет вычислять производные, определенные и неопределенные интегралы, пределы, суммы, произведения и разложение функции в ряд Тейлора.

Declare (объявить) – служит для объявления функций, переменных, матриц и векторов.

Expand (расширять) – выполняет преобразование выражений, пытаясь растянуть его, используя встроенные формулы преобразования.

Factor (фактор) – упрощает выражение на основе встроенных тождеств.

Напрмер: упростить выражение $Cos(x/2)^2-Sin(x/2)^2$

Для решения задачи необходимо ввести выражение в окно алгебры, оно запишется под номером 1 и затем ввести команду *Factor*. Результат запишется в окне алгебры под номером 2.

1. Cos(x/2)^2-Sin(x/2)^2
   Factor
2. Cos(x)

При выполнении команды последовательно запрашиваются имена переменных, по которым приводятся подобные и тип преобразования (*Triviale* – простейшие, *Squarefree* - квадратичное, *Rational* - рациональное, *raDicale* коренное, *Complex* с использованием полного набора выражений).

Help (помощь) – получение помощи. При выборе данного пункта открывается дополнительное меню, которое позволяет выбрать тему. Для просмотра помощи используются команды *Next* – переход к следующему окну помощи, *Previous* – переход к предыдущему окну помощи, *Resume* – выход в главное меню помощи.

Jump (прыжок) – переход к указанной задаче. Номер задачи запрашивается.

Solve (решать) – решение уравнения. Запрашивается номер задачи и имя переменной. Решение записывается в аналитической форме в новую строку.

Manage (управлять) – установка режимов.

Options (параметры). Открывает доступ к командам выбора типа адаптера монитора (*Display*), цвета окна, панели диалога и т.д. (*Color*), режима количества значащих цифр в числе (*Digits*), ввода имен переменных (*Input*) и др. На команде *Input* остановимся несколько подробнее. Эта команда позволяет установить режим ввода имен переменных. По умолчанию установлен режим односимвольных имен переменных (Character), например, X, Y. При попытке ввести имена X1, X2 программа разделяет переменные, вставляя между символами пробел (соответствует по умолчанию умножению переменных). Для использования длинных имен переменных необходимо установить режим Word.

Plot (чертить) – переход в графическое окно. При первоначальном вводе данной команды на экран выводится подменю, позволяющее выбрать способ деления рабочей части экрана на окно алгебры и окно графики: Beside (рядом) – окно делится на две части по вертикали; Under (ниже) – окно делится на две части по горизонтали; Overlay (сверху) – графическое окно занимает весь экран. При выборе режима деления окна по вертикали или по горизонтали дополнительно запрашивается размер выделяемой части экрана в строках или столбцах. Как известно, в текстовом режиме экран имеет 25 строк и 80 столбцов. Поэтому число 10 при делении окна по горизонтали будет означать, что под окно графики выделяется 10 строк, а при делении окна по вертикали число 40 также будет означать, что под окно графики выделяется половина экрана.

Quit (выход) - завершение работы с программой.

Remove (удаление) – удаление задач из окна алгебры. Номер начальной и конечной задач запрашивается.

Simplify (упрощать) – упростить выражение или найти его точное значение.

Transfer (передача) – обеспечивает связь с внешними устройствами: сохранение файла программы на диске (*Save*), загрузку файла с диска (*Load*), вывод файла на печать (*Print*), очистку окна алгебры (*Clear*) – удаляются задачи из окна алгебры и все переменные из памяти, присоединить выражение из файла к задаче в окне алгебры (*Merge*), опция *Demo* обеспечивает запуск демонстраци-

онного файла с расширением *.dmo*. Расширение имени файла задач по умолчанию принимается *.mth*.

Move (двигать) – позволяет перемещать задачи в окне алгебры.

Window (окно) - управление окнами: открытие, закрытие, переход из одного окна в другое, деление окна.

approX (приближать) – вычислить приближенное значение выражения.

7.3.3. Алгоритмы решения типовых задач

1. Вычисление выражения

Задача. Вычислить значение выражения

$$y = \frac{\mathrm{Log}_3(x^2 + a) + e^{x+1.5} + \sin^3(x^2)}{\mathrm{ctg}(x) + \sqrt[3]{2x^2 + \arccos^2(x)}} \qquad (7.3.1)$$

при x=0,653, a=1, $754*10^{-4}$

Алгоритм решения.

- Выбрать пункт меню *Author* и ввести числитель выражения (7.3.1);
- Выбрать пункт меню *Author* и ввести знаменатель выражения (7.3.1);
- Присвоить переменной *a* заданное значение: a:=1.754
- Присвоить значение переменной *x*: x:=0.653
- Сформировать с помощью команды *Author* или *Build* выражение Y=#1/#2;
- Вычислить выражение: выделить задачу и ввести команду *approX*.

Пример решения задачи приведен на рис. 7.3.3.

Из рисунка видно, что задача записывается в окне алгебры в виде, близком к привычной нам записи в тетради. Показатели степени выводятся над чертой.

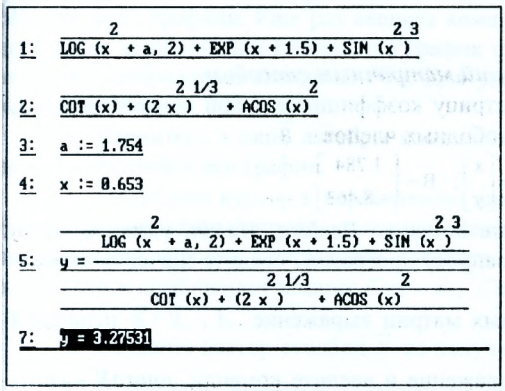


Рис.7.3.3. Вычисление выражения

2. Вычисление производной

Задача. Найти производную от выражения $\mathrm{Cos}(x)e^x$ при *x* равном 0,5.

Алгоритм решения:

- Введите задачу;
- Введите команду *Calculus, Differentiate;*
- На запрос программы укажите номер задачи #1, параметр – *x*, порядок производной – 1;
- Для решения задачи в аналитическом виде введите команду *approX*;

• Для получения численного значения производной присвойте *x* значение и введите команду *approX*.

Решение приведено на рис.7.3.4.

На рис. 7.3.4. в задаче 5 вместо основания натурального логарифма *e* выводится символ И. Это несоответствие обозначений связано с используемой кодовой таблицей.

```
1:  COS (x) EXP (x)

        d
2:  ------ (COS (x) EXP (x))
        dx

        x
5:  И  (COS (x) - SIN (x))

6:  x := 0.5
```

Рис. 7.3.4. Вычисление производной

Аналогично вычисляется определенный и неопределенный интеграл, предел функции. При вычислении предела функции для обозначения бесконечности используется функция INF. Вид выражения предел функции y=Sin(x)/x при x стремящемся к бесконечности приведен на рис. 7.3.5.

```
          SIN (x)
lim  ------------
x→ь         x
```

Рис. 7.3.5. Предел функции

3. Полное исследование системы линейных уравнений

Задача. Решить систему уравнений с двумя неизвестными матричным способом, с помощью обратной матрицы, методом Крамера, графическим способом. По результатам решения системы уравнений графическим способом вычислить вектор невязки и норму вектора невязки. Найти вектор собственных значений и обусловленность матрицы.

Алгоритм решения.

Пусть дана система уравнений

$$\begin{cases} 5x + 3y = 1{,}784 \\ -2x + 5y = -8{,}463 \end{cases} \qquad (7.3.2)$$

1. Решение системы уравнений матричным способом

• Введите три матрицы: матрицу коэффициентов при неизвестных - *A*, вектор неизвестных – $\overline{X}$ и вектор свободных членов – $\overline{B}$

$$A = \begin{bmatrix} 5 & 3 \\ -2 & 5 \end{bmatrix}; \quad \overline{X} = \begin{bmatrix} x \\ y \end{bmatrix}; \quad \overline{B} = \begin{bmatrix} 1.784 \\ -8.463 \end{bmatrix}$$

Для ввода матрицы выберите команды ***Declare, Matrix***, укажите число строк и число столбцов. Затем по запросу программы введите значения элементов матрицы.

• Сформируйте из указанных матриц выражение $A \cdot \overline{X} = \overline{B}$ используя команду ***Author*** или ***Build***.

• Выделите полученное выражение и введите команду ***approX*** - получится система уравнений, подобная выражению (7.3.2).

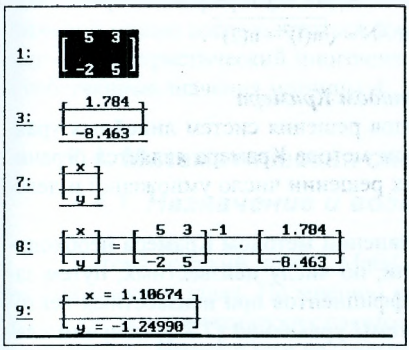


Рис. 7.3.6. Решение системы линейных уравнений с помощью обратной матрицы

• Введите команду ***soLve*** для решения полученной системы уравнений. Если ответ получается в виде рациональной дроби, снова введите команду ***approX***.

2. Решение системы уравнений с *ий с*
помощью обратной матрицы

• Введите матрицу коэффициентов, матрицу неизвестных и матрицу свободных членов.

• Сформируйте выражение

$$\bar{X} = A^{-1} \cdot \bar{B}$$

• Введите команду ***approX*** для решения полученного выражения.

Результат вычисления приведен на рис. 7.3.6.

3. Решение системы линейных уравнений графическим способом

• Введите уравнения согласно выражению 7.3.2.

• Выразите из этих уравнений значение *y* в явном виде. Для этого выделите первое уравнение и введите команду ***soLve***. По запросу программы укажите переменную, относительно которой хотите решать уравнение, - y. Выполните аналогично преобразование для второго уравнения. В результате получите два выражения:

$$y = (-5x + 1{,}784) / 3 \qquad (7.3.3)$$
$$y = (2x - 8{,}463) / 5$$

• Выделите первое уравнение и введите команду Plot. Программа переходит в окно графики. Еще раз введите команду Plot – строится график функции. Если по окончании построения график отсутствует на экране, уменьшите масштаб изображения клавишей *F10*. Для увеличения масштаба используется клавиша *F9*.

• Вернитесь в окно алгебры командой ***Algebra***. Выделите второе уравнение и постройте его график.

• Подведите курсор к пересечению линий и определите значение *x* и *y* (значения выводятся в строке состояния). Запишите эти значения. Для быстрого перемещения курсора используйте клавиши *PgUp*, *PgDn* – перемещение по вертикали и комбинации клавиш *Ctrl* - ←, *Ctrl*- → – перемещение по горизонтали.

• Создайте вектор-столбец $\bar{X}$ из полученных значений *x* и *y*.

• Вычислите расчетное значение вектора свободных членов $\bar{B}$, умножив матрицу A на вектор $\bar{X}$.

• Вычислите вектор невязки $\bar{N} = \bar{B} - \bar{B}$.

• Вычислите норму вектора невязки. Норма вектора равна корню квадратному из суммы квадратов его компонент - $N = \sqrt{n(1)^2 + n(2)^2}$.

3. Решение системы уравнений методом Крамера

Одним из простых и понятных методов решения систем линейных уравнений является метод Крамера. Недостатком метода Крамера является ограничение на порядок уравнений, так как при их решении число умножений и делений имеет порядок N –факториал.

Для решения системы линейных уравнений методом Крамера необходимо составить дополнительные определители, по числу неизвестных, путем замены в главном определителе столбца коэффициентов при неизвестных вектором свободных членов, например, для системы уравнений (7.3.2) получим следующие дополнительные определители:

$$\Delta_x = \begin{bmatrix} 1.784 & 3 \\ -8.463 & 5 \end{bmatrix}; \quad \Delta_y = \begin{bmatrix} 5 & 1.784 \\ -2 & -8.463 \end{bmatrix}.$$

Значения неизвестных получим путем деления дополнительного определителя на главный определитель: $x = \Delta_x / \Delta$, $y = \Delta_y / \Delta$. Для вычисления определителя в математической системе Derive используется функция DET. Тогда значения *x* и *y* вычислим с помощью выражений:

$$x = DET(\Delta_x) / DET(\Delta); \; y = DET(\Delta_y) / DET(\Delta).$$

4. Вычисление вектора собственных значений.

Собственным значением матрицы *A* называется такое число λ, для которого выполняется условие $A\bar{e} = \lambda\bar{e}$, где $\bar{e}$ - ненулевой вектор. Матрица имеет столько собственных значений, какова ее размерность.

Для отыскания всех собственных значений матрицы *A* составляется характеристическое уравнение

$$DET(A - \lambda E) = 0,$$

где E – единичная матрица. Для системы уравнений с двумя неизвестными (7.3.2) получим следующее выражение:

$$DET \begin{bmatrix} 5-\lambda & 3 \\ -2 & 5-\lambda \end{bmatrix} = 0. \qquad (7.3.4)$$

Раскрывая определитель в левой части равенства (7.3.4), получим квадратное уравнение

$$\lambda^2 - 10\lambda + 31 = 0,$$

решая которое, получим корни характеристического уравнения $\lambda_1 = 5 - \sqrt{6}M$, $\lambda_2 = 5 + \sqrt{6}M$.

Обусловленность матрицы определяется как отношение максимального значения числа обусловленности матрицы к наименьшему значению числа обусловленности матрицы:

$$cond\ A = |\lambda max / \lambda min|.$$

Для получения характеристического уравнения и вектора собственных значений можно воспользоваться функциями программы Derive: *CHARPOLY(A, mu)* – характеристический многочлен матрицы *A* от *mu* и *EIGENVALUES(A, mu)* – собственные значения матрицы *A*.

7.4. Математическая система MathCAD

7.4.1. Назначение и возможности

Математический пакет MathCAD предназначен для решения физико-математических задач. По мнению специалистов, он является одним из наиболее удобных математических пакетов для несложных расчетов на персональном компьютере. Имеется уже несколько версий данной программы. В данном пособии описан интерфейс MathCAD 2001.

Программа имеет естественный язык представления математических зависимостей и инструменты их выбора (рис. 7.4.1). Имеется возможность вводить размерности переменных и автоматически контролировать соответствие размерностей операндов и результата. MathCAD имеет также встроенный текстовый процессор, который позволяет, например, подготовить статью без помощи других средств редактирования текста. Этот редактор обеспечивает экспорт данных, графиков, используя буфер обмена, а также механизм вставки и внедрения (DDE, OLE). Основным достоинством пакета разработчики считают возможность только собственными средствами сформулировать задачу в привычных обозначениях, исследовать ее, обработать исходные данные, выбрать метод решения, получить результаты, задокументировать их и передать по сети. Вместе с пакетом могут использоваться прикладные дополнения (обработка сигналов и изображений, анализ электрических цепей, численный анализ, "продвинутая" математика и статистика, теория очередей). Имеется ресурсный центр, позволяющий получать необходимую информацию и программное обеспечение через Интернет.

7.4.2. Рабочее окно

После запуска программы на экране появляется рабочее окно MathCAD (рис. 7.4.1.). При загрузке на экране появляются те окна, которые были на экране при выходе из программы.

Рабочее окно MathCAD - типичное окно приложения Windows. Оно содержит строку заголовка с кнопками открытия и закрытия окна, главное меню и панели инструментов. По умолчанию на экран выводятся стандартная панель инструментов, панель инструментов форматирования и панель инструментов математика (на рис.7.4.1. открыты все окна инструментов панели инструментов математика). На чистом рабочем поле располагается *курсор* – красный крестик, который отмечает *точку ввода*.

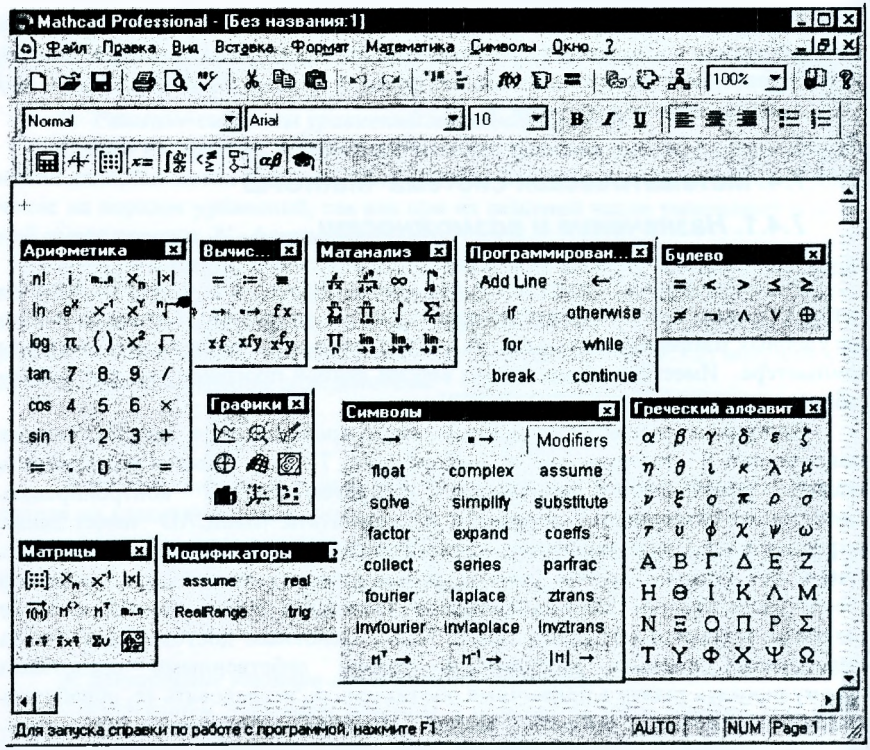


Рис.7.4.1. Рабочее окно Mathcad

Запуск программы можно произвести через главное меню Windows или щелчком мыши по пиктограмме на рабочем столе.

MathCAD имеет мощную систему встроенной помощи, вызываемую кнопкой F1, а также ресурсный центр, который позволяет получить подсказку, рассмотреть пример решения и, при необходимости, скопировать пример на рабочее поле.

7.4.3. Ресурсный центр

Ресурсный центр (рис. 7.4.2.) содержит три раздела:

- *Overview and Tutorials* – оглавление и характеристика пакета. Данный раздел открывает доступ к систематическому оглавлению помощи, описанию порядка работы с трехмерной графикой, анализа данных, решения дифференциальных уравнений, документирования и публикации трудов.
- *QuickSheets and Reference Tables* – быстрый поиск и справочная таблица. Например, для получения подсказки о правилах записи алгебраических выражений необходимо войти в раздел и выбрать опцию Arithmetic and algebra.

- *Extending MathCAD* – расширенный MathCAD.

Пять желтых кнопок – *Collaboratory, Web Library, Mathcad.com, Support* и *Web Store* предназначены для получения информации из сети.

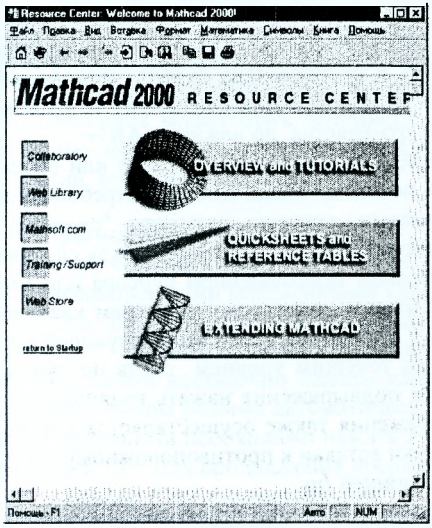


Рис. 7.4.2. Ресурсный центр

7.4.4. Начало работы

Встроенный редактор MathCAD довольно сложный и требует запоминания многих комбинаций клавиш и соответствия нажатия отдельных клавиш получаемым результатам. В этом его существенный недостаток.

Особенности ввода и редактирования выражений:

- данные вводятся в *точке ввода*, отмеченной красным крестиком;
- числа в научной (полулогарифмической) нотации вводятся как произведение мантиссы на 10 в соответствующей степени, например, 0,7364*10+5;
- углы по умолчанию задаются в радианах;
- мнимая единица записывается как i или j сразу после числового множителя, например, 5i или 5j;
- латинские буквы, цифры и знаки операций, круглые скобки набираются непосредственно с клавиатуры. С клавиатуры можно вводить также имена встроенных функций;
- переменные могут иметь имена с индексами (подстрочным текстом), например, x_{min}, переход к набору индексов осуществляется вводом символа "." ;
- латинские символы могут быть преобразованы в греческие с учетом регистра исходного символа нажатием клавиш Ctrl+G сразу после ввода латинского символа. Большинство латинских символов соответствуют греческим символам, исключение составляют пары c - χ, q - Θ , x - ξ , y - ψ
- символы в нижнем и в верхнем регистре для Mathcad разные символы;
- символ ∞ выбирается из панели с операторами анализа или комбинацией клавиш Ctrl+Z, а число π – комбинацией клавиш Ctrl+P;
- знаки операций *, /, ^ вводятся с клавиатуры, но при выводе на экран заменяются точкой, обыкновенной дробью и показателем степени, соответственно;
- открывающая квадратная скобка "[" преобразуется в индекс. Возврат к основному шрифту осуществляется в этом случае нажатием клавиши *Пробел;*
- обратный слэш «\» вызывает шаблон квадратного корня;

- символ апостроф – появление круглых скобок вокруг выделенного выражения;

- вертикальная черта – вызывает шаблон для вычисления абсолютной величины или определителя матрицы.

Для вызова контекстного меню следует нажать Shift+F1. Появившийся вопросительный знак нужно передвинуть в нужную зону экрана и щелкнуть мышью. Нажатие *Esc* отменяет режим. Для пояснения сообщения об ошибке следует нажать F1. Ошибочное действие отменяется по нажатию Alt+← .

При работе с инструментами MathCAD: выделении функции или оператора, - на экране появляется в текущей позиции ввода шаблон затребованной конструкции с черными прямоугольниками, которые называют "*слотами*", вслед за ними следует ввести нужные аргументы. Перемещение между слотами организуется с помощью клавиш управления перемещением курсора или мыши, а между верхними и нижними пределами оператора - нажатием клавиши *Tab*. Начинать набор предпочтительнее с выбора операции, формирующей слоты. В процессе набора нужно следить за текущим уровнем слота по синему (уголковому) курсору и при завершении подвыражения нажать клавишу *Пробел*. Расширение выделенной части выражения также осуществляется с помощью клавиши *Пробел*. Перемещение точки вставки к противоположному концу выделенного выражения выполняется клавишей *Ins*.

Для присвоения переменным числовых значений применяется конструкция:

<имя>:<число>

при вводе двоеточие заменяется символом присвоения [:=]. Вывод результатов выполняется по нажатию клавиши [=]. Знак равенства в условиях и уравнениях вводится комбинацией клавиш Ctrl+=. Набор выражения завершается нажатием *Enter*. Переменная, получившая значение, может быть использована в последующих выражениях, в том числе для обозначения других переменных.

Для набора более сложных выражений может быть использован список встроенных функций, вызываемый кнопкой f(x) стандартной панели инструментов, и математическая панель. При отсутствии математической панели на экране необходимо войти в пункт меню View и установить флажок MathPalette.

Все операции в Mathcad выполняются над выделенными выражениями. Можно выделять часть выражений. Выделение выражений осуществляется мышью путем протаскивания указателя мыши над выделяемым выражением или уголковым курсором синего цвета.

Функции

При щелчке мышью по значку f(x) в панели инструментов стандартная или вводе команды *Вставка, Функция* открывается окно диалога для вставки функций. Все функции для облегчения поиска разбиты на категории. В левой части окна выводится список категорий, а в правой – список соответствующих функций. Выберите нужную категорию, а в ней - требуемую функцию. Для решения вычислительных задач используются категории "логарифм", "тригонометрия",

"усечение и округление". Для ввода встроенных функций удобно пользоваться панелями инструментов, например: *Вид, Панели инструментов, Арифметика.*

Функции, определяемые пользователем

Кроме встроенных функций MathCAD позволяет создавать и функции пользователя. Синтаксис функции пользователя:

<имя функции> (<аргументы>):<выражение>

Функция может быть запомнена на текущий сеанс по Shift+F9. Новую функцию можно определить через функции, заданные ранее, а также через функции, встроенные в MathCAD.

Вычисление выражений

При числовых расчетах в MathCAD переменные считаются, по умолчанию, вещественными переменными удвоенной длины.

Для получения результата после ввода выражения достаточно нажать клавишу "=". Если выражение заканчивается знаком равенства, то для получения результата достаточно нажать клавишу *Enter*. Данный режим действует при включенном режиме автоматического пересчета. В ином случае для получения результата необходимо нажать *F9*. Прервать процесс вычисления можно клавишей *Esc*. Для продолжения прерванного вычисления повторно нажать *F9*.

Пример 7.4.1. Вычислить 5+5

Введите 5+5 нажмите [=] - ответ 10. Ответ появляется после знака [=].

$$5 + 5 = 10$$

Пример 7.4.2. Вычислить y=ab/c при a=5, b=10, c=5,675

Введите a=5 - появится a:=5

Введите b=10 - появится b:=10

Введите c=5.675 - появится c:=5.675

Введите a*b/c - появится a*b/c, нажмите [=] - появится ответ 8.811

$$a := 5 \qquad b := 10 \qquad c := 5.675$$

$$a \cdot \frac{b}{c} = 8.811$$

Для удаления задачи выделите ее курсором и нажмите Del.

Пример 7.4.3. Вычислить выражение

$$y = \frac{\mathrm{Sin}^3(x^2) + e^{-2x} + \arcsin(x)}{\sqrt{2x^3 + 5x} - \sqrt[3]{7x - 6}} + \ln(3x^3)$$

Решение.

$$x := 0.5$$

$$\frac{\exp(-2\cdot x) + \sin\left(x^2\right)^3 + \operatorname{asin}(x)}{\sqrt{2\cdot x^3 + 5\cdot x} - (7\cdot x - 6)^{\left(\frac{1}{3}\right)}} + \ln\left(3\cdot x^3\right) = -0.601 + 0.455i$$

Пример 7.4.4. Протабулировать функцию $y = \cos(x) + e^x$ на отрезке $[0, \pi]$ с шагом 0,5.

Решение.

Для табулирования значений функции используется следующий синтаксис:

z: нач_знач, след_знач; конеч_знач

Здесь нач_знач – начальное значение аргумента, след_знач – следующее значение, конеч_знач – конечное значение аргумента.

Двоеточие, при вводе заменяется символом присвоения [:=], точка с запятой заменяется горизонтальным двоеточием.

Порядок работы:

- введите значение аргументов: z:0, 0.5; 3.14
- запишите функцию: cos(z)+exp(x)=

$z := 0, 0.5 .. 3.14$

$\cos(z) + \exp(z)$

2
2.526
3.259
4.552
6.973
11.381
19.096

После ввода знака равно автоматически генерируется ряд значений функции. Функция выводится с четырехразрядной мантиссой (три знака после десятичной точки). Число разрядов можно изменить командой *Формат, Формат числа*. Можно регулировать также порог мнимости, значение нуля, начальный индекс компонент массива.

7.4.5. Решение типовых математических задач

Решение уравнений

Для решения уравнений вида f(x)=0 переменной x присваивается начальное значение, а затем набирается команда

root(f(x),x)

Полиномиальные уравнения решаются с помощью функции polyroots(a), где a – вектор-столбец коэффициентов многочлена, расположенный в порядке возрастания степеней.

Пример 7.4.5. Решить уравнение $5x^2 - 2x - 7 = 0$

Решение.

$x := 1$

$\text{root}(5x^2 - 2x - 7, x) = 1.4$

То же с использованием функции polyroots(a):

- введем имя функции polyroots;
- щелкнем мышью в панели математика по пиктограмме $\begin{bmatrix}\bullet\bullet\bullet\\ \bullet\bullet\bullet\\ \bullet\bullet\bullet\end{bmatrix}$, в окне

диалога матрицы, щелкнем мышью по пиктограмме матрицы и в открывшемся окне диалога укажем число строк – 3 и число столбцов – 1, нажмем клавишу равно – ответ готов:

$$\text{polyroots}\begin{pmatrix}-7\\-2\\5\end{pmatrix}=\begin{pmatrix}-1\\1.4\end{pmatrix}.$$

Пример 7.4.6. Решить полиномиальное уравнение $2x^5-4x^4+3x^2-2x-7=0$.

Решение. Порядок решения аналогичен порядку, рассмотренному в предыдущем примере.

$$\text{polyroots}\begin{pmatrix}-7\\-2\\3\\0\\-4\\2\end{pmatrix}=\begin{pmatrix}-0.837+0.52i\\-0.837-0.52i\\0.849+1.05i\\0.849-1.05i\\1.975\end{pmatrix}.$$

Решение систем линейных уравнений

Решние систем линейных уравнений в MathCAD осуществляется матричным способом с помощью функции lsolve(A,b), где A – матрица коэффициентов, b – вектор свободных членов. A и b должны быть предварительно сформированы как матрица и как вектор-столбец.

Пример 7.4.7. Решить систему линейных уравнений третьей степени:

$$\begin{cases}6x-4y+7z=3\\-5x+8y+z=8\\2x+5y-4z=-3\end{cases}$$

Решение.

- введите матрицы A и вектор столбец b;
- запишите функцию lsolve(A,b) и нажмите клавишу [=].

$$A:=\begin{pmatrix}6&-4&7\\-5&8&1\\2&5&-4\end{pmatrix}\qquad b:=\begin{pmatrix}3\\8\\-3\end{pmatrix}$$

$$\text{lsolve}(A,b)=\begin{pmatrix}-0.506\\0.538\\1.169\end{pmatrix}$$

Решение систем нелинейных уравнений

Для решения систем нелинейных уравнений используются ключевые слова *Given* и *Find*. Переменным присваиваются начальные значения, уравнения записываются, как обычно принято в математических системах. Для вставки знака [=], исполь-

$x:=1\qquad y:=1$

Given

$x^2+y^2=6$

$x+y=0$

$$\text{Find}(x,y)=\begin{pmatrix}1.732\\-1.732\end{pmatrix}$$

зуется комбинация клавиш *Ctrl+=*. После записи выражения Find(x,y) нажать клавишу [=].

При решении сложных систем нелинейных уравнений может случиться,

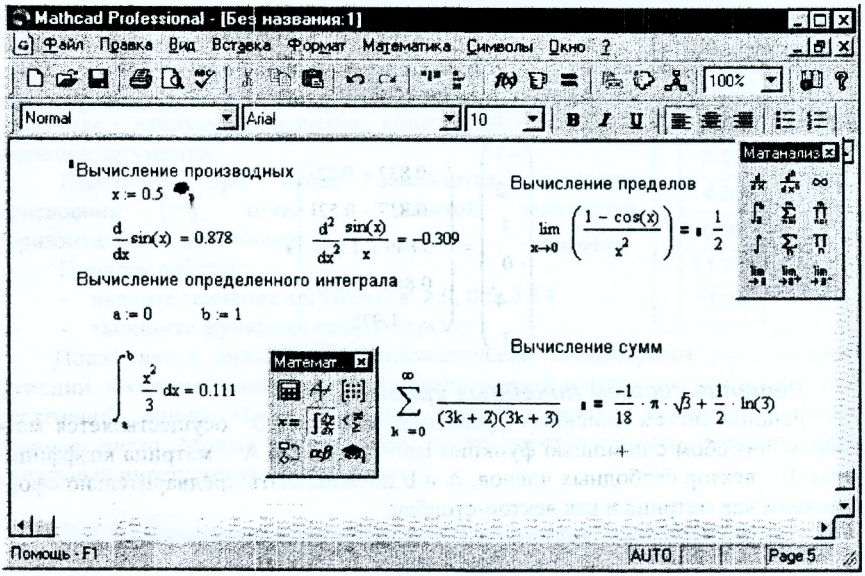


Рис. 7.4.3. Решение задач математического анализа

что программа откажется от решения какой-либо из систем. Это означает лишь, что лежащий в ее основе алгоритм недостаточно универсальный. В этом случае можно рекомендовать алгоритмы из монографии Деннис Дж, Шнабел Р. Численные методы безусловной оптимизации и решения нелинейных уравнений.- М.: Мир, 1988.

Вычисление определенного интеграла, производных, пределов, сумм

Для решения задач математического анализа необходимо воспользоваться панелью инструментов Математический анализ (*Вид, Панели инструментов, Математика* и щелкнуть по пиктограмме *Матанализ* – изображение интеграла) (рис. 7.4.3).

При вычислении пределов и сумм рядов для получения результата необходимо нажать комбинацию клавиш *Shift+F9*. Результат появится ниже формулы (на рисунке для экономии места результат помещен в той же строке и вставлен знак [=]).

Символические вычисления

В пакет MathCAD включены средства символических вычислений: алгебраических операций, операций математического анализа, символического

решения уравнений и дискретных преобразований. Они могут применяться и к выделенным подвыражениям.

Для символьных вычислений служит команда *Символ, Расчеты, Символические*. Получить значения производных и интегралов в символьном виде можно также, применив к выделенному выражению комбинацию клавиш Shift + F9. Примеры:

$$\frac{d}{dx}\frac{(1-\cos(x))}{x^2} \rightarrow \frac{\sin(x)}{x^2} - 2\cdot\frac{(1-\cos(x))}{x^3}$$

$$\int \frac{\ln(x)}{x^2}\,dx \rightarrow \frac{-\ln(x)}{x} - \frac{1}{x}$$

MathCAD позволяет также упрощать выражения и преобразовывать их, используя встроенные тождества. Для этой цели служат команды *Упростить, Расширить, Фактор, Подобные* меню *Символ*. Пример упрощения выражения:

$$\frac{\left(x^2+x-20\right)}{x+5} + 2x + 3 \rightarrow 3\cdot x - 1$$

7.4.6. Построение графиков функций

Для работы с графиками служит команда *Формат, График*. MathCAD позволяет строить графики в декартовых и в полярных координатах, а также трехмерные графики.

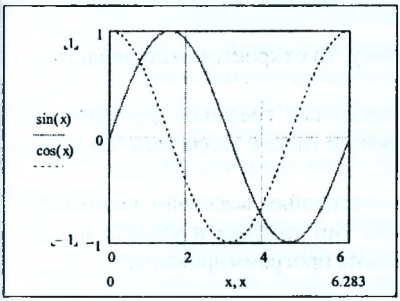


Рис.7.4.4. График функции

Для построения графика необходимо определить функцию и ее аргументы в некотором диапазоне. В Mathcad используется только линейная интерполяция, поэтому для получения плавных кривых шаг табулирования функций должен быть достаточно малым.

Режим построения графика в декартовой системе координат вызывается комбинацией клавиш Shift+@, а трехмерного – Ctrl+@. При вызове мастера построения графиков указанным способом правила построения графика достаточно простые (рис. 7.4.4):

- написать возле левого слота функцию, график которой надо построить и щелкнуть мышью по полю вне бокса.

$a := 0 \qquad b := 2 \cdot \pi \qquad n := 100$

$h := \frac{(b - a)}{n} \qquad i := 1..n \qquad x_i := a + i \cdot h$

$y_i := \sin\left(x_i + 2 \cdot \frac{\pi}{3}\right) \qquad y1_i := \sin(x_i) \qquad y2_i := \sin\left(x_i + 4 \cdot \frac{\pi}{3}\right)$

y =

	0
0	0
1	0.833
2	0.797
3	0.757
4	0.714
5	0.669
6	0.621
7	0.571
8	0.518
9	0.463
10	0.407
11	0.349
12	0.289
13	0.228
14	0.167
15	0.105

y1 =

	0
0	0
1	0.063
2	0.125
3	0.187
4	0.249
5	0.309
6	0.368
7	0.426
8	0.482
9	0.536
10	0.588
11	0.637
12	0.685
13	0.729
14	0.771
15	0.809

y2 =

	0
0	0
1	-0.896
2	-0.922
3	-0.944
4	-0.963
5	-0.978
6	-0.989
7	-0.996
8	-1
9	-0.999
10	-0.995
11	-0.986
12	-0.974
13	-0.957
14	-0.937
15	-0.914

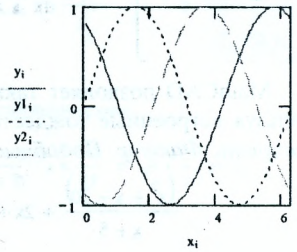


Рис. 7.4.5. Пример решения задачи 7.4.8.

Интервал построения графика по умолчанию установлен от –10 до +10. Этот интервал можно изменить непосредственно на графике: выделить цифры разметки оси X-ов и ввести другое значение.

Если щелкнуть дважды мышью по графику, то откроется окно диалога для изменения описания графика.

Можно на одном графике построить несколько графиков функций. Для этого достаточно записать функции через запятую (после ввода запятой курсор автоматически переходит на другую строку).

При построении графиков по ранее вычисленным массивам необходимо задать диапазон изменения индексов, выбрать тип графика и указать на осях соответственно x[i, y[i . Эта задача уже похожа на программирование.

Пример 7.4.8. Построить график трехфазного тока на отрезке [0,2π] (рис. 7.4.5).

Решение:

- определить начало и конец отрезка табулирования функции *a* и *b*, задать число отрезков *n*;
- вычислить шаг табулирования функции *h*;
- задать интервал изменения индекса I и записать формулу для расчета текущего значения аргумента x_i. Для перехода в режим ввода индекса пере-

менной нажмите клавишу "[", для отмены режима ввода индекса нажмите клавишу *Пробел*;

- определить функции $y_i = \sin(x+2\pi/3)$; $y1_i = \sin(x)$; $y2_i = \sin(x+4\pi/3)$;
- сгенерировать ряды значений y, y1, y2: установить точку вставки в требуемое место экрана, ввести имя переменной и нажать клавишу [=];
- построить графики функций, как описано выше.

7.4.7. Ввод и вывод данных

Рабочий лист MathCAD состоит из отдельных областей или зон, в которых помещаются формулы, результаты, графики, текст и т. п. Области объявляются командой *Вставка, Область*. Область для вывода текста создается командой *Вставка, Текстовая область*. Текстовая область создается так же, если перед вводом текста ввести символ ["] или после набора текста нажать клавишу *Пробел*.

Для ввода русского алфавита необходимо использовать шрифты, имеющие добавление *Cyr*, например, *Arial Cyr, Times New Roman Cyr*.

Ранее созданная область выделяется для редактирования щелчком мыши или обводкой ее мышью. Выделенные перекрывающиеся зоны можно разделить командой Формат, Отделить области.

Результаты работы сохраняются по умолчанию в корневом каталоге системы MathCAD. В режиме *Сохранить как* ... результаты работы могут быть сохранены в любом каталоге.

Контрольные вопросы

1. Что представляет собой кодовая таблица? Для чего она предназначена?
2. Дайте определение: константы, переменные, операторы.
3. Запишите основные математические функции.
4. Что такое математическое выражение? Приведите синтаксис математического выражения и дайте необходимые пояснения?
5. Приведите синтаксис логического выражения.
6. Какие ограничения могут быть при вычислении математических выражений?
7. Поясните назначение математической программы Mercury.
8. Как получить помощь по работе в среде Mercury?
9. Поясните назначение команд главного меню программы Mercury?
10. Как сохранить программу на диске в программе Mercury?
11. Как построить график функции в программе Mercury?
12. Как подготовить отчет в программе Mercury?
13. Как ввести матрицу (вектор) в окно алгебры?
14. Как решить систему линейных уравнений матричным способом?
15. Опишите порядок решения системы линейных уравнений графическим методом в математической системе Derive?.
16. Как вычисляется определитель матрицы в Derive?

17.Что такое характеристическое уравнение, как оно составляется?

18.Что такое вектор невязки, как он вычисляется?

19.Как решить систему линейных уравнений с использованием обратной матрицы в Derive?

20.Что такое вектор собственных значений, как он вычисляется?

21.Что понимается под обусловленностью матрицы?

22. Как сохранить задачу на диске в математической программе Derive?

23. Назовите основные элементы рабочего окна программы MathCAD.

24. Какие "инструменты " имеются в панели инструментов Математика?

25. Перечислите особенности ввода и редактирования текста в MathCAD.

26. Поясните порядок решения полиномиальных уравнений в MathCAD.

27. Поясните порядок решения систем линейных и нелинейных уравнений MathCAD.

28. Как построить график функции в MathCAD?

29. Как построить график функции в MathCAD по ранее вычисленным массивам?

30. Как вычислить производную или интеграл в символьном виде?

8. Алгоритмизация

8.1. Этапы жизненного цикла программ

Процесс подготовки задачи к решению на ЭВМ распадается на несколько этапов. Конечным результатом этой работы является программа. Программа может находиться в эксплуатации длительное время или короткое время в зависимости от ее назначения, качества. Когда отпадает надобность в использовании данной программы, она снимается с эксплуатации, поэтому принято говорить о жизненном цикле программы. Основными этапами этого цикла являются (рис. 8.1.1): постановка задачи, разработка математической модели, разработка

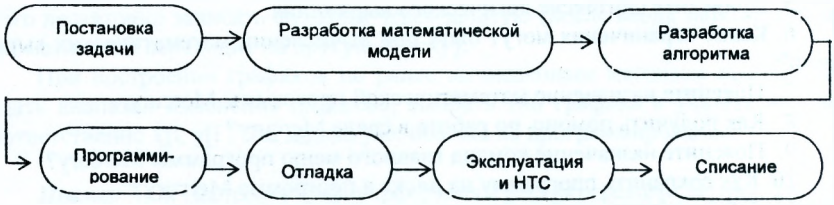


Рис. 8.1.1. Этапы жизненного цикла программы

алгоритма, программирование, отладка программы, передача программы в эксплуатацию и научно-техническое сопровождение (НТС) программы, завершение жизненного цикла.

Постановка задачи

Задача – это проблема, подлежащая решению.

Постановка задачи – один из важных и ответственных этапов жизненного цикла. На данном этапе определяются основные цели и функции, выполнение которых должна обеспечивать программа, исходные данные, требования к исходным данным, выходные данные. От качественной проработки всех вопросов на данном этапе зависит в конечном итоге качество программы и сроки ее разработки. Конечно, в процессе работы над программой многие вопросы могут уточняться, дополняться и т. д., но время разработки программы при этом увеличивается. Различают содержательную и математическую постановку задачи.

Содержательная постановка задачи заключается в формулировке задачи на естественном языке.

Математическая постановка задачи сводится к точному описанию исходных данных, условий задачи и целей ее решения с использованием математических выражений в общем виде. При этом должен применяться системный подход, то есть предмет должен быть исследован всесторонне, учтены все внешние и внутренние связи и их влияние на конечные результаты.

Любую задачу можно представить в виде "черного ящика" (рис. 8.1.2.), на вход которого поступают исходные данные – вектор $\overline{X}$, ограничения на входные параметры - вектор $\overline{Q}$, требования к входным и выходным параметрам – вектор $\overline{T}$, а выходом является вектор $\overline{Y}$. Ни для заказчика, ни для разработчика программы на данном этапе не имеет значение, каким образом будет обрабатываться информация.

Разработка математической модели

На данном этапе производится декомпозиция задачи, формализация, разработка математической модели, выбор метода решения.

Под ***декомпозицией*** понимается разделение задачи на простые блоки, каждый из которых может разрабатываться самостоятельно и связан с другими частями программы только входными и выходными данными.

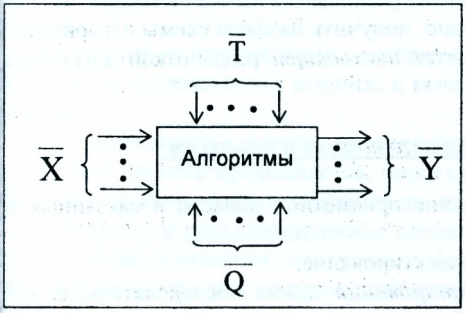


Рис. 8.1.2. Модель решения задачи

Для деления задачи на блоки чаще всего используется функциональный подход. Например, в каждой вычислительной задаче можно выделить такие блоки как ввод данных, вычислительный блок, блоки сохранения результатов вычислений на дисках, анализа результатов вычислений, графического представления результатов вычислений, печати результатов.

Под ***формализацией*** задачи понимают представление исходных данных и условий решения задачи, сформулированных словесно, в виде, удобном для ввода и последующей обработки в ЭВМ.

Математическая модель задачи представляет собой совокупность математических выражений, описывающих данную задачу. В общем виде математическую модель можно представить следующим образом:

Функция_цели = $F(x_1, x_2, \ldots, x_n, q_1, q_2, \ldots, q_m, t_1, t_2, \ldots, t_p)$
min

Функциональные ограничения:

$q_1 <= \varphi_1(x_1, x_2, \ldots x_n)$

$q_2 <= \varphi_2(x_1, x_2, \ldots x_n)$

...

$q_m = \varphi_m(x_1, x_2, \ldots x_n)$

Ограничения на значения параметров:

$x_1 >= T_1$

$x_2 >= T_2$

...

$x_n >= T_n$

Метод решения задачи выбирается из известных методов. Если для решения данной задачи имеется несколько методов, то выбирается тот метод, который обеспечивает получение решения за меньшее время и с заданной точностью. Если для решения данной задачи нет известных методов, то необходимо разработать такой метод или вернуться на первый этап и уточнить задачу, исходные данные и требования к ним.

Разработка алгоритма программы

На данном этапе разрабатывается алгоритм решения задачи, то есть определяется последовательность действий. Разработка алгоритма предполагает определение состава функциональных модулей и формирование общей схемы алгоритма, разработку алгоритмов функциональных модулей. В зависимости от сложности задачи алгоритм представляют вначале в общем виде (укрупненном). Затем каждый из блоков алгоритма разбивается на более мелкие задачи таким образом, чтобы на конечном этапе получить базовые схемы алгоритмов. Такой метод проектирования называется *нисходящей* разработкой алгоритма (проектирования).

Основные подходы к разработке алгоритмов и программ:

- структурное проектирование;
- информационное моделирование предметной области и связанных с ней приложений;
- объектно-ориентированное проектирование.

В основе структурного проектирования лежит последовательная декомпозиция, целенаправленное структурирование на отдельные составляющие. Типичными методами структурного проектирования являются:

- нисходящее проектирование, кодирование и тестирование программ;
- модульное программирование;

- структурное программирование.

Нисходящее проектирование предполагает разложение общей функции обработки данных на простые функциональные элементы "сверху – вниз" (рис. 8.1.3). В результате образуется *функциональная схема алгоритма.*

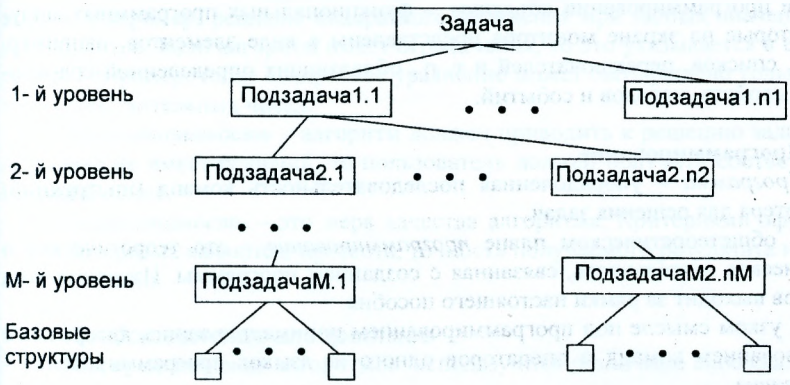


Рис. 8.1.3. К понятию "нисходящее проектирование"

При реализации принципа нисходящего проектирования программа разрабатывается в следующей последовательности:

- на основе анализа задача разбивается на подзадачи, выделяются уровни и подуровни, функции отдельных блоков. В результате составляется иерархическая структура – функциональная структура программы;
- для каждого уровня:

– разрабатывается математическая модель и определяется метод решения задачи;

– определяются основные блоки программы и разрабатывается укрупненная схема алгоритма;

– определяются входные и выходные переменные, общие для данного уровня;

– разрабатываются схемы алгоритмов для реализации основных блоков программы, определяются входные и выходные переменные соответствующего уровня;

– определяется порядок взаимодействия с другими блоками.

Этот процесс продолжается, пока схема алгоритма не будет доведена до базовых структур соответствующего языка программирования.

Модульное программирование основано на понятии модуля. *Модуль* – логически взаимосвязанная совокупность функциональных элементов, оформленных в виде отдельных программных модулей, имеющих один вход и один выход.

Структурное программирование основано на модульной структуре программного продукта и типовых управляющих структурах алгоритмов обработки данных различных программных модулей.

Информационное моделирование предметной области и связанных с ней приложений предполагает определение состава и способа представления исходных данных и результатов вычислений.

Объектно-ориентированное проектирование основано на использовании при программировании *объектов* - функциональных программных модулей, которые на экране монитора представлены в виде элементов, например, кнопок, списков, переключателей и т. п., обладающих определенной совокупностью свойств, методов и событий.

Программирование

Программа – упорядоченная последовательность команд (инструкций) компьютера для решения задач.

В общетеоретическом плане *программирование* – это теоретическая и практическая деятельность, связанная с созданием программы. Изучение этих вопросов выходит за рамки настоящего пособия.

В узком смысле под программированием понимается запись алгоритма с использованием команд и операторов одного из языков программирования - *кодирование*.

Отладка программы

Отладка программы заключается в проверке правильности функционирования алгоритма решения задачи с помощью контрольных примеров, результаты решения которых заранее известны, устранении обнаруженных синтаксических и логических ошибок.

Научно-техническое сопровождение

Научно-техническое сопровождение программы предусматривает контроль за работой программы и устранение ошибок, обнаруженных в процессе эксплуатации, доработку программы и ее совершенствование в соответствии с требованиями заказчика.

8.2. Схема алгоритма

Алгоритм - точное, однозначное предписание последовательности действий (операций), приводящее к решению задач данного класса за конечное число шагов или заданное время.

Основными свойствами алгоритма являются дискретность, определенность, массовость, результативность, эффективность.

Дискретность – данное свойство вытекает из самой сущности цифровой вычислительной техники. Все вычисления выполняются последовательно, шаг за шагом, таким образом, что результаты вычислений на предыдущем шаге используются на последующем шаге.

Определенность – алгоритм не должен допускать различных толкований. Действия, которые необходимо произвести, должны быть строго и недвусмысленно определены в каждом конкретном случае.

Массовость – алгоритм должен позволять решать все задачи данного класса. Например, решение квадратного уравнения при любых значениях коэффициентов. Если алгоритм имеет ограничения, то это указывается в его описании. Например, то же квадратное уравнение может быть решено только в области действительных чисел.

Результативность – алгоритм должен приводить к решению задачи. Если задача не имеет решений, то пользователь должен получить соответствующее сообщение.

Эффективность – это мера качества алгоритма. Критериями эффективности алгоритма являются: простота, точность получаемого результата и время его реализации.

Представление алгоритмов

Для представления алгоритмов используются различные способы:

- словесное;
- на алгоритмическом языке;
- в виде схем алгоритмов;
- предикатная форма записи.

Словесная форма представления алгоритма самая простая. Словесная форма позволяет подробно описать последовательность действий, рассмотреть все возможные варианты. Недостатком данной формы описания является ее громоздкость и невозможность в ряде случаев перейти непосредственно от алгоритма к программе. Тем не менее, при разработке сложных алгоритмов без использования словесной формы нельзя обойтись.

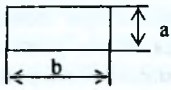
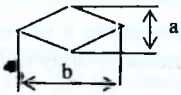
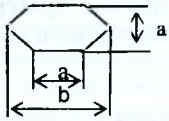
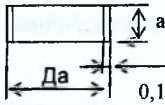
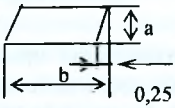
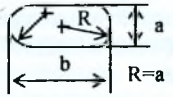
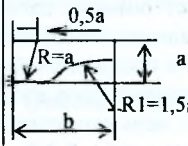
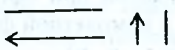
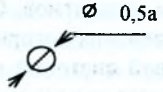
Запись алгоритма на алгоритмическом языке используется, как правило, при обучении. Эта форма записи позволяет описать алгоритм на языке близком к естественному. В качестве примера алгоритмического языка можно назвать язык Мега-Е, используемый в школах и в средних учебных заведениях для обучения программированию. В состав служебных слов входят такие слова как: **алг** – алгоритм, **дано** – ввод данных, **надо** – цель выполнения алгоритма и др. Основные конструкции этого языка приведены в табл. 8.2.2.

Предикатная форма записи алгоритма удобна при оформлении документации. Она позволяет представить алгоритм в компактной форме, но не обладает наглядностью, например: P(x<y, a, b); P1(x, y, g , h).

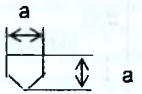
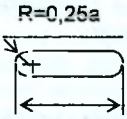
Схема алгоритма. На практике для представления отчетов и оформления проектной документации чаще используются схемы алгоритмов. Схема алгоритма является одним из способов наглядного представления алгоритма с помощью специальных элементов, предусмотренных единой системой конструкторской документации (ЕСКД). Некоторые из этих элементов приведены в табл. 8.2.1.

Таблица 8.2.1

Основные графические элементы схем алгоритмов

Наименование	*Обозначение*	*Функция*
Процесс		Выполнение операции или группы операций, в результате которых изменяется значение, форма представления или расположение данных
Выбор		Выбор направления выполнения алгоритма или программы в зависимости от некоторых переменных условий
Модификация		Выполнение операций, меняющих команды, или группы команд, изменяющих программу
Предопределенный процесс		Использование созданных ранее или отдельно описанных алгоритмов или программ
Ввод-вывод		Преобразование данных в форму, пригодную для обработки (ввод) или отображения результатов обработки (вывод)
Дисплей		Ввод данных с подключенного к компьютеру дисплея или вывод данных на дисплей
Документ		Ввод-вывод данных, носителем которых служит бумага
Линии потока		Указание на последовательность связи между символами. Можно без стрелки, если линия направлена слева направо или сверху вниз, со стрелкой в остальных случаях
Соединитель		Указание на связь между прерванными линиями потока, соединяющими операторы в пределах листа.

Продолжение таблицы 8.2.1

Наименование	*Обозначение*	*Функция*
Соединитель		Указание на связь между прерванными линиями потока, соединяющими операторы на разных листах.
Пуск-останов		Начало, конец, прерывание процесса обработки данных или выполнения программы
Комментарий		Связь между элементами схемы с пояснениями

Размеры элементов схемы строго определены. Размер стороны *a* выбирается из ряда 10, 15, 20, … мм с шагом 5 мм. Размер стороны *b* принимается равным *1,5a*, допускается *2a*. Элементы схемы объединяются линиями потока. Стрелки на линиях потока указываются, если поток направлен справа налево или снизу вверх. Допускается изображать стрелки во всех направлениях. Стрелка проставляется на линии потока в конце потока. Блок выбора имеет один вход (сверху или снизу) и два выхода (снизу/сверху и влево или вправо). Блоки в схеме алгоритма нумеруются. Блок "Начало" не нумеруется. Номера блоков проставляются у левого верхнего края блока или в разрыве линии.

При программировании многие структуры алгоритмов повторяются. Различают три основные (базовые) структуры алгоритмов: линейную, выбор и цикл. Эти базовые структуры одинаковы для всех языков программирования. Отличия могут заключаться только в форматах используемых операторов. Базовые структуры схем алгоритмов приведены в табл. 8.2.2.

Линейная структура представляет собой последовательность операторов, следующих один за другим, ветвления и циклы отсутствуют.

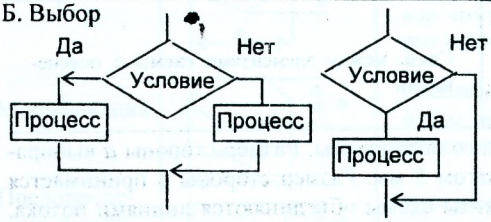
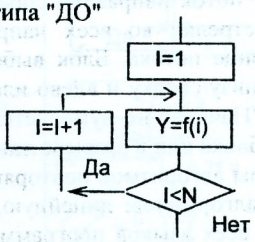
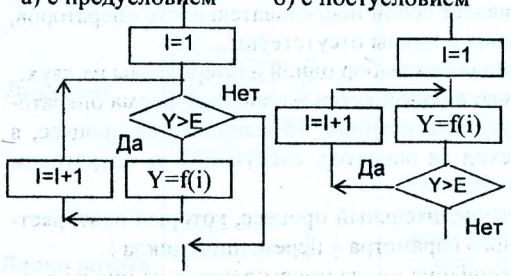
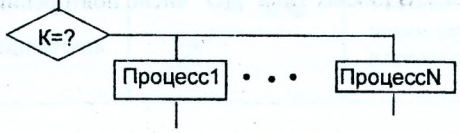
Выбор (или решение) - предполагает выбор одной альтернативы из двух. При этом процесс может быть только в одной ветви – неполная форма оператора выбора. В этом случае, если условие истинно, то выполняется процесс, в ином случае осуществляется переход на оператор, следующий за оператором выбора.

Цикл – представляет собой вычислительный процесс, который повторяется многократно при изменении одного параметра – переменной цикла I.

В зависимости от условия окончания цикла циклы делятся на циклы с заданным числом повторений (циклы типа "ДО") и циклы с параметром (бесконечные циклы или циклы типа "Пока"). В циклах типа "ДО" число повторений

Таблица 8.2.2

Базовые структуры алгоритмов

А. Следование 	*нач* | <выражение> | ... | <выражение> *кон*
Б. Выбор 	*если* <условие> | *то* <выражение> | ... | *иначе* <выражение> *все*
В. Цикл типа "ДО" 	*нц для* I *от* N1 *до* N2 *шаг* N3 | <выражение> | ... | <выражение> *кц*
Г. Цикл типа "Пока" а) с предусловием б) с постусловием 	*нц пока* <условие> | <выражение> | ... | <выражение> *кц*
Д. Вычисляемый оператор перехода 	*выбор* | при условии 1:<выраж.> | . . . | при условии N:<выраж.> *все*

известно перед началом цикла и условием окончания цикла является выполнение заданного числа повторений. В циклах типа "Пока" число повторений неизвестно заранее, условием окончания цикла является достижение некоторой переменной, вычисляемой в теле цикла, определенного, наперед заданного значения.

В зависимости от момента, когда проводится проверка условия окончания цикла, циклы делятся на циклы с предусловием и циклы с постусловием. В циклах с предусловием проверка условия окончания цикла проводится в начале цикла, а затем выполняется тело цикла. В циклах с постусловием проверка условия окончания цикла выполняется после выполнения тела цикла.

Как разновидность операторов выбора можно указать структуры алгоритмов, позволяющие осуществлять выбор не из двух, а из множества альтернатив. Структура такого алгоритма приведена в табл.8.2.2. д.

8.3. Примеры разработки алгоритмов типовых структур

Линейные структуры

Задача 8.3.1. Даны переменные A и B. Требуется поменять их значения.

Решение.

1. Выбор метода решения.

Ввести вспомогательную переменную T. Сохранить значение переменной A в переменной T, присвоить переменной A значение B, присвоить переменной B значение вспомогательной переменной T.

2. Алгоритм решения будет включать пять операторов (рис. 8.3.1).

Запись алгоритма решения задачи на алгоритмическом языке

```
алг обмен_данными
    дано A, B
    надо поменять значения
нач
   T := A
   A := B
   B:= T
   рез A, B
кон
```

3. Запись алгоритма на языке программирования (QBasic):

```
INPUT "Введите значение A и B ", a,b
T=a: a=b: b=T
PRINT a,b
END
```

4. Для отладки программы достаточно ввести значения a и b и проконтролировать результат.

Задача 8.3.2. Вычислить площадь треугольника, если известны длины сторон a, b, c.

Решение.

1. Выбор метода решения.

Задача может быть легко решена, если воспользоваться формулой Герона: площадь треугольника равна корню квадратному из произведения полупериметра на разности полупериметра и каждой из сторон треугольника.

2. Математическая модель задачи может быть представлена двумя формулами:

$$p = 1/2(a + b + c) \qquad (8.3.1)$$

$$S = \sqrt{p(p-a)(p-b)(p-c)} \qquad (8.3.2)$$

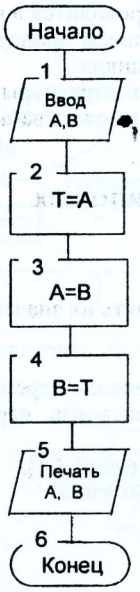


Рис.8.3.1. Схема алгоритма

Для оформления отчетной документации необходимо описать все переменные, используемые в программе по форме, представленной на (рис. 8.3.2). Это позволит также систематизировать исходные данные и результаты вычислений, избежать ошибок при программировании, например, повторного использования имен переменных. Схема алгоритма решения данной задачи аналогична схеме, представленной на рис. 8.3.1.

2. Запись алгоритма на языке программирования:

```
REM Вычисление площади треугольника
INPUT "Введите длины сторон A, B, C ",a, b, c
p=(a+b+c)/2
S=SQR(p*(p-a)*(p-b)*(p-c)
PRINT "S=";S
```

4. Отладка программы. Для отладки программы необходимо ввести данные, для которых результат известен или может быть рассчитан вручную. Например, для прямоугольного треугольника с равными сторонами длина третьей стороны равна $a\sqrt{2}$, а площадь S равна $1/2a^2$. При a=1, S= 1,41.

Схемы разветвляющихся алгоритмов

С выбором вариантов действий человек сталкивается постоянно в повседневной жизни, часто не задумываясь об этом: пойти на лекцию или в кино, пойти с другом в театр или на дискотеку, купить нужную вещь или подождать. Выбор решения при этом зависит как от внутренних, так и от внешних факторов: есть настроение или его нет, есть на улице дождь или нет, есть деньги в бумажники или нет и т. д.. Выбор решения сопровождается как правило постановкой вопроса "Что будет, если ... ?". Выбор может быть простой или сложный. При простом выборе имеется всего два варианта решения (альтернативы), при сложном выборе – несколько. При наличии двух альтернатив задача выбора формулируется следующим образом:

Если <условие> *То* <действие первое> [, *иначе* <действие второе>]

Здесь *если, то* и *иначе* - ключевые слова. Если число альтернатив больше двух,

STREUG (Имя программы)	Вычисление площади треугольника (Назначение программы)		Лист	1
			Листов	1

Переменная			Комментарий
обозначение в формуле	имя переменной	тип переменной	
a	x	вещественная	длина стороны a
b	y	вещественная	длина стороны b
c	z	вещественная	длина стороны c
p	p	вещественная	длина полпериметра

Рис. 8.3.2. Форма для описания переменных задачи

то операция выбора повторяется многократно, сравнивая последовательно две альтернативы.

Понятие о полной группе событий

События, связанные с выбором решения, образуют полную группу событий, если к данной группе событий нельзя добавить никакие другие события. Например, если в урне находятся белые и черные шары, то при последовательном вынимании шаров из урны возможно только два исхода: вынут белый шар или черный шар. При бросании монеты возможно тоже два исхода: монета легла "орлом" или "решкой" (вероятность того, что монета встанет на ребро очень мала и ей можно пренебречь).

Выберем на числовой оси *X* произвольную точку *A*. Случайная величина *x* может в этом случае принять по отношению к точке *A* три значения: меньше A, больше A или равно A. Это полная группа событий. Графическое представление этих условий приведено на рис. 8.3.3.

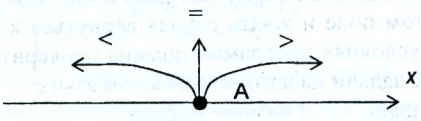


Рис. 8.3.3. Полная группа событий для точки на числовой оси

Условия записываются с помощью условных выражений. В условных выражениях в качестве операндов используются переменные и выражения, операнды соединяются знаками отношений (табл. 7.1.3):

Примеры простых условных выражений:

x < a; x > a; Sin(x) <= 1; a+b <= a*b, при a <> b и a > 1, b > 1

Примеры сложного выражения: a < x < b

То же самое можно записать с использованием логического оператора AND:

x > a AND x < b

Примеры

Задача 8.3.3. При первом запуске программа запрашивает имя пользователя и запоминает его. При последующих запусках она запрашивает имя пользователя и сравнивает его с тем, что записано в памяти. Если ответ правильный, то программа продолжает работу, если нет, то программа завершает работу. Написать программу для проверки владельца компьютера по его имени.

Рис. 8.3.4. Схема алгоритма к задаче 8.3.3.

Решение. Имя пользователя в данной задаче играет роль пароля для доступа к информации, обозначим его "П". Опишем решение на алгоритмическом языке.

```
алг Проверка_пароля
  дано  Имя
  нач
     если   Имя = П
        то продолжить работу
     иначе
        конец программы
     Все
  кон
```

Задача 8.3.4. Известная сказка о вещем камне: «Налево пойдешь - жизнь потеряешь, прямо пойдешь – коня потеряешь, направо пойдешь – женату быть». Требуется разработать алгоритм для решения данной задачи.

Решение. Выполним формализацию задачи: введем обозначения: налево – 1; прямо – 2; направо – 3, выбор пользователя обозначим i. Группа событий неполная, так как неизвестно, что будет, если вариант не выбран (вероятно, герой сказки в этом случае должен вернуться домой или как его старшие братья остаться гулять в чистом поле и ждать случая вернуться к отцу не с пустыми руками). При данных условиях программа должна проверить все варианты выбора. Опишем решение задачи на алгоритмическом языке.

```
алг Вещий_камень    дано  i     | вариант выбора
    нач
       если      i = 1
          то  Текст1 =  "Жизнь потеряешь": Вывод Текст1

    все
       если  i = 2
          то Текст1 = "Коня потеряешь": Вывод Текст1
       все
       если  i = 3
          то Текст1 =  "Женату быть": Вывод Текст1
          иначе  Текст2 = "Нет выбора": Вывод Текст2
       все
    кон
```

Схема алгоритма приведена на рис. 8.3.5.

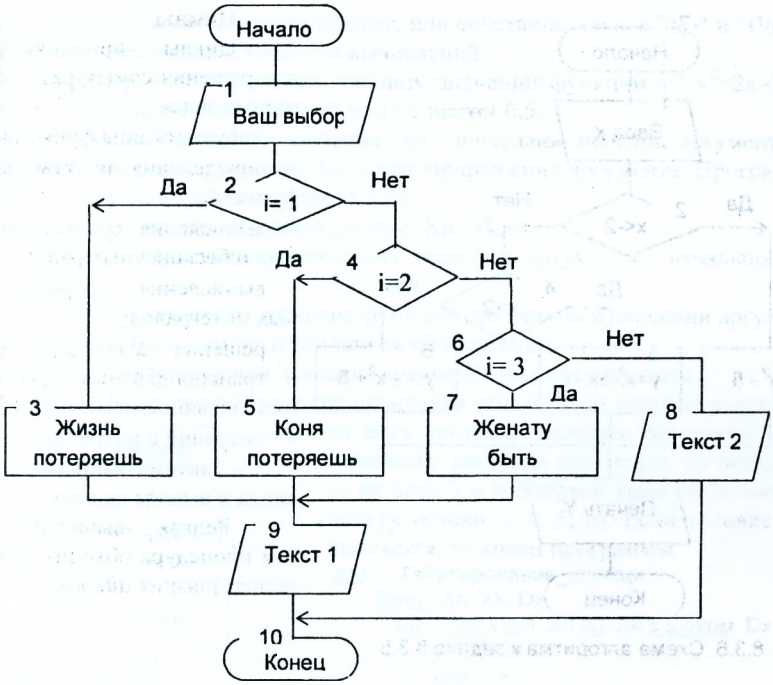


Рис. 8.3.5. Схема алгоритма к задаче 8.3.4

Задача 8.3.5. Вычислить значения функции

$$Y = \begin{cases} x^2-5, & \text{если } x<-2 \\ -x^2+5, & \text{если } -2 \le x \le 2 \\ x^3-5x-1, & \text{если } x>2 \end{cases}$$

Решение. В данной задаче группа событий полная, поэтому для вычисления значения Y при любых значениях аргумента достаточно выполнить две проверки. Схема алгоритма приведена на рис. 8.3.6.

Опишем решение задачи на алгоритмическом языке.

```
алг условный_оператор
    дано x
нач
    если  x < -2
          то  y = x^2 - 5
          иначе  если  x > 2
                 то  y = x^3 - 5x –1
                 иначе y = - x^2 + 5
          все
    все
    рез  y
кон
```

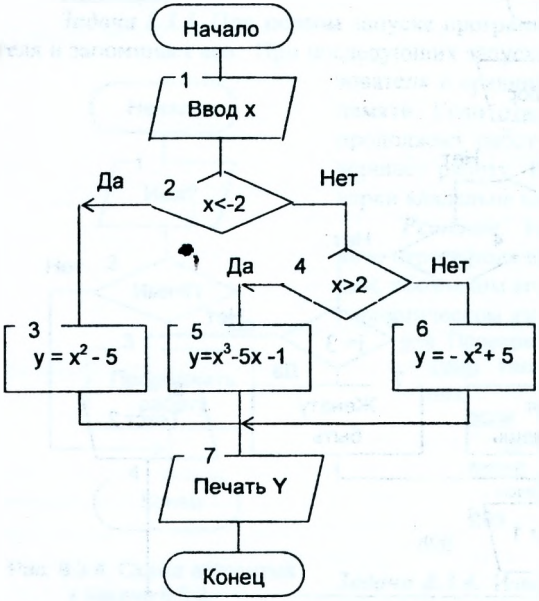


Рис. 8.3.6. Схема алгоритма к задаче 8.3.5

Циклы

Циклы применяются для решения самых разнообразных задач:

– табулирования функций;
– определения экстремумов функций;
– вычисления сумм конечных и бесконечных рядов;
– вычисления определенных интегралов;
– решения алгебраических и трансцендентных уравнений численными методами;
– операций с матрицами;
– автоматизированного ввода и вывода данных.

Редкая вычислительная процедура обходится без использования циклов.

Табулирование функции одной переменной

Под табулированием понимают конструирование или вычисление (составление) различных математических таблиц.

Наиболее широко этот метод использовался до появления вычислительной техники. Например, для вычисления значений тригонометрических функций, интегралов широко использовались таблицы, рассчитанные в научных центрах и представленные в виде таблиц. Табличные методы задания функций могут применяться и в малых вычислительных машинах для сокращения времени вычисления с использованием методов интерполяции и экономии памяти, необходимой для хранения сложных программ вычисления по точным формулам. Табулирование может применяться также для построения графиков функций.

Суть табулирования функции состоит в том, что весь диапазон изменения независимой переменной разбивают на равные интервалы и для каждого значения аргумента в граничных точках интервалов *(узлах интерполяции)* вычисляется значение функции одним из известных методов с требуемой точностью. Результаты расчетов представляются в виде таблицы, в одной из колонок которой приводятся значения аргумента, а в другой - соответствующее ему значения функции. Алгоритм расчета представляет собой цикл типа "До", так как

число шагов обычно известно заранее, или сочетание циклов "До" и "Пока", если заданы требования к точности вычислений.

Пример 8.3.6. Составить таблицу значений функции $y= x^2+2x-4$ для x, изменяющегося в диапазоне от 0 до 10 с шагом 0,5.

Решение. Введем обозначения: Xn – начальное значение аргумента, Xk – конечное значение аргумента, Dx – шаг приращения аргумента. Программа работает следующим образом (рис. 8.3.7):

Блок 1. Вводим исходные данные: Xn, Xk, Dx.

Блок 2. Присваиваем текущему значению аргумента x начальное значение Xn.

Блок 3. Вычисляем значение функции при текущем значении аргумента.

Блок 4. Выводим на экран или печать значения x и y.

Блок 5. Увеличиваем значение аргумента на величину шага.

Блок 6. Проверяем условие окончания цикла. Если условие выполняется, то есть текущее значение аргумента меньше конечного значения аргумента, то возвращаемся на блок 3 и повторяем вычислительную процедуру (блоки 3, 4, 5, 6). Если условие не выполняется, то конец программы.

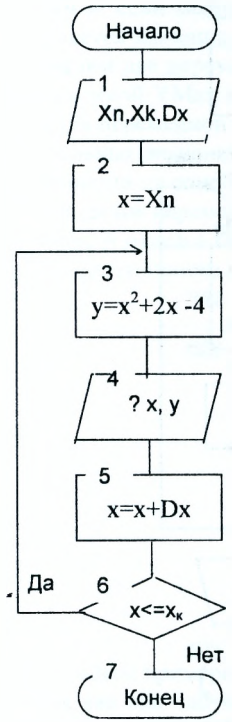


Рис. 8.3.7. Схема алгоритма к задаче 8.3.6

```
алг   Табулирование_функции
   дано  Xn, Xk, Dx
     нц  для x от Xn до Xk с шагом Dx
         y = x²+2x -4
         рез x, y
      кц
кон
```

Пример 8.3.7. Вычислить сумму и произведение чисел натурального ряда от 2 до 100.

Решение. Введем обозначение Xn – начальное значение аргумента, N – число членов ряда, Dx – шаг, S – сумма чисел, P – произведение чисел.

```
алг Вычисление суммы
   дано Xn, N
нач
    S=0
    P=1
    нц  для i от Xn до N с шагом Dx
        S=S+i
        P=P*i
    кц
    рез S, P
кон
```

В данном алгоритме (рис. 8.3.8) для вычисления суммы чисел переменной S до входа в цикл присваивается начальное значение нуль S=0, а для вы-

числения произведения переменной P присваивается начальное значение единица P=1. В теле цикла проводится вычисление S: к текущему значению суммы прибавляется текущее значение переменной цикла S=S+i. Для вычисления произведения текущее значение произведения P умножается на текущее значение переменной цикла – P=P*i.

Данный алгоритм не отвечает свойству массовости, так как не позволяет

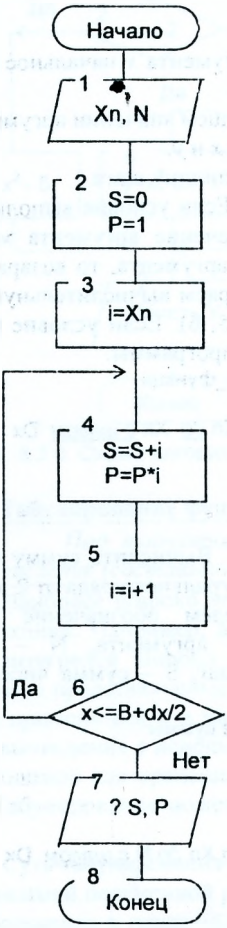


Рис. 8.3.8. Схема алгоритма к задаче 8.3.7

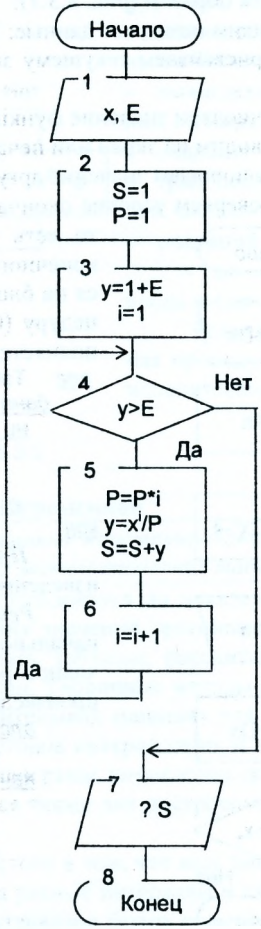


Рис. 8.3.9. Схема алгоритма к задаче 8.3.9

вычислять отдельно сумму четных или нечетных чисел. Чтобы обеспечить такую возможность, необходимо формировать запрос, что надо подсчитать. Кроме того, необходимо обеспечить контроль значения Xn, так как при вычисле-

нии произведения чисел Xn не может быть равным нулю, при вычислении суммы или произведения четных чисел начальное значение Xn должно быть четным, а при вычислении суммы или произведения нечетных чисел – нечетным.

Проверить четность числа можно с помощью функции MOD. Функция MOD вычисляет отношение двух целых чисел и возвращается значение 0, если остаток от деления равен нулю, то есть число четное, и возвращает значение один, если остаток от деления не равен нулю, то есть число нечетное. Например: 5 MOD 2, - результат 1).

Пример 8.3.8. Найти приближенное значение глобальных экстремумов функции y=F(x) на отрезке [A,B].

Решение. Задача решается путем табулирования функции с некоторым, достаточно малым шагом. На каждом шаге табулирования выполняется сравнение вычисленного значения функции с некоторой переменной, принятой за максимум и, если значение функции больше этой переменной, то переменной присваивается значение функции. Аналогично выполняется поиск минимума.

Введем две переменные YMax и YMin. В качестве начального значения для переменной YMax можно взять большое отрицательное число, например - 1E38, а для переменной YMin – большое положительное число - +1E38. В качестве начальных значений переменных YMax и YMin можно принять также значение функции на одной из границ отрезка табулирования функции.

Присвоим переменным YMax и YMin значения, равные значению функции в точке A отрезка табулирования функции.

Алгоритм поиска минимума и максимума функции:

```
алг    Поиск_мин_max
   дано A, B, Dx
нач
   Ymax= F(A): YMin=F(A)
   ни   для x от A до B+Dx/2 с шагом Dx
        y=F(x)
        если y>Ymax
          то  YMax= y
        все
        если  y<YMin
          то  YMin= y
        все
    кц
    рез YMax, YMin
кон
```

В данном примере y=F(x) - любая заданная функция.

Так как шаг табулирования выбирается произвольно, то можно потерять значение функции на конце отрезка. Чтобы этого не произошло, условием окончания цикла принимается не B, а B+Dx/2. Схема алгоритма во многом будет похожа на схему, представленную на рис. 8.3.8, только в тело цикла включаются проверки значений функции на минимум и максимум.

Пример 8.3.9. Составить таблицу значений функции e^x для заданного значения x с точностью 0,0001.

Решение. Функцию e^x, можно представить в виде ряда

$$e^x = 1 + \frac{x^1}{1} + \frac{x^2}{1*2} + \frac{x^3}{1*2*3} \ldots + \frac{x^i}{i!} + \ldots$$

или

$$e^x = 1 + \sum_{1}^{\infty} \frac{x^i}{i!}, \qquad (8.3.3)$$

Данный ряд сходящийся. Под точностью в данном случае понимается значение отбрасываемого члена ряда. Полная ошибка может быть в несколько раз больше значения отбрасываемого члена ряда. Для знакопеременных рядов ошибка вычисления суммы сходящегося ряда не превышает значения отбрасываемого члена ряда.

Алгоритм вычисления функции реализуется с помощью цикла типа "Пока" (рис.8.3.9).

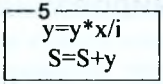
Первый блок вводит исходные данные: x - значение аргумента и E — точность вычисления значения текущего члена ряда. Второй и третий блоки осуществляют начальное присвоение переменной цикла, S – начальное значение суммы, P – вспомогательная переменная для вычисления факториала. Начальное значение суммы равно единице, так как в выражении 8.3.3. имеется соответствующее слагаемое. Начальное значение y также принимаем равным 1+E, чтобы на первом шаге программа вошла в цикл. Четвертый блок проверяет условие окончания цикла – достижение заданной точности вычисления текущего члена ряда. Если условие выполняется, то выполняется тело цикла, иначе осуществляется переход на блок 7 – печать результата и выход из программы – блок 8. Пятый блок обеспечивает вычисление значения функции: вычисляется текущее значение факториала - P, значение текущего члена ряда – Y, и значение функции на текущем шаге – S.

С точки зрения вычислительной математики блок 5 построен нерационально, так как на каждом шаге необходимо вычислять выражение $x^i/(i!)$, кроме того, при больших значениях i наступает переполнение разрядной сетки. Скорректируем математическую модель:

$$e^x = 1 + \sum_{1}^{\infty} \frac{\prod_{1}^{i} x}{\prod_{1}^{i} i} = 1 + \sum_{1}^{\infty} \prod_{1}^{i} \frac{x}{i} \qquad (8.3.4)$$

В данной модели степень переменной x заменена ее произведением, а в числителе факториал заменен произведением ряда натуральных чисел от единицы до i. При такой модели значение текущего элемента будет равно значению предыдущего элемента умноженного на x и деленного на i.

С учетом сделанных замечаний блок 5 алгоритма (рис. 8.3.8) будет выглядеть следующим образом:



Данный пример показывает, как от правильного построения математической модели зависит эффективность алгоритма.

Контрольные вопросы и упражнения

1. Назовите основные этапы жизненного цикла программы.
2. Охарактеризуйте основные этапы жизненного цикла программы.
3. Приведите определение алгоритма, назовите и поясните основные свойства алгоритма.
4. Что такое нисходящая разработка?
5. Что такое модуль, в чем заключается особенность модульного проектирования?
6. В чем состоит особенность структурного программирования?
7. Изобразите основные элементы схем алгоритмов.
8. Изобразите базовые схемы алгоритмов.
9. Опишите базовые структуры на алгоритмическом языке.
10. Изобразите структуру линейной программы.
11. Пусть S= -1, если x<0; S=0, если x=0; S=1, если x>0. Опишите решение задачи на алгоритмическом языке и с помощью схемы алгоритма.
12. Разработайте схему алгоритма для решения квадратного уравнения в области действительных чисел.
13. Протабулируйте функцию $y=Sin(x)+1/x^2$ на отрезке [-3,14; 3,14] и найти максимум функции. Описать решение задачи на алгоритмическом языке и с помощью схемы алгоритма.
14. Разработайте алгоритм для печати таблицы умножения.
15. Разработать алгоритм программы для суммирования штрафного времени при игре в хоккей.
16. Разработать схему алгоритма программы для печати результатов соревнования по плаванью. В заплыве плывут М спортсменов. На экран выводить список участников заплыва в порядке показанных результатов. На экран выводить также мировой и европейский рекорды, а также фамилии участников заплыва, превысивших эти рекорды.
17. Разработать схему алгоритма для определения дальности полета снаряда. Известна высота орудия, начальная скорость стрельбы и угол стрельбы.

9. Система программирования QBasic

9.1. Историческая справка о языке программирования Basic

Бейсик системы имеют уже несколько поколений языков программирования, обусловленных изменением технических характеристик ПК, а также потребностями программистов и пользователей.

Язык программирования Basic был разработан сотрудниками Дартмутского колледжа во главе с Дж. Кемени и Т. Куртцем в мае 1964 года и предназначался для обучения программированию. Разработку языка поддерживала фирма Дженерал Электрик. Новый алгоритмический язык обеспечивал режим диалогового программирования. Язык программирования быстро получил большое распространение прежде всего благодаря простоте, а также тому, что к нему проявили большой интерес фирмы производители мини-ЭВМ.

К оригинальным разработкам Бейсик - интерпретаторов для отечественных ЭВМ того периода относятся: М-20 (Горьковский университет, 1970 г.); Basic- 6 – использовался на ЭВМ БЭСМ-6 (Вычислительный центр АН СССР, 1972 г.); Бейсик-Гамма – на ЭВМ Минск-32 (Институт математики АН БССР, 1974 г.).

Бейсик системы могут быть двух типов: интерпретирующие и компилирующие.

В системах *интерпретирующего* типа сразу после ввода программной строки проводится ее анализ и при обнаружении ошибки выдается сообщение об этой ошибке. По окончании набора программы ее можно сразу же поставить на исполнение непосредственно из среды разработки.

В системах *компилирующего* типа программа должна быть сначала полностью набрана, после чего она ставится на компиляцию. При этом проводится проверка программы, обнаружение синтаксических ошибок и создание объектного модуля, который может быть выполнен. После обнаружения ошибки и устранении ее программистом, программа должна быть вновь поставлена на компиляцию и решение. Процесс отладки в этом случае затягивается. Однако, программы, созданные трансляторами выполняются значительно быстрее, чем программы, созданные интерпретаторами.

Второе поколение Бейсик – систем (1975-1985 г.) тесно связано с появлением первых персональных ЭВМ, которые обладали сравнительно небольшой оперативной памятью (32 – 64 Кбайт). В середине 70-х годов Билл Гейтс, будущий основатель фирмы Microsoft Corporation, разработал интерпретатор *Basic-80*, который функционировал на наиболее популярных восьмиразрядных процессорах того времени Zilog-80 и Intel-8080. В него уже вошли такие языковые средства, как элементы машинной графики, процедуры организации звуковых эффектов и взаимодействия с активными внешними устройствами, имелись достаточно развитые процедуры обработки текстовой информации и управле-

ния файлами. Наиболее полно эти возможности были представлены в одной из лучших версий интерпретатора второго поколения – GW-Basic.

Среди отечественных разработок того времени следует отметить серию интерпретаторов компилирующего типа Бейсик/F, Бейсик/Fs и Бейсик/Fsc разработанных на Рижском производственном объединении ВЭФ.

Третье поколение языка Бейсик (1985-1990 г.) обязано своим появлением быстрому росту производительности компьютеров, объему оперативной и внешней памяти. К этому периоду относятся разработки языков программирования QBasic фирмы Microsoft Corporation и Turbo-Basic фирмы Borland International. Обе эти системы программирования – системы компилирующего типа.

Учитывая специфику ПК, современные трансляторы дают пользователю, как правило, следующие возможности работы с программами: редактирование программ *(Edit)*, загрузку *(Load)*, запуск *(Run)*, компиляцию (*Compile*), отладку *(Debug)*, сохранение файлов, результатов прогонки, протокола работы *(Save)*.

Все трансляторы содержат редакторы, обеспечивающие быстрый набор программ. Эти редакторы обеспечивают копирование и удаление фрагментов текста, поиск информации по контексту и др. Широкое распространение получили трансляторы *Turbo* для языков Бейсик, Си, Паскаль, Фортран, Пролог, Ассемблер.

Работа с *Turbo*-трансляторами всех языков внешне совершенно одинакова, поскольку пользователю предоставляется стандартный набор функций, отображающихся в главном меню, построенного по принципу "раскрывающегося меню": команды главного меню расположены по горизонтали, при выборе пункта меню под ним открывается окно с меню опций этой команды, при выборе опции может открыться окно с меню опций выбранной опции и так далее.

Разработчики построили *Turbo*-трансляторы по единой идеологии что позволяет:

– подключать компиляторы с любого языка к *Turbo*-оболочке, которая обеспечивает работу с пользователем в диалоговом режиме;

– отлаживать совместно в некоторых случаях программы, написанные на разных языках;

– поддерживать единый стиль диалога с пользователем путем использования стандартного набора функциональных и управляющих клавиш Например Esc - возврат в предыдущее состояние работы транслятора; F1 - вывод подсказки; Enter - выбор режима или вида работы; стрелки ←, →, - управление движением курсора по экрану и внутри "окна"; использование горячих клавиш для выбора пунктов меню.

Turbo-трансляторы имеют мощную систему помощи, которая существенно упрощает освоение программы и работу с ней.

Основное достоинство *Turbo*-оболочек состоит в том, что они позволяют выполнять все функции не выходя из *Turbo*-среды. Кроме того они имеют функцию, позволяющую выйти временно в *DOS* , а затем вернуться к прерванному состоянию.

Четвертое поколение (с 1990 года) языка программирования Basic обусловлено развитием графических операционных систем и принципов объектно-ориентированного программирования. С 1990 выпущено уже несколько версий этого языка программирования, получившего название Visual Basic. Последняя версия языка – Visual Basic 7.0. Она позволяет разрабатывать 32-х разрядные приложения для работы в среде Windows (Windows 95, Windows 98, Windows NT и последующих версиях), а также программные средства для работы в Internet. Версии этого языка программирования применяются в таких приложениях Windows как Word, Excel, Access для разработки пользовательских процедур и функций.

9.2. Начальные сведения о языке программирования QBasic

9.2.1. Синтаксис языка

Алфавит

Алфавит языка Бейсик включает все отображаемые и управляющие символы, представленные в кодировочной таблице, которая в данный момент загружена в ОЗУ или хранится в ПЗУ компьютера.

Кодовую таблицу можно вывести на печать с помощью простой программы:

```
REM - программа вывода символов кодовой таблицы на печать
PRINT TAB(32); "Кодовая таблица"
FOR i = 32 TO 255
    PRINT i; "="; CHR$(i),
NEXT i
```

В языке QBasic, также как и в других программах используются *команды, операторы, функции, константы и переменные*. Команды, операторы и функции являются зарезервированными словами, их нельзя использовать для обозначения меток или переменных.

Команды оперируют целиком программами, чаще всего используются в режиме прямого исполнения, редко включаются в текст программы. В системе QBasic большинство команд закреплено за пунктами меню. Выделив курсором пункт меню и нажав клавишу Enter, мы вводим определенную команду.

Операторы служат для реализации алгоритма программы. Различают декларативные и исполняемые операторы. Декларативные операторы служат указанием транслятору, что надо сделать до начала выполнения программы. В процессе выполнения программы они игнорируются программой. К декларативным операторам относятся операторы объявления типов переменных и структур данных (DIM, REDIM, DEFINT ...), задания способов распределения данных в памяти ЭВМ (COMMON, LOCAL, STATIC,...), формирования внутреннего блока данных (DATA), описание нестандартных функций (SUB, FUNCTION, DEF SEG), меток, комментариев (REM) и др.

Декларативные операторы используются обычно в начале программы. На них распространяется правило: сначала опиши (объяви), а потом используй.

Функции - это подпрограммы, используемые для выполнения типовых вычислительных операций. QBasic имеет большое число встроенных функций для работы с числовыми и строковыми переменными, допускает использование пользовательских внутренних и внешних функций.

Переменные и константы.

Переменные снабжаются уникальными именами. Имя переменной может содержать от одного до 40 символов. В имени переменной допускается использовать буквы латинского алфавита и цифры.

QBasic допускает использование именованных констант. Объявление именованных констант осуществляется с помощью оператора CONST, имеющего следующий формат:

CONST <имя константы> = <выражение> [,<выражение>]

Имена констант формируются по тем же правилам, что и имена переменных.

9.2.2. Типы данных

Все данные могут быть нескольких типов (см. раздел 7.1):целые одинарной длины, целые удвоенной длины, вещественные одинарной точности, вещественные удвоенной точности, строковые. Для обеспечения правильного обмена данными при выполнении программы тип данных должен быть объявлен. По умолчанию все данные считаются вещественными переменными одинарной точности.

Имеется несколько способов указания типов данных: явно с помощью суффиксов – специальных символов, записываемых после константы или после имени переменной, неявно с помощью оператора DEFTYPE и явно с помощью описателей.

Явное объявление типов данных:

Целые одинарной длины – символ %: 124%, A1%

Целые удвоенной длины – символ &: 537&, B7&

Вещественные одинарной точности – символ !: 3.14!, D6!

Вещественные удвоенной точности – символ #: 673,05643#, DD8#

Строковые переменные - символ $: C1$, b2$

Неявное объявление типов данных с помощью оператора DEFTYPE:

DEFINT – объявление переменных целого типа одинарной длины;

DEFLNG - объявление переменных целого типа двойной длины;

DEFSNG - объявление вещественных переменных одинарной точности;

DEFDBL - объявление вещественных переменных удвоенной точности;

DEFSTR - объявление строковых переменных.

Формат оператора объявления типов переменных:

DEFTYPE <список переменных>

Например:

DEFINT I-N – все переменные, имена которых начинаются с символов I, J, K, L, M, N будут считаться переменными целого типа одинарной длины.

DEFSNG A, D, F – все переменные, имена которых начинаются с символов A, D, F, будут считаться вещественными переменными одинарной точности.

DEFDBL X,Y,Z - все переменные, имена которых начинаются с символов X, Y, Z, будут считаться вещественными переменными удвоенной точности.

DEFSTR C - все переменные, имена которых начинаются с символа C, будут считаться строковыми переменными.

Неявное объявление типов переменных с помощью описателей:

- INTEGER - целые одинарной длины;
- LONG - целые удвоенной длины;
- SINGLE - вещественные переменные одинарной точности;
- DOUBLE – вещественные переменные удвоенной точности.

Эти описатели используются в операторах COMMON, DIM, REDIM, SHARED, TYPE.

Например: DIM a, b AS INTEGER – переменные a и b объявляются как переменные целого типа одинарной длины.

9.3. Среда программирования

9.3.1. Загрузка программы

Загрузка *QBasic* осуществляется из соответствующего каталога выбором файла *qbasic.exe* .

Выход из программы осуществляется выбором опции *Quit* в команде *File*.

После запуска программы на экране дисплея появляется рабочий экран (рис. 9.3.1). Если в середине экрана размещается заставка с указанием версии языка программирования, то для продолжения работы следует нажать клавишу *Esc*. Программа переходит в режим редактирования. Для выхода из режима ре-

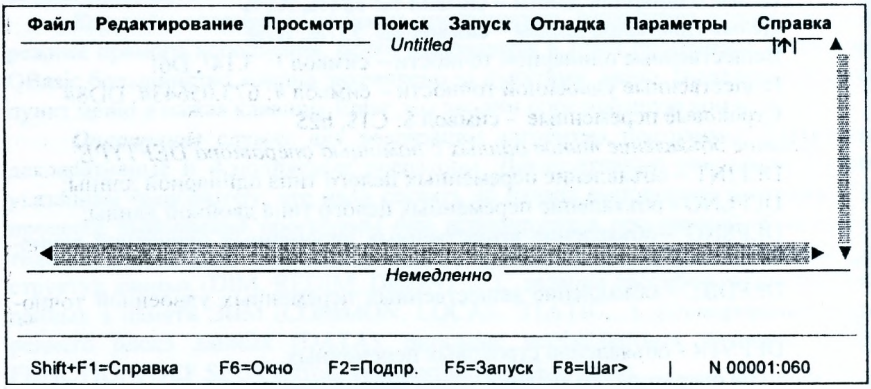


Рис. 9.3.1. Среда программирования языка QBasic

дактирования в меню нажмите клавишу *Alt*.

В верхней части экрана размещаются команды главного меню, а внизу экрана выводится строка состояния, в которой указано назначение функциональных клавиш. Программа имеет два окна: окно программирования (Utlited) и окно непосредственного выполнения *Немедленно*. Эти два окна соответствуют двум режимам работы QBasic: *программному* режиму и режиму *непосредственного выполнения*. В программном режиме текст программы вначале должен быть полностью набран после чего программа ставится на решение командой *Shift -F5* или *F5*. Окно непосредственного выполнения служит для выполнения несложных линейных программ. Программа выполняется сразу же после нажатия клавиши Enter.

Справа и внизу окна программирования расположены полосы прокрутки, которые служат для быстрого просмотра текста программы. Управление осуществляется щелчком мыши по соответствующей стрелке, расположенной в конце полосы прокрутки. Стрелка в верхней части окна программирования служит для развертывания окна пограммы на весь экран или восстановления первоначального состояния.

Выбор команд осуществляется клавишами управления курсором или мышью. Для выполнения команды выделите ее курсором и нажмите *Enter* – открывается меню второго уровня. Для выхода из меню выбранной команды нажмите *Esc*. При работе с мышью команда выполняется при двойном щелчке мышью по соответствующей команде.

Функциональные клавиши, указанные внизу экрана, имеют следующее назначение:

Shift - F1 (Справка) - выводит на экран контекстную подсказку;

F6 (Окно) – переход из окна программирования в окно непосредственного выполнения и обратно:

F2 (Подпрограмма) - открывает окно для редактирования подпрограммы;

F5 (Запуск) - запуск программы на выполнение;

F8 (Шаг) – пошаговое выполнение программы.

9.3.2. Команды главного меню

Файл (File) - работа с файлами, имеет следующие опции:

Новый (New) – создание нового файла. Старые данные теряются.

Открыть (Open) - загрузка файла с диска. После ввода команды на экране появляется окно диалога с шаблоном вызова файлов: **.Bas*. Если нажать клавишу *Enter*, то на экране выводится меню файлов, содержащихся в текущем каталоге (рис. 9.3.2). Выберите нужный файл и нажмите *Enter*. Загружаемый файл появится на экране. Переход внутри окна диалога от одного окна к другому осуществляется клавишей *Tab*. Для выбора нужного каталога перейдите в окно "Каталоги" выделите курсором требуемый диск и нажмите *Enter* – имя выбранного диска и каталога появится в окне диалога. Введите в окне "Имя

файла" имя открываемого файла или выберите его из списка "Файлы" и нажмите Enter;

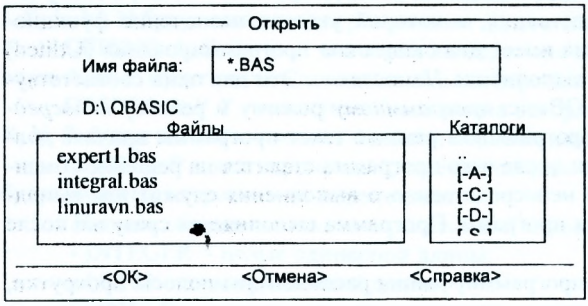


Рис. 9.3.2. Открытие файла

Сохранить (Save)- сохранение текущего файла на диске;

Сохранить как... (Save as ...) – сохранение нового файла или переименование текущего файла при записи его на диск. Принцип работы в окне диалога при сохранении файла на диске аналогичен работе в окне открытия файла;

Печать (Print) – служит для управления выводом программы на печать. Можно напечатать только выделенный текст, текущее окно или всю программу;

Выход - выход из *QBasic*.

Редактирование - редактирование файлов. Процесс редактирования в системе *Turbo* в основном соответствует порядку принятому в текстовых редакторах, то есть команды набираются в строках с помощью клавиатуры. После набора каждой строки следует нажимать клавишу *Enter*. При редактировании текста используются клавиши и их комбинации. Некоторые комбинации клавиш приведены в табл. 9.3.1.

Команды редактирования

Система QBasic снабжена мощной системой помощи. Справку можно получить путем ввода команды *Справка* главного меню. Этот режим позволяет получить справку по содержанию и по ключевым словам, операторам, функциям. При поиске справки по ключевым словам, можно просмотреть примеры скопировать их и включить в текст программы. Можно получить контекстную помощь по ключевым словам: выделить ключевое слово в программе и нажать клавишу F1.

В меню *Редактирование* имеется также две команды для создания новых подпрограмм и функций пользователя. При вводе соответствующих команд выводятся диалоговое окно запроса имени программы (функции), после чего на экран выводятся ключевые слова заголовков подпрограмм, например:

```
SUB factorial
END SUB
```

Просмотр. Данный пункт меню имеет три опции: Subs, Разбить и Экран вывода. Опция *Subs* открывает диалоговое окно для работы со списком модулей, процедур (страниц). Оно позволяет выбрать процедуру или функцию для редактирования или удаления. Первым в списке стоит имя головного модуля –

Utilities. Опция *Разбить* делит окно на две части по горизонтали. Окна равнозначны, и в них отображается текст одной и той же программы. Переход из одного окна в другое осуществляется нажатием клавиши F6. Опция *Экран вывода*

Таблица 9.3.1

Команды редактирования текста

Команды	*Клавиши и их комбинации*
Выделение текста	
Выделение символов или строк	Shift - <стрелки>
Выделение слов	Shift - Ctrl - <стрелки>
Вставка и удаление	
Переключение режима вставки/замены. Режим вставки символов. Символ вставляется в позицию курсора, весь текст, находящийся справа от курсора смещается вправо. В режиме замены вновь вводимый символ стирает символ в текущей позиции.	Ins
Вставить строку сверху/снизу	Home Ctrl-N / End - Enter
Удалить строку с сохранением в буфере	Ctrl - Y
Удалить текст от текущей позиции до конца строки с сохранением в буфере	Ctrl - Q Y
Удалить выделенный текст	Del
Удалить выделенный текст с сохранением в буфере	Shift - Del
Удалить правое слово	Ctrl - T
Копирование текста	
Копировать выделенный текст	Ctrl - Ins
Поиск	
Выделенного текста	Ctrl - \
Повторный поиск	F3
Отладка программы	
Просмотр результатов выполнения программы (видеопамяти)	F4
Продолжение выполнения программы	F5
Выполнения программы от текущего положения курсора	F7
Помощь	
Очистка экрана помощи	F5
Справка о системе помощи	Shift – F1
Выход в меню помощи	Alt - H

(F4) выводит на экран содержание видеобуфера на момент завершения или прерывания программы. Обратный переход осуществляется повторным нажа-

тием клавиши F4.

Поиск - позволяет найти нужный фрагмент текста программы, а при необходимости и заменить. Команда выводит на экран соответствующее окно диалога.

Запуск. Команда имеет три режима запуска: *выполнение программы полностью* - команда *(Shift - F5)*; *перезапуск* - прерывание программы и выполнение ее с самого начала; *продолжить* – продолжение программы от точки останова *(F5)*.

Отладка. Эта команда позволяет устанавливать пошаговый режим работы – опции *Шаг* и *Процедура на шаг,* контрольные точки – опция *Контрольная точка*, трассировку программы – опция *Трассировка*.

Опция *Шаг* устанавливает пошаговый режим работы: программа выполняет один оператор или вычисляет одно выражение и останавливается. Для продолжения работы следует нажать клавишу *F8*. Режим *Процедура на шаг* отличается от предыдущей тем, что процедуры и функции рассматриваются программой как одно целое, что ускоряет отладку. Для продолжения работы после остановки программы используется клавиша *F10*.

В режиме *Трассировка* каждая выполняемая строка выделяется, что позволяет проследить за процессом выполнения программы.

Контрольная точка (маркер), устанавливается на строке или выражении, которое необходимо проконтролировать. При достижении маркера программа приостанавливает работу. Установка контрольной точки производится через меню или клавишей *F9*. Отмена контрольных точек осуществляется соответствующей командой меню *Отладка*.

Параметры. Эта команда позволяет настроить цветовое оформление рабочей среды (дело бессмысленное), установить маршрут для подключения файла справки и включить или отключить режим проверки синтаксиса программы. Если режим проверки синтаксиса включен, то после нажатия клавиши Enter в конце строки текста программы QBasic проверяет синтаксис операторов, а также переводит все ключевые слова (команды, операторы, встроенные функции) в верхний регистр. Это позволяет пользователю обнаружить самому ошибки в тексте программы, если ключевое слово написано неправильно.

9.4. Программирование в среде QBasic

9.4.1. Принципы построения Бейсик программ

Программа набирается в окне программы построчно. Длина программной строки не должна превышать 255 символов. Автоматического перехода на другую строку не предусмотрено. Поэтому если длина строки превышает 80 символов, текст программы сдвигается влево и исчезает с экрана. Это затрудняет проверку текста программы и его отладку. Допускается переносить длинные строки. Для этого в конце первой части строки вставляется пробел и символ подчеркивания "_". Разрешается размещать в одной строке несколько операторов разделяя их двоеточием. Строка завершается символом возврата каретки, который автоматически вводится при нажатии клавиши Enter.

QBasic не требует обязательной нумерации строк.

Для перехода на другую строку необходимо нажать клавишу Enter в ре-

жиме вставки символов. В режиме замещения символов курсор возвращается в начало строки.

Следует избегать включения в одну строку исполняемых и декларативных операторов.

Для изменения процесса вычислений в программе используются *метки*. Имя метки формируется по тем же правилам, что и имя переменной. Имя метки заканчивается двоеточием, например: M1:, V1:, Pause1: и т. д. В одной строке с меткой не допускается записывать исполняемые операторы. Допускается включать неисполняемые операторы и комментарии

Чтобы сделать программу более читабельной, в нее рекомендуется включать комментарии. Правее комментария нельзя включать никакие операторы, но можно вставлять комментарий после операторов. Комментарии вводятся оператором REM или апострофом – символ (’). В строке комментария разрешается использовать любые символы:

Пример программы.

```
REM процедура вычисления суммы конечного ряда
' табулирование функции одной переменной
    REM линейная программа
    ' Поиск минимума функции одной переменной
    Input "Укажите границы A и B интервал табулирования функции ", A, B
    dX= 0.1          'шаг табуляции
    Max=-1E38        ' начальное присвоение
    FOR x=A  TO B STEP dX   'заголовок цикла
            y= EXP(x)*SIN(x^2)
            IF y>Max THEN Max=y
    NEXT x
    PRINT "Max=";Max
    END
```

Порядок работы.

1. Загрузите программу: войдите в каталог QBasic и запустите файл *qbasic.exe*.
2. После загрузки программы нажмите *Esc*. Заставка с указанием версии программы убирается и программа готова к набору текста.
3. Сохраните сразу программу на диске, например R:

- нажмите клавишу Alt для выхода в меню;
- введите команды Файл, Сохранить как;
- в окне диалога перейдите клавишей Tab в окно Каталоги и выберите диск R:, введите имя файла в окне Имя файла;
- выделите команду OK и нажмите Enter.

Программа возвращается в окно редактирования.

4. Если будет использоваться уже существующий файл, выберите опцию Открыть и загрузите нужный файл (рис. 9.3.2). Загружаемый файл выводится на экран.
5. Отредактируйте загруженную программу или напишите новую.
6. Сохраните программу на диске командой *Файл, Сохранить*.
7. Поставьте программу на решение командой *Shift - F5*.
8. Проанализируйте результаты решения.
9. Выйдите из программы.

9.4.2. Линейные программы

Линейные программы представляют собой, как уже отмечалось, последовательность операторов, циклов и условных переходов нет. Структурная схема такой программы включает блок ввода данных, блок вычислений и блок вывода результатов.

Операторы ввода данных

Для ввода данных используются операторы LET, DATA, READ, RESTORE, INPUT, LINE INPUT, функция INPUT$.

Оператор ***LET***. Служит для присвоения значений переменным. Формат оператора: LET <имя переменной> = <выражение>

Оператор LET может быть опущен, поэтому выражения

LET A=exp(x)+ sin(x) и A=exp(x)+ sin(x)

эквивалентны.

Операторы ***DATA*** и ***READ*** позволяют вводить данные из программы. Оператор *DATA* заносит данные в специальную область оперативной памяти компьютера, а оператор *READ* считывает эти данные из оперативной памяти и присваивает их переменным. Эта область данных содержит специальный указатель. При запуске программы указатель устанавливается в начале области данных. При считывании данных из области памяти он также перемещается. Если число переменных в операторе *READ* больше числа значений в операторе *DATA*, то указатель выходит за область данных и будет выдано сообщение об ошибке.

Типы данных в операторах *DATA* и *READ* должны соответствовать друг другу. Эти операторы могут располагаться в любом месте программы. Строковые переменные записываются без кавычек, если в них нет пробелов, и заключаются в кавычки, если содержат пробелы.

Форматы операторов:

DATA <список данных>

READ <список переменных>

Пример 9.4.1. Использование операторов DATA, READ

```
DATA 125, 34.78, 1.24E-5, БРЕСТ, "МИНСК - СТОЛИЦА"
READ A,B,D,C$,C1$
```

Переменным A, B и D будут присвоены, соответственно, числовые значения 125, 34.78 и 0.0000124, переменным C$ и C1$ - "БРЕСТ" и "МИНСК - СТОЛИЦА".

Для повторного использования данных используется оператор ***RESTORE***. Формат оператора: RESTORE [<метка>].

Если метка в операторе RESTORE не указана, то указатель устанавливается в начало области данных, иначе указатель устанавливается на соответствующую метку в области данных.

Пример 9.4.2. Использование операторов DATA, READ, RESTORE

```
10 REM ПРИМЕР ИСПОЛЬЗОВАНИЯ ОПЕРАТОРОВ DATA, READ, RESTORE
20 DATA 136.75, 18E5,123.45,1978,.9875
25 DATA .5439,1.567,4.65,12.23,48.56
30 READ A1,A2,A3,A4,A5 : REM чтение данных из строки 20
...150 READ B1,B2 : REM чтение данных из строки 25
```

```
...250 RESTORE 25 : REM перевод указателя данных на метку 25
260 INPUT "Введите размерность массива N, не более 5",N
260 DIM D2(N)
270 FOR i=1 TO N
280 READ D2(i) : REM чтение данных из строки 25 в одномерный массив
290 NEXT i
```

Оператор ***INPUT*** служит для ввода данных с клавиатуры в режиме диалога с пользователем. Формат оператора:

INPUT [;] "текстовое выражение"[;/,]<список переменных>

Здесь [;] – необязательный параметр, оставляет курсор в текущей строке; [;/,] - означает, что в качестве разделителя может использоваться один из указанных знаков ";" или ",".

При выполнении оператора *INPUT* на экран выдается запрос на ввод данных. Если в качестве разделителя используется ";", то запрос сопровождается выводом на экран вопросительного знака, при использовании в качестве разделителя "," вопросительный знак на экран не выводится. Текстовое сообщение позволяет сделать запрос понятным пользователю. В списке переменных данные разделяются запятыми или пробелами (предпочтительнее разделять данные запятыми). С помощью оператора INPUT можно вводить как числовые, так и символьные переменные. Если символьная переменная не содержит пробелов и других разделителей, то при вводе данных заключать ее в кавычки не обязательно.

Не используйте оператор INPUT без текстового комментария.

Пример 9.4.3. Использование оператора Input

```
INPUT "Введите переменные A и B"; A,B
INPUT "Введите Ваш год рождения", GR$
```

Здесь A и B - числовые переменные, GR$ - символьная переменная.

Оператор ***LINE INPUT*** служит для ввода одной символьной переменной. При вводе значение символьной переменной в кавычки не заключается, в ней допускается наличие пробелов и других разделителей. Конец строки определяется символом возврата каретки - нажатие клавиши ENTER. Формат оператора LINE INPUT аналогичен формату оператора INPUT.

Функция ***INPUT$*** служит для ввода символов, не отображаемых на экране, например пароля. Формат функции: c1$ = INPUT$(n [,[#]nf])

Здесь n - число вводимых символов, # - необязательный параметр, признак номера канала, nf – номер канала. Эти параметры используются при вводе данных из файла.

Пример 9.4.4. Использование оператора LINE INPUT и функции INPUT$

```
LINE INPUT "Введите фамилию, имя и отчество ",FIO$
B$=INPUT$(5)
```

Символьной переменной B$ присваивается код из пяти символов, при выдаче запроса вводимые символы отображаться на экране не будут. Когда в программе встречается оператор INPUT$(n), программа останавливается и ожидает ввода данных. Поэтому данный оператор в формате B$= Input$(1) может быть использован для остановки программы до нажатия любой клавиши.

Функция Input$(1) может быть использована для остановки программы до нажатия любой клавиши.

Операторы вывода данных

Для вывода данных используются операторы PRINT и LPRINT, PRINT USING, LPRINT USING.

Оператор PRINT выводит данные на экран монитора, а LPRINT – на принтер. Оба оператора имеют одинаковый формат. Операторы PRINT USING и LPRINT USING имеют то же назначение, но дополнительно позволяют форматировать выводимые данные.

Формат оператора PRINT:

PRINT "Комментарий" [;/ /,] <список выражений> [;/,]

Комментарий служит для пояснения выводимой информации; символы [;/ /,] – разделители. Если в качестве разделителя используется символ [;] или пробел, то выводимый текст размещается один возле другого без пробелов, при выводе чисел одна позиция резервируется для знака числа. Если в качестве разделителя используется запятая, то каждое новое сообщение печатается в новой зоне. Оператор PRINT формирует выводную строку, которая разбита на 5 зон по 14 символов. Это позволяет размещать выводимую информацию по колонкам. Если текст не умещается в зоне, то он занимает соседнюю зону, а новая порция информации начинает печататься в соседней свободной зоне. В список выражений может помещаться любая информация, в том числе и алгебраические выражения. Выводимые выражения в списке разделяются запятыми. В конце строки могут быть расположены управляющие символы: точка с запятой или запятая. При наличии этих управляющих символов курсор остается в текущей позиции, и следующий оператор PRINT будет выводить информацию с этой позиции. Действуют эти управляющие символы так же как и управляющие символы после комментария.

Пример 9.4.5. Использование оператора PRINT

```
PRINT "X=";x  ' выводится значение x
PRINT "X=";x,"Y=";y ' выводятся значения  x и y
PRINT "X="x,"Y=";SIN(x) ' выводится значение x, вычисляется и
                        ' выводится значение функции SIN(x)
```

Для управления выводом информации в список выражений оператора PRINT могут включаться функции TAB и SPC.

Функции TAB и SPC перемещают точку вывода на заданное число позиций. Отличие в использовании функций TAB и SPC состоит в том, что функция TAB перемещает точку ввода относительно края экрана, а функция SPC – относительно текущей позиции (рис. 9.4.1)

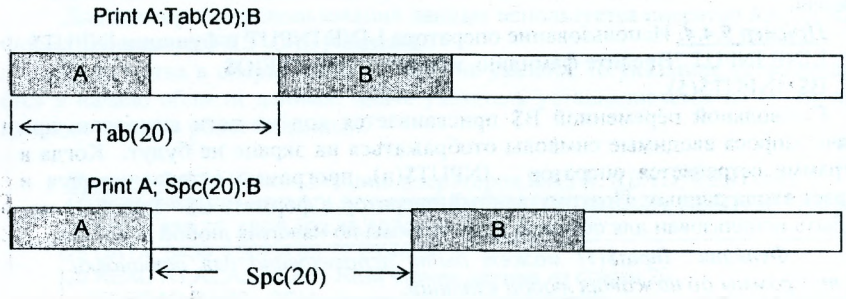


Рис. 9.4.1. Использование функций Tab и Spc

Оператор *PRINT USING* предназначен для вывода информации на экран с форматированием.

Формат оператора PRINT USING:

PRINT USING {C$|e$}; <список переменных>

Здесь C$ и e$ - шаблоны для вывода символьной или числовой информации. Шаблоны для вывода символьной информации будут рассмотрены в разделе 9.4.9.

Шаблоны для вывода числовой информации (табл. 9.4.1):

Если число знаков, указанных в целой части числа шаблона меньше, чем фактическое, то число выводится на экран полностью, но перед первым символом числа ставится знак %.

В примерах 4 и 7 числа .01236787# и 1236787.45673# - вводятся как константы удвоенной точности

Операторы *LPRINT* и *LPRINT USING* имеют тот же синтаксис, что и операторы *PRINT* и *PRINT USING,* но выводят информацию на печатающее устройство.

Таблица 9.4.1

Шаблоны для вывода числовой информации

Символ	*Описание*	*Пример*
#	- резервирует знакоместо для числа, пробела, знака $ или *	PRINT USING "#####"; a, b, c – зарезервировано 5 знакомест при выводе числа на экран монитора
.	– позиция десятичной точки	PRINT USING "##.###";a, в, c – числа a, b, c будут выведены на экран с тремя знаками после десятичной точки
^^^^	- местоположение порядка числа (E+pp) (E -pp)	PRINT USING "#.####E+pp"; a, b, c – числа будут выведены в экспоненциальной форме
±	- выводит знак числа. Может располагаться в начале или в конце шаблона	PRINT USING "-##.###"; a, b , c – перед отрицательными числами будет выводиться знак числа
**	- заполнение лидирующих пробелов символами "*"	PRINT USING "**###.##"; a, b, c – если число знаков в числе меньше, чем число знакомест, указанных в шаблоне, то лидирующие пробелы заполняются звездочками
$	- вывод знака $ перед старшей значащей цифрой	PRINT USING "$###.##"; a, b, c - перед старшей значащей цифрой будет выведен знак $
,	- разделитель разрядов числа	PRINT USING "###,###,###.##";a, b, c – целая часть числа разделяется на триады от младшего разряда к старшему, а дробная часть на триады не разделяется

Примеры:

№	*Строка программы*	*Результат*
1	PRINT USING "#####.##"; 123.6787	123.68
2	PRINT USING "#####.##"; -123.6787	-123.68
3	PRINT USING "+#.##"; 123.6787	%+123.68
4	PRINT USING "#.##^^^^"; .01236787#	0.12D-01
5	PRINT USING "**#####.##"; 123.6787	****123.68
6	PRINT USING "$####.##"; 123.6787	$123.68
7	PRINT USING "###,###,###.#######"; 1236787.45673#	1,236,787.4567300

Пример линейной программы.

В качестве примера линейной программы рассмотрим вычисление производной от функции в заданной точке.

Производная от функции может быть вычислена численными методами. Известно, что:

$$f'(x) = \lim_{x \to \pm 0} \frac{(f(x+\Delta x) - f(x-\Delta x)}{2\Delta x} + 0(x) \qquad (9.4.1)$$

отсюда вытекает способ численного дифференцирования. Если заменить предел Δx его конечным значением h, то получим приближенные формулы для вычисления первой и второй производных:

$$f'(x) \approx \frac{f(x+h) - (f(x-h)}{2h} \qquad (9.4.2)$$

$$f''(x) \approx \frac{f(x+h) + f(x-h) - 2f(x)}{h^2} \qquad (9.4.3)$$

Эти выражения представляют собой усеченные интерполяционные многочлены (многочлен Стирлинга). Одной из серьезных проблем в данном случае является выбор величины шага h. *При уменьшении шага уменьшается ошибка усечения, но возрастает ошибка округления при вычислении производной.* Поэтому стремятся выбрать оптимальную величину шага, при которой ошибка усечения и ошибка округления будут примерно равны. Для формулы (9.4.2) оптимальный шаг определяется из выражения:

$$h = \sqrt[3]{\frac{3\varepsilon}{M_3}} \qquad \text{или} \qquad \frac{1}{6h}\left|\Delta^3 y\right| \approx \frac{1}{2}\frac{\varepsilon}{h}, \qquad (9.4.4)$$

где h - шаг, $\Delta^3 y$ - конечная разность 3-го порядка, ε - абсолютная погрешность вычисления функции, M_3 - максимальное значение конечной разности 3-го порядка. Таким образом, ошибка усечения равна примерно половине ошибки округления. Полная погрешность не превзойдет при этом 0,5 ε/h.

Пример 9.4.6. ЭВМ выводит результат с 8 знаками после запятой, при этом семь знаков точные. Требуется определить значение шага при вычислении производной первого порядка, чтобы ошибка усечения не превышала 0,0001.

Решение. Абсолютная погрешность вычисления функции равна $0{,}5*10^{-7}$. из (9.4.4) получаем

$$h = \frac{\varepsilon}{2\Delta} = \frac{0{,}00000005}{2*0{,}0001} = 0{,}00025.$$

При ϵ=0.001 и Δ=0.001 величина шага будет равна 0,5.

Для выражения (9.4.3) величина шага определяется из следующего соотношения

$$\frac{1}{12h^2}\left|\Delta^4 y\right| \approx 4\frac{\varepsilon}{h^2}. \qquad (9.4.5)$$

В условиях примера 9.4.6 величина шага будет равна ~ 0,05.

Таким образом, для получения приемлемого значения результата не следует стремиться сильно уменьшать шаг приращения аргумента, а также следует учитывать точность, с которой вычисляются значения аргумента и функции. Напомним, *абсолютная*

погрешность суммы не превышает суммы погрешностей слагаемых, абсолютная погрешность произведения (частного от деления двух чисел) не превышает наибольшей из абсолютных погрешности сомножителей.

9.4.3. Разветвляющиеся программы

Разветвляющиеся программы организуются с помощью условного оператора IF. Различают однострочные и многострочные условные операторы.

Форматы однострочного оператора IF:

а) простой однострочный оператор

IF <условие> THEN <операторы>

При выполнении оператора IF проверяется условие и, если оно истинно, то выполняется действие, указанное после оператора THEN. Если выражение ложно, то управление передается на оператор, следующий за оператором IF.

```
IF x<e THEN GOTO m1
IF x<a THEN y=SIN(x)*EXP(2^LOG(x)): GOTO m2
```

б) простой расширенный оператор IF

IF <условие> THEN <операторы 1 > ELSE <операторы 2>

При выполнении оператора IF, если условие истинно, выполняются операторы, указанные после оператора THEN, в ином случае выполняются операторы, следующие за оператором ELSE. После выполнения соответствующей группы операторов управление передается на оператор, следующий за оператором IF.

```
IF x<a THEN y=SIN(x)*EXP(2^LOG(x)) ELSE y=3*x^2-5*x+4
```

После операторов THEN и ELSE может быть указано несколько операторов, разделенных двоеточием. Однако, число операторов ограничено длиной строки. Этих недостатков лишен многострочный оператор IF.

Пример 9.4.7. Вычислить выражение

$$y = \begin{cases} x^2 - 5, & \text{если } x<a \\ x, & \text{если } a<= x <=b \\ -x, & \text{если } x>b \end{cases}$$

Решение.

```
INPUT "Введите значение X", X
INPUT "Введите значения a и  b ",a,b
  IF  x<a THEN y = x^2 –5: GOTO m1
  IF  x>=a AND x<=b THEN y=x: GOTO m1
  y = -x
  m1:
  PRINT "x="; x, "y="; y
END
```

При использовании многострочного IF алгоритм для решения примера 9.4.7 будет выглядеть следующим образом:

```
INPUT "Введите значение X", X
INPUT "Введите значения a и  b ",a,b
   IF  x<a THEN
   y = x^2 –5
    ELSEIF  x>=a AND x<=b THEN
       y=x
    ELSE
       y = -x
END IF
   PRINT "x="; x, "y="; y
END
```

Форматы многострочного оператора IF:

а) простой	б) расширенный

IF <условие> THEN
  <первая группа операторов>
ELSE
  <вторая группа операторов>
END IF
	IF <условие> THEN
	  <первая группа операторов>
	ELSEIF <условие> THEN
	  <вторая группа операторов>
	ELSE
	  <третья группа операторов>
	END IF

При записи операторов следует обращать внимание на структуру записи. Структура должна соответствовать той, что указана в примере.

Достоинство многострочного IF состоит в том, что число операторов в группах не ограничено.

9.4.4. Циклические программы

Циклом называется процедура, в которой вычислительные операции выполняются многократно заданное число раз или до достижения некоторой переменной, вычисляемой в теле цикла, определенного, наперед заданного значения.

Понятие о циклах и их классификации приведено в разделе 8.2.

Циклы могут быть организованы с использованием оператора IF или с использованием специальных операторов: FOR/NEXT, WHILE/WEND, DO/LOOP. Примеры организации циклов приведены в табл. 9.4.2.

Оператор FOR/NEXT

Оператор FOR служит для организации циклов с заданным числом повторений. Цикл относится к циклам с постусловием. Формат оператора:

```
FOR i=Iнач TO iкон STEP di
   <тело цикла>
NEXT i
```

В данном формате Iнач – начальное значение переменной цикла, Iкон – конечное значение переменной цикла, a di – шаг приращения значения переменной цикла.

Операторы FOR и NEXT образуют операторные скобки, между которыми заключено тело цикла.

Пример 9.4.8. Напечатать таблицу перевода температуры из градусов по шкале Цельсия (С) в градусы по шкале Фаренгейта (F) для значений температуры от 15°С до 30°С с шагом 1°С. Формула перевода: F=1,8C+32.

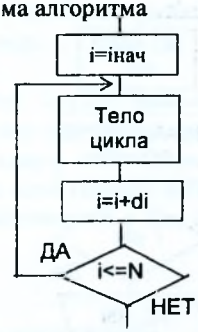
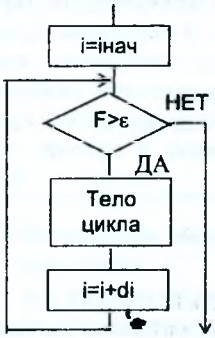
Решение. Число шагов в задаче известно, поэтому для решения задачи применим оператор FOR/NEXT. Переменная цикла целочисленная, изменяется с шагом 1.

```
INPUT "Введите начальное значение температуры ° C: ", tN
INPUT "Введите Конечное значение температуры ° C: ", tK
INPUT "Укажите шаг изменения  температуры ° C: ", dT
PRINT Tab(10);"T° C"; Tab(20);"T° F"
FOR i = tN TO tK STEP dT
```

```
        F= 1.8*i+32
        PRINT Tab(10);i ; Tab(20); y
    NEXT i
    END
```

Таблица 9.4.2

Операторы циклов

а) цикл с постусловием REM использование для организации цикла REM оператора IF i=iнач: REM заголовок цикла M1: <тело цикла, вычисление функции F(i)> i=i+di IF i<=iкон THEN GOTO M1: REM конец цикла PRINT "Результат"; F --- REM использование оператора FOR/NEXT FOR i=iнач TO N STEP di <тело цикла, вычисление функции F(i)> NEXT i PRINT "Результат";F	Схема алгоритма 
б) цикл с предусловием REM Использование оператора IF F=<выражение> i=iнач M1: IF F<=ε THEN GOTO M2 <тело цикла, вычисление F(i)> i=i+di GOTO M1 M2: PRINT "Результат"; F --- REM Использование оператора REM WHILE/WEND F=<выражение> i=iнач WHILE F>ε <тело цикла, вычисление F(i)> i=i+di WEND PRINT "Результат";F	Схема алгоритма 

Пример 9.4.9. Вычислить многочлен $P(x) = 5x^3 + 2x^2 - 15x - 6$ при $x = 2{,}5$

Решение. Задачу можно решить, что называется, "в лоб" – записать выражение по правилам языка Бейсика – и дело сделано. Однако, для повышения скорости вычисления целесообразно заменить операции умножения операция-

ми сложения и вычитания. Особенно это актуально в управляющих ЭВМ, работающих в реальном масштабе времени.

Для вычисления многочлена используется *схема Горнера* (рис.9.4.2.).

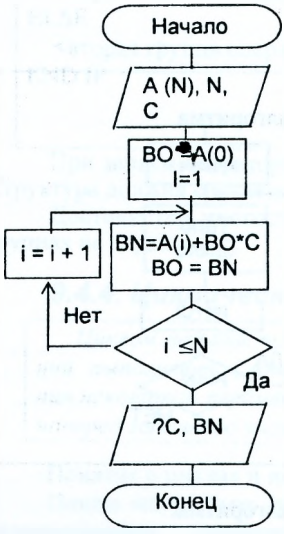


Рис. 9.4.2. Схема алгоритма к примеру 9.4.9.

Пусть задан многочлен:

$$P(x) = a_0 x^n + a_1 x^{n-1} + \ldots + a_{n-1} x + a_n \qquad (9.4.6)$$

Требуется вычислить значение многочлена при x = c, т.е.

$$P(c) = a_0 c^n + a_1 c^{n-1} + \ldots + a_{n-1} c + a_n$$

Запишем эту формулу в виде:

$$P(c) = (\ldots((a_0 c + a_1) c + a_2) c + a_3) c + \ldots + a_{n-1}) c + a_n$$

Введем вспомогательную переменную b:

$b_0 = a_0$

$b_1 = a_1 + b_0 c$

$b_2 = a_2 + b_1 c$

...

$b_n = a_n + b_{n-1} c$.

Отсюда $b_n = P(c)$.

Таким образом, вычисление значения многочлена $P(x)$ при $x = c$ сводится к повторению элементарных операций:

$b_i = a_i + b_{i-1} \cdot c$, $(i = 1,2, \ldots, n)$, где $b_0 = a_0$.

Вычисление многочлена $P(x)$ при $x = c$ обычно осуществляется по схеме, известной как схема Горнера.

В общем виде:

	a_0	a_1	a_2	...	a_n
	+				
c		$b_0 c$	$b_1 c$	...	$b_{n-1} c$
	b_0	b_1	b_2	...	$b_n = P(c)$

В числовом выражении:

	5	2	-15	-6
	+			
2,5		12,5	36, 25	53, 125
	5	14,5	21,25	47,125 = P(2,5)

То есть P(2.5) = 47,125

Схема алгоритма вычисления многочлена по схеме Горнера приведена на рис. 9.4.2, где

A(N) – вектор коэффициентов многочлена, N – степень многочлена, C – значение аргумента, BO – коэффициент b_i, BN – текущее значение многочлена.

Оператор WHILE/WEND

Оператор WHILE служит для организации циклов с параметром (бесконечных циклов), относится к циклам с предусловием. Условием

окончания цикла является достижение функцией, вычисляемой в теле цикла, некоторого наперед заданного значения. Формат оператора:

WHILE <условие>
<тело цикла>
WEND

Тело цикла выполняется в том случае, когда условие истинно. Если условие ложно, то программа выходит из цикла. Особенностью использования данного цикла является то, что значение вычисляемой функции должно быть известно перед входом в цикл. Если начальное значение функции меньше заданной величинt ε, то программа не войдет в цикл.

Оператор DO - универсальный оператор, служит для организации циклов типа "ПОКА". Он может быть организован как цикл с предусловием, так и как цикл с постусловием. Когда в теле цикла используется оператор WHILE, то тело цикла выполняется в том случае, если условие истинно. Когда используется оператор UNTIL, то тело цикла выполняется в том случае, если условие ложно.

Оператор DO

а) цикл с предусловием

DO <WHILE|UNTIL>
<тело цикла>
LOOP

б) цикл с постусловием

DO
<тело цикла>
LOOP <WHILE|UNTIL>

Пример 9.4.10. Вычислить площадь фигуры, ограниченной прямыми Xn=a, Xk =b, y=0 и графиком функции $y=e^x+5Sin(x)$, с точностью *E*.

Решение. Площадь фигуры может быть вычислена с помощью определенного интеграла. Для вычисления определенного интеграла численными методами необходимо организовать цикл типа "ДО", а чтобы вычислить интеграл с заданной точностью применяется метод двойного прохода и реализуется эта процедура с помощью цикла "Пока".

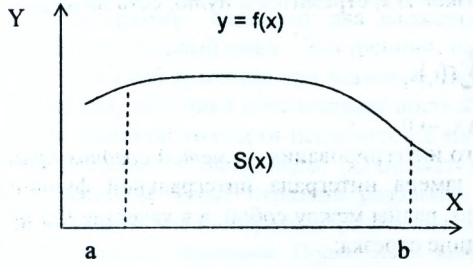


Рис.9.4.3. К определению определенного интеграла

Вычисление определенного интеграла

Пусть функция *y =f(x)* не отрицательна и непрерывна на отрезке *[a; b]*. Тогда площадь *S(x)* криволинейной трапеции представляет собой функцию от *x* (рис. 9.4.3).

Производная от *S(x)* равна *f(x)* всюду на отрезке *[a; b]*. Иными словами,

S(x) оказывается *первообразной* для *f(x)*.

Процедура нахождения первообразной *S(x)* от функции *f(x)* называется интегрированием.

Приращение любой первообразной для функции *f(x)*, получаемое при переходе от a к *b*, называется *определенным интегралом* функции *f(x)*, взятым в пределах от *a* к *b*, и обозначается символом $\int_a^b f(x)dx$. Здесь *a* и *b* называют соответственно нижним и верхним пределами интегрирования.

С геометрической точки зрения определенный интеграл представляет собой площадь криволинейной трапеции, ограниченной графиком функции *f(x)*, прямыми $x = a$, $x = b$ и осью *x* (рис. 9.4.3):

$$S = \int_a^b f(x)dx. \qquad (9.4.7)$$

По определению:

$$S = \int_a^b f(x)dx = F(b) - F(a),$$

где *F(b)* и *F(a)* — первообразные для функции *f(x)* при $x = b$ и $x = a$ соответственно. Для большинства функций первообразную *f(x)* не удается выразить через элементарные функции, поэтому чаще интеграл вычисляют *численными методами*.

Сущность метода численного интегрирования состоит в том, что отрезок интегрирования *[a; b]* разбивают на *n* элементарных отрезков $[x_{i-1}, x_i]$ и заменяют интеграл функции *f(x)* суммой прямоугольников $S_i = f(t_i)\,x_i$ (рис. 9.4.4), где $f(t_i)$ — значение функции в некоторой точке (t_i) внутри *i-го* отрезка; Δx_i — длина отрезка, $\Delta x_i = x_i - x_{i-1}$, то есть

$$S_n = S_1 + S_2 + \ldots + S_{n-1} + S_n + R = \sum_{i=1}^{n} f(t_i)\Delta x_i + R \qquad (9.4.8)$$

Здесь S_n - интегральная сумма, R - погрешность усечения. Предел этой суммы при неограниченном увеличении числа отрезков разбиения, когда длина наибольшего из элементарных отрезков Δx_i стремится к нулю, есть интеграл от функции *f(x)* на отрезке *[a; b]*:

$$\int_a^b f(x)dx = \lim_{\max \Delta x_i \to 0} \sum_{i=1}^{n} f(t_i)x_i$$

Простейший метод численного интегрирования — *метод средних прямоугольников*. В нем используется замена интеграла интегральной функцией (9.4.4). В этом случае все отрезки Δx_i равны между собой, а в качестве $f(t_i)$ принимается значение функции в середине отрезка:

$$\int_a^b f(x)dx = \sum_{i=1}^{n} h\, f(x_{i-1/2}) = \sum_{i=1}^{n} hf(a + hi - h/2) = \sum_{i=1}^{n} h_i f(a + (2i-1)h/2),$$

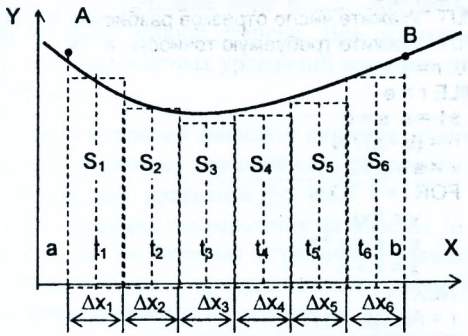


Рис. 9.4.4. Вычисление определенного интеграла методом средних прямоугольников

где h — длина элементарного отрезка, $h = (b-a) / n$; $x_{i-1/2}$ — значение аргумента в середине i-го отрезка, $x_{i-1/2} = a + (2i - 1)(b - a)/(2n)$ Схема алгоритма приведена на рис. 9.4.5.

В других методах вычисляют не площадь прямоугольника, а площадь криволинейной трапеции, заменяя функцию $f(x)$ на элементарном отрезке интерполирующей функцией. Если в качестве интерполирующей функции используется линейная функция, то получаем *метод трапеций:*

$$\int_a^b f(x)dx = \frac{1}{2}\sum_{i=1}^{n} h_i\left(f(x_i) + f(x_{i-1})\right) + R \qquad (9.4.9)$$

В зависимости от используемой интерполирующей функции различают методы Ньютона, Симпсона, Ромберга. В методе Симпсона, например, в качестве интерполирующей функции используется многочлен второй степени $ax^2 + bx + c$. Эти методы обеспечивают получение более точных результатов при меньшем числе шагов.

Формула трапеций (9.4.9) после преобразования принимает вид:

$$S = h\left(\frac{y_0 + y_n}{2}\right) + \sum_{i}^{n-1} y_i\,, \qquad (9.4.10)$$

а формула Симпсона записывается следующим образом:

$$S = \frac{h}{3}(y_0 + y_{2m} + 2(y_2 + y_4 + \ldots + y_{2m-2}) + 4(y_1 + y_3 + \ldots + y_{2m-1})) \qquad (9.4.11)$$

Программа вычисления определенного интеграла методом средних прямоугольников с заданной точностью

Программа содержит два вложенных цикла: цикл For/Next и цикл While/wend. Первый цикл – внутренний, он обеспечивает вычисление площади криволинейной трапеции при заданном числе отрезков разбиения. Второй цикл – внешний, этот цикл обеспечивает достижение заданной точности. Для достижения заданной точности используется метод двойного прохода. Суть этого метода состоит в следующем. Вычисляется площадь криволинейной трапеции при заданном числе отрезков разбиения и результат запоминается. Затем увеличивают число отрезков разбиения вдвое и снова вычисляют площадь криволинейной трапеции. После этого вычисляют ошибку интегрирования по приближенной формуле. Процесс вычисления заканчивается, когда ошибка интегрирования станет меньше или равной некоторой наперед заданной величине.

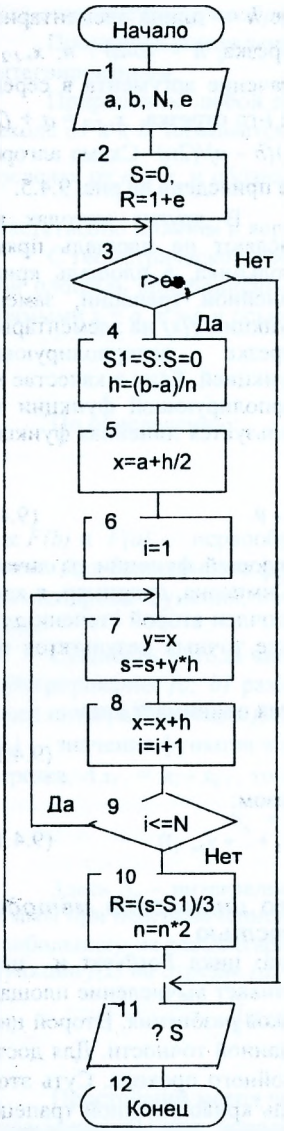


Рис. 9.4.5. Схема алгоритма к задаче 9.4.4

```
INPUT "Введите границы отрезка интегрирования
_ a и b "; a, b
INPUT "Укажите число отрезков разбиения N ", N
INPUT "Укажите требуемую точность, e ", e
s = 0: r = 1 + e
WHILE r > e
    s1 = s: s = 0
    h = (b - a) / N
    x = a + h / 2
    FOR i = 1 TO N
        y = f(x)
        s = s + y * h
        x = x + h
    NEXT i
    r = ABS(s - s1) / 3
    N = N * 2
WEND
PRINT "S="; s
END
```

В программе обозначено: S – площадь фигуры на текущем шаге, S1 - площадь фигуры на предыдущем шаге, R – погрешность усечения.

Оценка погрешности усечения осуществляется по сложным формулам, однако для практических целей можно воспользоваться приближенными формулами (принцип Рунге):

$$\Delta \approx \frac{1}{3} | I_n - I_{2n} |$$ - для формулы трапеции

(9.4.12)

$$\Delta \approx \frac{1}{15} | I_n - I_{2n} |$$ - для формулы Симпсона.

(9.4.13)

Здесь I_n – значение интеграла при разбиении отрезка интегрирования на n частей, а I_{2n} - значение интеграла при разбиении отрезка на 2n частей.

Решение алгебраических и трансцендентных уравнений

Нелинейные уравнения вида $F(x) = 0$ принято называть *алгебраическими*, если они содержат только алгебраические функции, и трансцендентными, если они содержат другие функции (тригонометрические, показательные, логарифмические и т.д.).

При решении нелинейных уравнений всегда возникают серьезные трудности. В аналитическом виде можно получить решение алгебраического уравнения не выше второй степени. Для решения уравнений большей степени, а также трансцендентных уравнений можно использовать графические и численные методы.

Под численным методом решения уравнений понимают метод нахождения численных значений корней уравнения с заданной точностью.

Пусть дано уравнение $f(x) = 0$,

где $f(x)$ - функция непрерывная на отрезке $[a; b]$ и дифференцируемая на интервале $]a; b[$, т.е. включая и граничные значения.

Всякое число c, принадлежащее отрезку $[a; b]$, такое, что $f(c) = 0$, называется корнем уравнения на отрезке $[a; b]$.

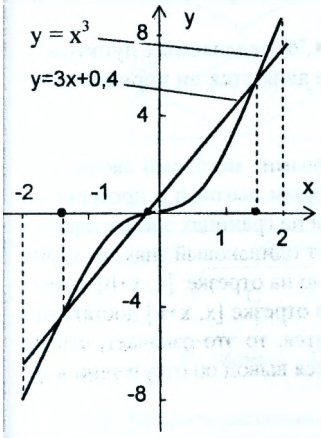
Процесс нахождения корней уравнения состоит из двух этапов:

1) отделения корней;

2) уточнения значения корней с заданной точностью h.

Отделением корней уравнения называется процесс выделения из области определения функции f(x) отрезков $[a_i; b_i]$, в каждом из которых содержится один, и только один, корень уравнения $f(x) = 0$.

Под уточнением значения корня с заданной точностью понимают сужение границ отрезка отделения корня $[a_i; b_i]$ до длины, не превосходящей заданной точности h.



9.4.6. Графический метод поиска корней уравнения

Рассмотрим пример решения задачи графическим методом.

Пример 9.4.11. Пусть дано уравнение

$$x^3 - 3x - 0,4 = 0.$$

Требуется отделить корни уравнения.

Задача может быть решена графически и аналитически, а также численными методами.

Отделение корней уравнения графически.

Решение. Запишем уравнение в виде:

$$x^3 = 3x + 0,4$$

и построим графики функций $y = x^3$ и $y = 3x + 0,4$ (рис. 9.4.6). Абсциссы точек пересечения этих графиков и будут корнями исходного уравнения. Из рисунка следует, что исходное уравнение имеет три действительных корня: c_1 — на отрезке [-2; -1]; c_2 — на отрезке [-1; 0]; c_3 — на отрезке [1; 2].

При графическом отделении корней уравнения результат зависит от точности построения графиков.

Отделение корней аналитически.

В основе аналитического метода лежит утверждение: если функция *f* непрерывна на отрезке *[a; b]* и принимает на концах этого отрезка значения противоположных знаков, то внутри отрезка существует корень уравнения и при том единственный, если производная на интервале *]a, b[* сохраняет постоянный знак.

Границами интервалов знакопостоянства производной от функции являются ее критические точки (точки минимумов и максимумов), в критических точках производная обращается в ноль. На каждом таком интервале будет находиться только один корень, если значения функции на концах интервала имеют противоположные знаки. В качестве границ интервалов знакопостоянства функции кроме критических точек необходимо рассматривать также и границы отрезка.

Для отделения корней используется следующая схема:

1. Найти производную *f'* .

$f'(x) = (x^3 - 3x - 0,4) = 3x^2 - 3 = 3(x^2 - 1) = 3(x + 1)(x - 1).$

2. Найти критические точки функции: -1, +1

3. Составить таблицу знаков функции *f:* на границах отрезка *[a; b]* и в критических точках:

x	-2	-1	+1	+2
f(x)	-	+	-	+

4. Определить отрезки, на концах которых функция принимает значения нения противоположных знаков.

Уравнение имеет три корня, так как происходит три перемены знака функции *f*: C_1 *[-2, -1]*; C_2 *[-1; 1]*;C_3*[1; 2]*.

Таким образом, корень находится в интервалах, определяемых пунктом 4. При решении задач необходимо также проверять, не являются ли корнем также и критические точки.

Отделение корней численными методами.

Процедура отделения корней уравнения численными методами сводится к табулированию функции на заданном отрезке с некоторым шагом h и проверке на каждом шаге условия знакопостоянства функции. Если на границах элементарного отрезка h значения функции y1=f(x) и y2=f(x+h) имеют одинаковый знак, то корня на этом отрезке нет, если y1 и y2 имеют разные знаки, то на отрезке [x, x+h] имеется корень. Для проверки знакопостоянства функции на отрезке [x, x+h] достаточно проверить условие y1*y2<0. Если условие выполняется, то это означает, что на данном отрезке есть корень, в противном случае делается вывод об отсутствии корня на отрезке [x, x+h].

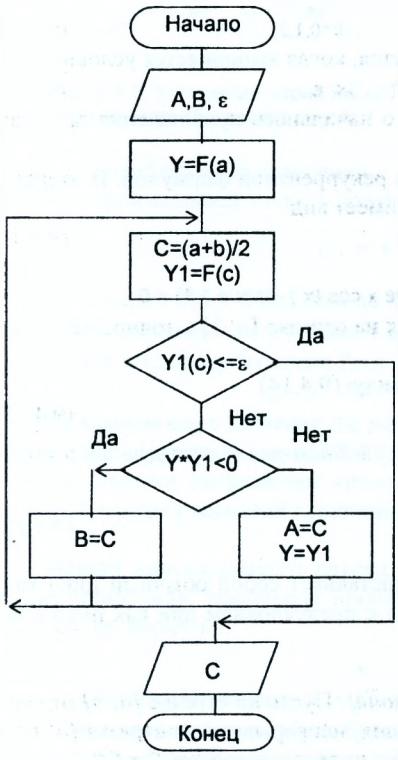
Уточнение значения корня на отрезках отделения

Уточнение значений корней на отрезках отделения осуществляется различными методами одномерной поисковой оптимизации: деления отрезка пополам, простых итераций, касательных и др.

При выборе метода следует исходить из:
а) сложности подготовки задачи к решению;
б) быстродействия алгоритма;
в) времени и точности решения задачи.
Рассмотрим некоторые методы.

Метод дихотомии (деления отрезка пополам)

Достоинством метода является его простота, а также то, что этот метод не накладывает никаких ограничений на исследуемую функцию. Достаточно, чтобы на данном отрезке был корень и притом только один.



9.4.7. Алгоритм решения уравнения методом деления отрезка пополам

Суть метода состоит в следующем. Отрезок отделения корня *[a ; b]* делят пополам и точку деления *c* принимают за значение корня. Сразу же проверяют, если значение функции в точке *c* станет меньше или равно требуемой точность вычисления значения функции, или длина отрезка [a, c] станет меньше требуемой точности уточнения значения корня, то точка *c* принимается за значение корня и процесс поиска корня заканчивается. В противном случае определяют, на каком из отрезков *[a ;c]* или *[c ; b]* функция меняет знак и выбирают его для последующего анализа. Если F(a)*F(c)<0, то корень находится слева от точки C, в ином случае корень находится справа от точки C. Пусть F(a)*F(c)<0, то есть корень лежит на отрезке *[a; c]*. Тогда точку B переносят в точку C, то есть B=C и этот отрезок снова делят пополам и так далее. Каждый раз корень уравнения оказывается локализованным на все меньшем и меньшем отрезке. Процесс поиска корня заканчивается, когда длина отрезка [a ; b] станет меньше или равна заданной точности уточнения корня ε или значение функции в точке *c* станет меньше или равно требуемой точности вычисления значения функции в этой точке. Метод всегда обеспечивает сходимость процесса, то есть последовательность приближенных значений корня c_1, c_2, c_3,... ,c_n будет сходиться к самому значению

корня *c*. Алгоритм уточнения значения корня на отрезке отделения методом деления отрезка пополам приведен на рис. 9.4.7.

Метод итераций.

В данном методе требуется задать начальное приближение x_0 на отрезке отделения корня [*a* ; *b*], выбрать шаг изменения переменной Δx и последовательно решать уравнение вида:

$$x = \varphi(x), \qquad (9.4.14)$$

полученное путем эквивалентных преобразований из исходного уравнения $f(x) = 0$. В результате решения этого уравнения получаем ряд приближений корня:

$$x_{k+1}=\varphi(x_k), \qquad k=0,1,2,... \qquad (9.4.15)$$

Процесс уточнения корня прекращается, когда выполняется условие

$$|x_{k+1} - (x_k)| <\varepsilon \quad \text{или} \quad f(x_{k+1})< \varepsilon \qquad (9.4.16)$$

Время поиска зависит от выбранного начального приближения x_0 и шага Δx.

Выражение вида (9.4.14) называется рекуррентной формулой. В итерационных алгоритмах рекуррентная формула имеет вид

$$x_{i+1} = \varphi(x_i). \qquad (9.4.17)$$

Пример 9.4.12. Пусть дано уравнение $x \cos(x) - \ln(x + 1{,}1) = 0$

Требуется уточнить корни уравнения на отрезке [*a; b*] с точностью до $\varepsilon = 0{,}01$.

Решение. Преобразуем уравнение к виду (9.4.14)

$$x\cos(x)=\ln(x+1,1) \qquad (9.4.18)$$

Представим выражение (9.4.18) в виде, удобном для использования в итерационном цикле:

$$x_{i+1} = \frac{\ln(x_i + 1{,}1)}{\cos(x_i)} \qquad (9.4.19)$$

Алгоритм итерационного цикла представляет собой обычный цикл типа "Пока" и может быть реализован как цикл с предусловием или как цикл с постусловием.

Метод касательных (метод Ньютона). Пусть на отрезке *[a; b]* отделен корень *c* уравнения $f(x) = 0$ и $f(x)$ — функция, непрерывная на отрезке *[a; b]*, и на интервале *]a; b[* существуют отличные от нуля производные f' и f''.

Выберем начальное приближение x_0.

Если на отрезке *[a; b]* $f'(x)f''(x) > 0$, то нулевое приближение корня выбираем в точке b: $x_0 = b$, если же $f'(x)f''(x) < 0$, то нулевое приближение выбираем в точке a: $x_0 = a$.

Построим график функции $y = f(x)$. Пусть для определенности $f'(x) > 0$ и $f''(x) > 0$, то есть функция вогнутая (рис. 9.4.8). Проведем касательную к графику функции в точке $B(b, f(b))$. Ее уравнение будет иметь вид:

$$y = f(b) + f'(b)(x - b). \quad (9.4.20)$$

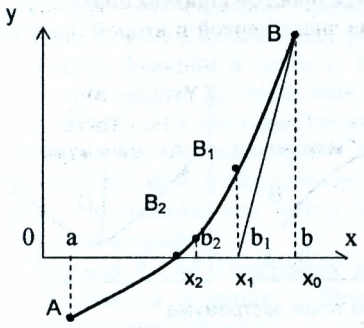


Рис. 9.4.8. Уточнение корня на отрезке методом касательной

Найдем точку пересечения касательной с осью x (точка b_1). Так как в этой точке $y=0$, то, решая уравнение (9.4.20) относительно x, получим:

$$x = b - \frac{f(b)}{f'(b)}. \quad (9.4.21)$$

Это и есть первое приближение решения уравнения - x_1.

Проведем касательную к графику функции в точке B_1 $(x_1; f(x_1))$. Найдем точку пересечения касательной с осью x – x_2.

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_2)}. \quad (9.4.22)$$

Это второе приближения уравнения.

Для произвольной точки i получим

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}. \quad (9.4.23)$$

Отношение $f(x_i)/ f'(x_i)$ есть не что иное, как приращение значения аргумента - Δx_i. Действительно, так как $f'(x_i)$ численно равно тангенсу угла наклона касательной к графику функции $f'(x_i) = tg(\alpha) = Bb/bb_1$, то

$$x_0 - x_1 = bb_1 = Bb/tg(\alpha). \quad (9.4.24)$$

На каждом шаге значение Δx_i меняется.

Таким образом, формула (9.4.23) дает последовательность приближений x_i корня. Процесс вычислений заканчивается, когда $f(x_{i+1})$ становится меньше некоторой наперед заданной величины ϵ, точности поиска корня.

Метод наискорейшего спуска. Пусть дано $f(x) = 0$. Функция дважды дифференцируемая. Начальное приближение x_0. Очередное приближение вычисляется по формуле:

$$x_{i+1} = x_i + \lambda_i v_i, \quad (9.4.25)$$

где v_i — направление поиска: $v_i = f'(x)$; λ_i - шаг: $\lambda_i = (f'(x_i))^2 / f''(x_i)$.

Поиск прекращается, когда будет выполнено условие $x_{i+1} - x_i < \varepsilon$, где ε — требуемая точность поиска корня.

Недостатками методов касательной и наискорейшего спуска является необходимость нахождения производных.

Приближенный поиск экстремумов

Во многих задачах вычислительной математики требуется найти максимальное или минимальное значение функции (экстремум) на заданном отрезке $[a; b]$. Экстремум может находиться на концах отрезка (рис. 9.4.9 а) или внутри

отрезка. Функция может быть выпуклая (выпуклая вверх) (рис. 9.4.9 б) или вогнутая (выпуклая вниз) (рис.9.4.9. в). Имеются простые способы анализа функций на отрезках, основанные на определении знака первой и второй производной.

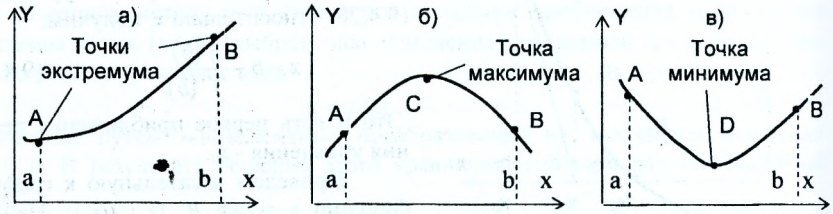


Рис. 9.4.9. К определению точек экстремума

Если функция непрерывна на отрезке, то:

если первая производная $f'(x)$ не меняет знак на концах отрезка, то функция является монотонно убывающей, если $f'(x) < 0$, и возрастающей, если $f'(x) > 0$. Монотонные функции достигают экстремума на концах отрезка; если на концах отрезка производная функции меняет знак, то экстремум находится внутри отрезка. Причем, если вторая производная больше нуля, то в точке экстремума функция достигнет минимума. Если вторая производная меньше нуля, то в точке экстремума функция достигает максимума.

Поиск экстремума функции внутри отрезка может быть выполнен с помощью табулирования функции. Для этого достаточно в процессе табулирования сравнивать текущее значение функции с переменной, значение которой принято за максимум (минимум).

Если на отрезке один экстремум, то наиболее эффективным методом поиска будет один из методов одномерной поисковой оптимизации. Для этой цели могут быть использованы те же методы, что и для уточнения корней: метод дихотомии, "золотого сечения", простых итераций.

Пример 9.4.13. Проанализировать функцию $y = 5x^2 - 8x + 1$ на отрезке $[-1, +1]$.

Решение. Находим первую производную функции y:

$$y' = 10x - 8; \quad y'(a) = -18; \quad y'(b) = 2.$$

Функция имеет экстремум на заданном отрезке, так как первая производная на концах отрезка меняет знак. Вторая производная функции y" = 10, то есть больше нуля. Следовательно, функция имеет на отрезке [-1; 1] точку минимума.

Метод "золотого сечения"

В отличие от метода итераций, метод "золотого сечения" не требует задания начального значения корня. Достаточно указать требуемую точность поиска. В этом методе отрезок отделения $[a; b]$ делится на неравные части. Одна точка (D) задается на расстоянии 0,618, а другая (C) – на расстоянии 0,382 длины отрезка $[a; b]$ от точки a (рис. 9.4.10) и определяют, в какой из точек C или D производная функции меняет знак по отношению к знаку производной от

этой функции в точке *a*. Если в точке D производная функции знак не меняет, то сразу переходят к анализу отрезка *[d , b]*.

Если в точке *d* производная функции меняет знак, то проверяют значение производной функции в точке *c*. Если в точке *c* производная функции знак не меняет, то переходят к анализу отрезка *[c,d]*, в противном случае анализируют отрезок *[a, c]*. Данный метод обеспечивает более быструю сходимость, чем метод дихотомии и метод итераций.

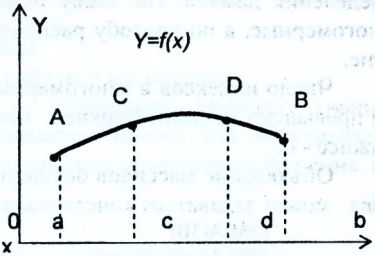


Рис. 9.4.10. Поиск экстремума функции методом "золотого сечения"

Пример 9.4.14. Программа для поиска экстремума функции $y=\sin(x^2)+0{,}5$ методом "Золотого сечения"

```
CLS
REM Определение экстремума функции методом золотого сечения
DEF fnf (x) = 2*x*COS(x ^ 2)        ' производная
DEF fnf1 (x) = SIN(x ^ 2) + .5      ' функция
INPUT "Введите границы отрезка A и B", a, b
INPUT "Укажите требуемую точность E ", e
k = b    ' сохранение значения b
m1:
c = a + .382 * (b - a): d = a + .618 * (b - a)
y = fnf(a)              ' fnf – функция пользователя
y1 = fnf(b)
IF b - a < e THEN x = (b + a) / 2: PRINT "экстремум: "; x, "f'(x)="; fnf(x), "f(x)="; fnf1(x): END
b = c: y1 = fnf(b)
IF y * y1 < 0 THEN GOTO m1    ' проверяется отрезок [A,C]
a = c: b = d: y = y1: y1 = fnf(b)
IF y * y1 < 0 THEN GOTO m1    ' проверяется отрезок [C, D]
a = d: b = k: y = y1: y1 = fnf(b)
IF y * y1 < 0 THEN GOTO m1    ' проверяется отрезок [D,B]
x = b:
PRINT "экстремум: "; x, "f'(b)="; y1, "f(b)="; fnf1(x)
END
```

9.4.5. Массивы

Понятие об индексированных переменных

Массивом называется совокупность индексированных элементов ***a(i,j)***:

$$\begin{bmatrix} a_{11}\ a_{12}\ \ldots\ a_{1j}\ \ldots\ a_{1n} \\ \ldots \\ a_{i1}\ a_{i2}\ \ldots\ a_{ij}\ \ldots\ a_{in} \\ \ldots \\ a_{m1}\ a_{m2}\ \ldots\ a_{mj}\ \ldots\ a_{mn} \end{bmatrix} \qquad (9.4.26)$$

Здесь a_{ij} – элемент массива. Индекс i означает номер строки, j – номер столбца.

Массивы можно классифицировать по числу измерений и способу распределения памяти. По числу измерений массивы делятся на одномерные и многомерные, а по способу распределения памяти на статические и динамические.

Число индексов в многомерных массивах ограничено и в системе QBasic не превышает восьми. Верхнее значение индекса может быть равно 32767, а нижнее - 0.

Объявление массивов осуществляется оператором DIM. Размерность массива может задаваться константами или переменными:

```
DIM A(10)
DIM A1(m), B(m,n)
DIM C(нижняя_граница TO верхняя_граница).
DIM A2(1 to m), B2(5 to 10, 1 to 20)
```

DIM A(10) – одномерный массив, содержит 11 элементов;

DIM B(5 TO 10, 1 TO 20) – двухмерный массив, имеет 6 строк и 20 столбцов. Нумерация строк начинается с 5, а нумерация столбцов с единицы. При объявлении массивов можно указать и их тип.

По умолчанию нумерация элементов массива начинается с нуля. Для изменения индексации с нуля на единицу используется оператор ***Option Base N***, где N может принимать значения 0 и 1. Однако, при объявлении массивов можно задавать произвольные значения верхних и нижних границ массива, как показано выше.

В зависимости от способа распределения памяти массивы делятся на массивы со статическим распределением памяти и массивы с динамическим распределением памяти.

Массивы со статическим распределением памяти (для краткости – статические массивы) объявляются операторами ***Dim*** с атрибутом ***STATIC***. Если размерность массива задана константами, то массив считается статическим:

```
DIM [STATIC] ABC(1 TO 5, 1 TO 2).
```

Размерность статического массива не может быть изменена при выполнении программы.

Массивы с динамическим распределением памяти (для краткости – динамические массивы) объявляются оператором ***DIM*** с атрибутом ***DINAMIC***. Размерности динамического массива задаются переменными:

```
DIM [DINAMIC]  A ( m, n)
```

При объявлении массивов вместе с оператором DIM и REDIM может использоваться также атрибут SHARED. Этот атрибут означает, что данный массив является массивом общего пользования, доступен нескольким программным модулям:

```
REDIM SHARED  B ( m, n)
```

Переобъявление размерностей массива осуществляется оператором *REDIM*:

```
REDIM B3(m, n)
```

Удаление массива осуществляется командой *Erase*:

ERASE < имя_массива>

Для статических массивов команда *Erase* не удаляет массив, а только очищает его.

Функции для работы с массивами

QBasic имеет несколько функций для работы с массивами:

LBOUND(ИмяМассива, Индекс) – возвращает нижнюю границу диапазона индекса массива. Индекс указывается только для многомерных массивов, и определяет, к какому измерению массива применяется функция. По умолчанию индекс равен 1. Например:

```
Dim A(5 To 10,15 To 20)
...
```

Lbound (A, 2) – определяется нижняя граница второго измерения массива;

UBOUND (ИмяМассива, Индекс) – возвращает верхнюю границу диапазона индекса массива.

Ввод данных в массив и вывод данных из массива

Так как число элементов массива представляет собой всегда счетное множество, то для работы с массивами используется оператор цикла FOR/NEXT.

Пример 9.4.15. Ввод и вывод одномерного массива.

```
REM Ввод одномерного массива
OPTION BASE 1
30  INPUT "Укажите размерность массива ",N
40  DIM A(N)
60  FOR   i = 1 TO N
70        PRINT "Введите"; i; "элемент массива ";
80        INPUT  "",A(i)
90  NEXT i
100 END
```

Вначале по запросу программы, с клавиатуры вводится размерность массива, строка 30. Объявляется одномерный массив А размерностью N (строка 40) как массив вещественных переменных одинарной точности (по умолчанию). Затем организуется цикл с заданным числом повторений (строки 60 – 90). В строке 70 оператор PRINT выводит на экран запрос на ввод текущего элемента массива. В конце оператора имеется символ "точка с запятой", который после вывода информации оставляет курсор в строке ввода, поэтому запрос на ввод данных с помощью оператора INPUT, строка 80, будет выведен в этой же строке, сразу же после вывода текстовой информации.

```
REM Вывод одномерного массива
110   FOR   i = 1 TO N
120           PRINT  A(i),
130   NEXT i
140   END
```

Алгоритм вывода массива на экран или печать подобен алгоритму ввода массива. В данном алгоритме необходимо обратить внимание на символ "запя-

тая" в конце оператора PRINT (строка 120). При наличии данного символа данные будут выводиться в пять колонок. Если этот символ удалить, то данные будут выведены в один столбец (см. описание оператора PRINT).

При вводе (выводе) данных в многомерный массив необходимо организовать столько вложенных циклов, какова размерность массива.

Пример 9.4.16. Ввод и вывод двухмерного массива

```
INPUT "Укажите число строк ",n
INPUT "Укажите число столбцов ",m
DIM A(n, m)
REM Ввод данных в двухмерный массив
FOR i = 1 TO n
   FOR j = 1 TO m
      PRINT "Введите A(" ; i ; "," ; j ; "элемент массива";
      INPUT  A(i, j)
   NEXT j
NEXT i
REM Вывод данных из двухмерного массива
FOR i = 1 TO n
   FOR j = 1 TO m
      PRINT  A(i, j);
   NEXT j
   PRINT
NEXT i
```

В данном примере вывод элементов массива осуществляется в виде матрицы. Это обеспечивается за счет использования разделителя ";" в конце оператора PRINT. Символ ";" в конце оператора PRINT оставляет курсор в текущей строке. Когда ввод данных в первую строку закончится, то оператор PRINT без параметров возвращает курсор в начало строки.

Процедуры ввода данных в массив и вывода данных из массива типичные, поэтому их можно оформить в виде подпрограмм (см. раздел 9.4.6), например:

```
SUB  Massiv1(A(),n)
REM Подпрограмма вывода данных из одномерного массива.
   FOR   i = 1 TO N
      PRINT  A(i),
   NEXT i
END SUB
```

Операции с массивами

Массивы широко используются для хранения данных. При работе с файлами данных, вводе и выводе информации удобнее всего записывать данные предварительно в массив.

Умножение массива на число, вектор, массив

Задача 9.4.1. Умножить все элементы двухмерного массива на число c.

Решение. Для умножения массива на число необходимо умножить на это число каждый элемент массива.

```
REM Умножение массива на число
INPUT "Укажите число строк ",n
INPUT "Укажите число столбцов ",m
INPUT "Введите число С ",C
DIM A(n, m)
REM Ввод данных в массив опущен
FOR i = 1 TO n
   FOR j = 1 TO m
       PRINT  A(i, j)*C;
   NEXT j
   PRINT
NEXT i
```

Задача 9.4.2. Найти скалярное произведение векторов А и В.

Решение. При скалярном умножении вектора на вектор получается переменная, значение которой равно сумме произведений каждого элемента второго вектора на соответствующий элемент первого вектора:

$$C= \sum a_i * b_i$$

```
REM Скалярное умножение вектора на вектор
INPUT "Укажите число элементов в векторе ",n
DIM A(n), B(n)
REM ввод данных в вектора  A и B опущен
C=0
FOR i = 1 TO n
      C=C+A(i)xB(i)
NEXT i
PRINT C
```

Задача 9.4.3. Умножить двухмерный массив А на вектор В, результаты поместить в вектор С.

Решение. При умножении массива на вектор необходимо чтобы число столбцов в массиве было равно числу элементов в векторе. При умножении массива на вектор получается новый вектор, в котором число элементов равно числу строк в массиве, а каждый элемент этого вектора равен сумме произведений каждого элемента строки массива на соответствующий элемент вектора, т. е. $c_i= \sum a_{ij} * b_j$. Примером может быть решение системы уравнений матричным способом (см. раздел 7.3.3).

```
REM Умножение массива на вектор
INPUT "Укажите число строк ",n
INPUT "Укажите число столбцов ",m
DIM A(n, m), B(m), C(n)
REM ввод данных в массив и вектор  A и B опущен
FOR i = 1 TO n
   FOR j = 1 TO m
       C(i)=C(i)+ A(i, j)*B(j);
   NEXT j
NEXT                                                    i
```

Задача 9.4.4. Умножить двухмерный массив А на массив В, результаты поместить в массив С.

Решение. При умножении массива А на массив В необходимо чтобы число столбцов в массиве А было равно числу строк в массиве В. При умножении массива А на массив В получается новый массив С, в котором число строк равно числу строк первого массива, число столбцов - числу столбцов второго массива, а каждый элемент этого массива равен сумме произведений каждого элемента строки массива А на соответствующий элемент столбца В, т. е. $c_{ik}= \sum a_{ij}*b_{jk}$.

```
REM Умножение массива на массив
INPUT "Укажите число строк массива A",n
INPUT "Укажите число столбцов массива A",m
INPUT "Укажите число столбцов массива C",P
DIM A(n, m), B(m,p), C(n,p)
REM ввод данных в массивы  A и B опущен
FOR k= 1 TO p
   FOR i = 1 TO n
      FOR j = 1 TO m
          C(i,k)=C(i,k)+ A(i, j)*B(j,k)
      NEXT j
   NEXT i
NEXT k
```

Вычисление сумм элементов массивов

Задача 9.4.5. Вычислить сумму элементов одномерного массива.

Решение.

```
S=0
FOR j = 1 TO n
    S=S+ A(i)
NEXT j
PRINT S
```

Задача 9.4.6. Вычислить сумму нечетных элементов двухмерного массива.

Решение. Элемент массива считается четным, если сумма его индексов четная, и наоборот – элемент массива считается нечетным, если сумма индексов элемента массива нечетная. Например, A(0,0), A(1,5) – четные элементы массива, а элементы A(0,1), A(4,7) – нечетные. Поэтому для определения суммы нечетных элементов массива необходимо в цикле подсчитывать сумму индексов элемента массива и проверять, является эта сумма четной или нечетной. Для проверки четности суммы индексов можно использовать арифметический оператор MOD (см. раздел 7.1.4.).

```
S=0
FOR i = 1 TO n
   FOR j = 1 TO m
       k=i+j
       IF k MOD 2 <>0 THEN S=S+ A(i, j)
   NEXT j
NEXT i
PRINT S
```

Поиск минимального и максимального элементов массива

Задача 9.4.7. Найти максимальный и минимальный элементы одномерного массива.

Решение. При поиске минимального и максимального значений элементов массива в качестве начального значения вспомогательным переменным Amax и Amin необходимо присвоить значение любого элемента массива, например, нулевого:

```
Amax=A(0): Amin=A(0)
FOR i = 1 TO n
    IF A(i)>Amax  THEN Amax=A(i)
    IF A(i)<Amin  THEN Amin=A(i)
NEXT i
PRINT "Amax =";Amax, "Amin=";Amin
```

Сортировка массивов

Одной из операций с массивами является сортировка данных.

Известно много различных способов сортировки массивов. Однако, независимо от используемого способа в программе осуществляется сравнение текущего элемента массива с другим элементом этого же массива и, в зависимости от результатов сравнения, осуществляется обмен данными между этими элементами. Qbasic имеет полезную функции ***SWAP***, которая предназначена для обмена значениями двух переменных.

Самая простая процедура сортировки – перебор.

```
FOR = i=1 To n - 1
    FOR j = i + 1 To n
        IF A(j) < A(i) THEN SWAP A(i), A(j)
    NEXT j
NEXT i
```

Приведенная процедура обеспечивает сортировку по возрастанию значения элементов массива. Для сортировки элементов массива по убыванию их значений достаточно поменять знак отношения в третьей строке.

Решение систем линейных уравнений

Системой линейных уравнений с n неизвестными $x_1, x_2, \ldots, x_n$ называется система вида:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1j}x_j + \ldots + a_{1n}x_n = a_{1,n+1} \\ \ldots a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{ij}x_j + \ldots + a_{in}x_n = a_{i,n+1} \\ \ldots a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nj}x_j + \ldots + a_{nn}x_n = a_{n,n+1} \end{cases} \quad (9.4.27)$$

Здесь a_{ij} — *коэффициенты* при неизвестных, $a_{i,n+1}$ — *свободные члены.* Индекс i означает номер уравнения, j — номер неизвестного.

Решением системы (9.4.27) называется совокупность таких значений переменных $c_1, \ldots, c_n$, при подстановке которых в данную систему каждое уравнение системы обращается в числовое равенство.

Система уравнений называется *совместной*, если она имеет хотя бы одно решение, и *несовместной*, если она не имеет ни одного решения. Две системы уравнений называются *эквивалентными*, если множество их решений совпадают. Преобразования, переводящие одну систему в эквивалентную ей систему, называются эквивалентными.

Перестановка двух уравнений системы, умножение уравнения системы на любое число, отличное от нуля, прибавление к уравнению другого уравнения, умноженного на любое число, отличное от нуля, являются эквивалентными преобразованиями.

Одним из распространенных методов решения систем линейных уравнений является метод Гаусса (метод исключений). Методом Гаусса получают аналитические выражения для вычисления точных значений переменных. В основе метода лежит прием последовательного исключения переменных для получения эквивалентной треугольной или трапецеидальной системы уравнений. Для решения системы уравнений методом Гаусса с использованием ЭВМ необходимо на основании исходной системы уравнений (9.4.27) составить матрицу коэффициентов и свободных членов - *расширенную матрицу*.

Рассмотрим пример решения системы методом Гаусса на основе системы третьего порядка. Составим расширенную матрицу:

$$\begin{vmatrix} a_{11} & a_{12} & a_{13} & b_1 \\ a_{21} & a_{22} & a_{23} & b_2 \\ a_{31} & a_{32} & a_{33} & b_3 \end{vmatrix} \qquad (9.4.28)$$

Прямой ход.

Сначала с помощью первого уравнения исключается x_1 из второго и третьего уравнений. Для этого каждый элемент первой строки разделим на коэффициент a_{11}, умножим на коэффициент a_{21} и вычтем полученные значения из соответствующих элементов второй строки:

$$a'_{22}=a_{22}-a_{12}/a_{11}*a_{21},\ a'_{23}=a_{23}-a_{13}/a_{11}*a_{21},\ b'_2=b_2-b_1/a_{11}*a_{12}$$
$$a'_{32}=a_{32}-a_{12}/a_{11}*a_{32},\ a'_{33}=a_{33}-a_{13}/a_{11}*a_{33},\ b'_3=b_3-b_1/a_{11}*a_{13}$$

или в общем виде:

$$a_{ij}=a_{ij}-a_{i1}/a_{11}*a_{1j}\quad i=2,3,\ j=2,3$$
$$b_i=b_i-b_1/a_{11}*a_{1i},\quad i=2,3.$$

Аналогично с помощью второго уравнения исключается x_2 из третьего уравнения:

$$a''_{32}=a'_{32}-a'_{22}/a'_{22}*a'_{32},\ a''_{33}=a'_{33}-a'_{23}/a'_{22}*a'_{33},\ b''_3=b'_3-b'_2/a'_{22}*a'_{23}$$

В результате получим треугольную матрицу:

$$\begin{vmatrix} a_{11} & a_{12} & a_{13} & b_1 \\ 0 & a'_{22} & a'_{23} & b'_2 \\ 0 & 0 & a''_{33} & b''_3 \end{vmatrix}$$

Обратный ход.

Обратный ход начинается с решения третьего уравнения системы:

$$x_3=b''_3/a''_{33}.$$

Используя это значение можно найти x_2 из второго уравнения, а затем x_1 из первого уравнения:

$$x_2=(b'_2-a'_{23}*x_3)/a'_{22};\qquad x_1=(b_1-a_{12}*x_2-a_{13}*x_3)/a_{11}.$$

В данном примере предполагалось, что все коэффициенты матрицы не нулевые. На практике необходимо проверять, чтобы элемент матрицы, на который производится деление не был нулевым. Если ведущий элемент матрицы равен нулю, то необходимо сделать перестановку строк матрицы.

Известны численные методы решения систем линейных уравнений, например метод простых итераций. Численные методы дают приближенное решение системы уравнений. Однако при большом числе переменных, например $n > 100$, сложность метода итераций меньше сложности метода Гаусса.

Программа решения системы уравнений методом Гаусса

Входные данные:

N — количество уравнении и неизвестных (степень системы уравнений);

A (N,N+1) — матрица коэффициентов и свободных членов (расширенная матрица системы уравнений).

Вспомогательный массивы *C(N)*.

Выходные данные: X(N) — решение системы

```
INPUT "Укажите степень системы уравнений ",N
DIM A(N,N+1), C(N), X(N)
REM  Ввод коэффициентов расширенной матрицы
FOR i=0 TO N
    FOR j=0 TO N+1
        PRINT "Введите коэффициент  a("I","j") ";
        INPUT " ",A(i,j)
    NEXT j
NEXT i
REM  Начало цикла вычислений
FOR K=0 TO N-1
    REM  Процедура перестановки строк
    IF A(K,K)<>0 THEN GOTO 7070
    FOR P=K+1 TO N
        IF A(P,K)>=1E-08 THEN GOTO 7030
    NEXT P
    PRINT "НЕТ РЕШЕНИЯ"
7015    STOP
7020    GOTO 7100
7030    FOR J=K TO N+1
            R=A(K,J)
            A(K,J)=A(P,J)
            A(P,J)=R
        NEXT J
' Конец процедуры перестановки строк
REM Прямой ход
7070  FOR I=K+1 TO N
        B=A(I,K)/A(K,K)
        FOR J=K TO N+1              ' вычисление коэффициентов
            A(I,J)=A(I,J)-B*A(K,J)
        NEXT J
    NEXT I
NEXT K
REM Обратный ход
X(N)=A(N,N+1)/A(N,N)
```

```
FOR K=N-1 TO 0 STEP -1
      R=0
      FOR J=K+1 TO N
            R=R+A(K,J)*X(J)
      NEXT J
      X(K)=(A(K,N+1)-R)/A(K,K)
NEXT K
GOTO 7020
PRINT "ОШИБКА": GOTO 7015
7100 REM вывод результата
      FOR i=0 TO N
            PRINT X(i)
      NEXT i
END ' конец программы
```

Интерполирование функций

При решении многих задач значения функции задаются в виде двухмерной таблицы, в одной строке которой задаются значения аргумента, а в другой соответствующие значения функции. Точки, в которых определены значения функции, называются *узлами интерполяции*. Но этого бывает недостаточно и требуется найти значения функций в некоторых точках, отличных от узлов интерполяции. Вычисление значения функции $f(x)$ в точках x, отличных от узлов интерполяции, называется *интерполяцией*.

Методы интерполяции используются в управляющих ЭВМ для сокращения времени вычислений. Например, требуемое состояние технологического процесса можно задать в виде таблицы значений функций в контрольных точках. Значение текущих параметров процесса рассчитывается ЭВМ с использованием указанных методов.

Для поиска значений функции в точках, отличных от узлов интерполяции, необходимо построить функцию F, принимающую в узлах интерполяции те же значения, что и функция $f(x)$, то есть $F(x_0) = f(x_0)$, $F(x_1) = f(x_1)$, ..., $F(x_n) = f(x_n)$. Таких функций может быть построено бесконечное множество.

Способ построения функции F, принимающей в узлах интерполяции те же значения, что и функция f, называется *интерполированием*, а функцию F называют *интерполирующей функцией*. Геометрически интерполированию соот-

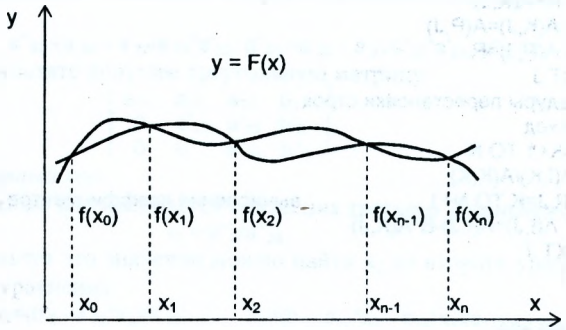


Рис. 9.4.11. Интерполирование

ветствует нахождение плавной кривой, проходящей через заданные точки $(x_i ; f(x_i))$ (рис. 9.4.11). Так как таких кривых может быть бесчисленное множество, то задача отыскания интерполирующей функции в общей постановке не решается однозначно. В качестве интерполирующей функции обычно используется полином Лагранжа, многочлен Ньютона, непрерывные дроби, многочлен вида $Pn(x) == a_0 x^n + a_1 x^{n-1} +... + a_{n-1} x + a_n$. Наибольшей простотой с точки зрения вычисления обладают интерполяционный полином Лагранжа и интерполяционный многочлен Ньютона.

Интерполяционный многочлен Лагранжа имеет следующий вид:

$$L_n(x) = \sum_{i=0}^{n} \frac{(x-x_0)(x-x_1)...(x-x_{i-1})(x-x_{i+1})...(x-x_n)}{(x_i-x_0)(x_i-x_1)...(x_i-x_{i-1})(x_i-x_{i+1})...(x_i-x_n)}. \qquad (9.4.29)$$

При n=1 получаем формулу для линейной интерполяции:

$$L_1 = y_0 \frac{x-x_1}{x_0-x_1} + y_1 \frac{x-x_0}{x_1-x_0}. \qquad (9.4.30)$$

Для ускорения вычислений многочлена при различных значениях n применяют схему Эйткена, в которой не требуется явного построения многочлена:

$$y(x,x_0,x_1) = \frac{1}{x_1-x_0}\begin{vmatrix} y_0 & x_0-x \\ y_1 & x_1-x \end{vmatrix},$$

$$y(x,x_0,x_2) = \frac{1}{x_2-x_0}\begin{vmatrix} y_0 & x_0-x \\ y_2 & x_2-x \end{vmatrix}, \qquad (9.4.31)$$

$$y(x,x_0,x_1,x_2) = \frac{1}{x_2-x_1}\begin{vmatrix} y(x,x_0,x_1) & x_1-x \\ y(x,x_0,x_2) & x_2-x \end{vmatrix}.$$

Теперь для вычисления значения функции в произвольной точке числовой оси не совпадающей с узлами интерполяции следует вычислить $y(x)=L_n(x)$.

Программа интерполяции функций методом Лагранжа

Входные данные:
N — количество интервалов интерполяции;
A — значение аргумента;
X(N), F(N) — массивы значений аргумента и функции;
Выходные данные:

```
INPUT "Укажите число интервалов интерполяции  ",N
INPUT "Укажите значение аргумента x ",A
DIM X(N), F(N)
FOR i=0 TO N
    PRINT "Укажите значения X и Y в "; I;" узле";
    INPUT " ",X(i), F(i)
NEXT i
REM интерполяция полиномом Лагранжа по Эйткену
FOR J=0 TO N-1
    FOR I=J+1 TO N
```

```
            F(I)=((A-X(J))*F(I)-(A-X(I))*F(J))/(X(I)-X(J))
        NEXT I
    NEXT J
    F1=F(N)
    PRINT "Значение функции "; F1
```

Интерполяционная формула Ньютона имеет следующий вид:

$$P_{n-1}(x)=y_1+(x-x_1)f(x_1;x_2)+\ (x-x_1)(x-x_2)f(x_1;x_2;x_3)+\ (x-x_1)(x-x_2)\ \ ...\ \ (x-x_{n-1})$$
$$f(x_1;x_2;...;x_n)=\sum_{i=0}^{n-1}A_k\varphi_i\ ; \qquad (9.4.32)$$

где $A0=y_1$, $A_k=f(x_1;x_2;...;x_{k+1})$ – разделенные разности k –го порядка.

Программа интерполяции по Ньютону

Программа позволяет найти значение функции по значению переменной в промежуточной точке между узлами интерполяции.

Входные данные:
N — степень полинома;
X(N), Y(N) — массивы *x, y.*
A – значение аргумента
Выходные данные:
L — значение полинома в точке A.

```
INPUT "Укажите степень полинома ",N
DIM X(N), Y(N)
FOR i=0 TO N
      PRINT "Укажите значения X и Y в "; I;" узле";
      INPUT " ",X(i), Y(i)
NEXT i
INPUT "Укажите значение аргумента  X ",A
REM подпрограмма интерполяции по Ньютону
L=Y(0): S=1
FOR I=N TO 1 STEP -1
      I1=N-I
      FOR K=0 TO I-1
            Y(K)=(Y(K+1)-Y(K))/(X(K+1+I1)-X(K))
      NEXT K
      S=S*(A-X(I1)): L1=L
      L=L+Y(0)*S
NEXT I
PRINT "Значение функции";L
END
```

9.4.6. Процедуры и функции пользователя

Внутренние и внешние функции

Если некоторую функцию необходимо вычислять в программе многократно, то целесообразно определить ее как функцию пользователя и использовать в дальнейшем так же, как и стандартную функцию языка программирования. Для этого язык QBASIC предоставляет возможность использовать внутреннюю функцию или внешнюю функцию. Внутренняя функция может исполь-

зоваться только в текущей программе, а внешняя функция может использоваться как в текущей, так и в любой другой программе.

Внутренняя функция объявляется оператором DEF FN. При этом может быть два варианта использования данной функции: однострочная и многострочная.

Формат однострочной функции DEF FN:

DEF FNnn...nt(x_1,x_2, ...,x_m)=<выражение>

Здесь DEF – оператор; FN –стандартное имя функции; nn...n – расширение имени функции; t – тип переменной (результата вычисления функции); x_i – формальные параметры. Имя функции может содержать до 40 латинских символов. Число формальных параметров может быть шестнадцать.

Пример 9.4.17. Вычислить путь, пройденный автомобилем, если известны: начальный путь ***S0***, скорость ***v***, ускорение – ***a***, и время движения автомобиля –***t***:

```
DEF FNput(S0,v,a,t)=S0+v*t+a*t^2/2
```

Формат многострочной функции DEF:

```
DEF FNnn...nt(x1,x2, ...,xm)
   ...
  y=<выражение>
  FNnn...nt=y
END DEF
```

Строка FNnn...nt=y присваивает результат вычисления переменной с именем, равном имени функции.

Пример 9.4.18. Вычислить факториал числа N

```
DEF FNfactorial#(N)
    F=1
    FOR Fact =1 TO N
        F=F*Fact
    NEXT Fact
    FNfactorial#=F
END DEF
```

Используются функции пользователя так же, как и стандартные функции языка Бейсик.

```
REM вычисление факториала
...
Input "Введите M ",M
y= FNfactorial#(M): PRINT "Факториал ";M;"равен"; y
```

или

```
PRINT "Факториал";M;"равен"; FNfactorial#(M)
```

Функция пользователя должна быть объявлена в начале программы, до ее использования.

Внешние функции описываются следующим образом:

```
FUNCTION <имя>[(формальные параметры)] [STATIC]
   <тело программы>
   <имя>=<результат>
END FUNCTION
```

Формат внешней функции аналогичен формату внутренней многострочной функции. Имя функции может начинаться с любых символов.

Используется внешняя функция так же, как и внутренняя. Однако, при использовании внешней функции есть особенности. Одна из них состоит в том, что функция должна быть объявлена в вызывающей программе:

DECLARE FUNCTION <имя>(параметры) [STATIC]

Другая особенность использования внешней функции связана с использованием глобальных переменных. В данном пособии этот вопрос не рассматривается.

Заголовок внешней функции может включать описание типов аргументов, среди которых, в отличие от внутренних функций могут присутствовать и имена массивов, сопровождаемые пустыми скобками, например V(). Указатель STATIC употребляется в тех случаях, когда необходимо сохранить значения локальных переменных между двумя последовательными обращениями к функции.

Подпрограммы

При необходимости многократного использования одних и тех же вычислительных процедур они могут быть оформлены в виде подпрограмм. Подпрограммы, так же как и функции, могут быть внутренними и внешними.

Внутренние подпрограммы должны быть согласованы по используемым переменным с главной программой. Вызов внутренних подпрограмм может быть организован с помощью операторов GOTO и GOSUB.

Примеры организации вызова подпрограмм с использованием операторов GOTO и GOSUB приведены на рисунке 9.4.12.

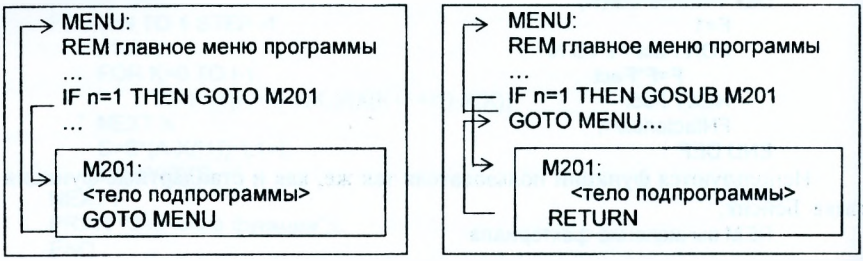


Рис.9.4.12. Организация вызова подпрограмм

Оператор *GOTO M* обеспечивает безусловный переход к указанной метке. В качестве метки может использоваться любой набор букв и цифр заканчивающийся двоеточием, например, *MENU:*

Оператор *GOSUB M* вызывает подпрограмму по указанной метке. Подпрограмма завершается оператором *RETURN*. После выполнения подпрограммы управление передается на оператор, следующий за оператором, вызвавшим подпрограмму.

Внимание:

1. Если в подпрограмму вошли по оператору *GOSUB*, то выход из подпрограммы происходит по оператору *RETURN* на метку, следующую за оператором, вызвавшим подпрограмму. Из подпрограммы можно выйти в этом случае и с помощью оператора *GOTO* на любую метку.

2. Если в подпрограмму вошли по оператору *GOTO*, то выйти из подпрограммы можно только по оператору *GOTO*. Если в этом случае программа выходит на оператор *RETURN*, то программа *ЗАВИСАЕТ*.

Внешние подпрограммы оформляются с помощью оператора *SUB*.

Формат оператора:

```
SUB <имя> [список формальных параметров]
    <тело программы>
END SUB
```

По формату внешняя подпрограмма ничем не отличается от внешних функций. В качестве формальных параметров могут использоваться простые переменные и имена массивов любой размерности. Для их выделения в списке формальных параметров вслед за идентификатором массива ставятся круглые скобки, например:

```
SUB <имя_подпрограммы >(V1() AS TYPE1 [ , V2 () AS TYPE2]) [STATIC]
```

В данном примере для массивов V1 и V2 явно указан тип массивов, назначение опции STATIC такое же, как и при описании функций пользователя.

Вызов подпрограммы осуществляется оператором CALL:

‹имя программы› (список параметров)

Подпрограмма должна быть описана в вызывающей программе:

```
DECLARE SUB <имя_подпрограммы>(список параметров)
```

В подпрограмме не принято описывать массивы, указанные в числе фактических параметров, поэтому, чтобы узнать размерности массивов применяют функции LBOUND(V,k) и UBOUND(V,k). Эти функции выдают, соответственно, значения нижней и верхней границ индекса массивов по заданному измерению k в массиве с именем V.

Пример 9.4.19. Подпрограмма суммирования элементов двухмерного массива:

```
SUB SummArray(A() As REAL, S As REAL)
    N1=LBOUND(A,1): M1=UBOUND(A,1)
    N2=LBOUND(A,2): M2=UBOUND(A,2)
    S=0
    FOR I=N1 TO M1
        FOR J=N2 TO M2
            S=S+A(I,J)
        NEXT J
    NEXT I
END SUB
```

Возврат из внешней подпрограммы всегда происходит на оператор, следующий за оператором CALL.

9.4.7. Графические операторы

Режимы работы мониторов

Видеомониторы (дисплей) предназначены для вывода на экран алфавитно-цифровой и графической информации.

В текстовом режиме на экран дисплея можно выводить только символьную информацию и простые рисунки, составленные из символов псевдографики, имеющихся в кодовой таблице ПК.

Графический режим используется для формирования сложных рисунков, а также символов, отличающихся от стандартных по размерам и форме.

Различие между мониторами персонального компьютера определяется такими характеристиками как разрешающая способность, количество воспроизводимых цветов. В современных компьютерах преимущественно используются растровые дисплеи, у которых изображение рисуется с помощью электронного луча, пробегающего по строкам начиная с верхнего левого угла слева направо и сверху вниз, как в обычном телевизоре.

Разрешающая способность монитора характеризуется числом знакомест (в текстовом режиме) или числом точек (в графическом режиме) умещающихся на экране по горизонтали и вертикали. По этому признаку различают мониторы с низкой, средней и высокой разрешающей способностью (табл. 9.4.3). Размер символа в графическом режиме определяется также числом точек по горизонтали и по вертикали (мини-растр) и в зависимости от используемого адаптера может иметь размеры 8x8, 8x14, 8x16, 9x14 или 9x16 точек.

Мониторы могут быть черно-белые (монохромные) и цветные. Монохромные мониторы используют в графическом режиме от 16 до 64 градаций черно-белого, цветные мониторы воспроизводят от 16 до 256 цветов.

Таблица 9.4.3

Разрешающая способность мониторов

Разрешение	*Режим работы*	
	Текстовый	*Графический*
Низкое	25 x 20	160 x 200
Среднее	25 x 40	320 x 200
Высокое	25 x 80	640 x 200
Повышенное	53 x 80	800 x 600 и более

Среди цветных мониторов получили наибольшее распространение RGB – мониторы, в которых для формирования цвета используются три основных цвета: красный – R, зеленый – G и синий – B.

Для хранения информации о каждом символе в текстовом режиме требуется два байта (рис. 9.4.13):

– первый байт содержит код символа;

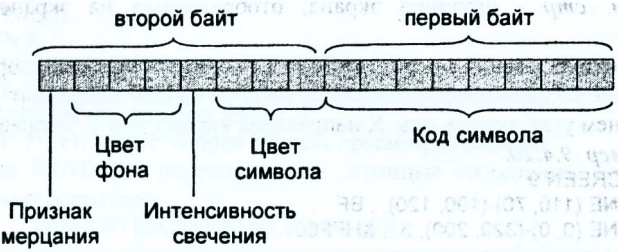


Рис. 9.4.13. Кодирование текстовой информации

– второй байт – атрибуты цвета : цвет символа – первый и третий биты, признак интенсивности свечения – четвертый бит, цвет заднего плана – пятый, шестой и седьмой биты и признак мерцания – восьмой бит.

При таком кодировании имеется возможность получить не более 16 цветов.

При работе с экраном выделяют четыре области: передний план, задний план (фон) и окантовка (рис. 9.4.14).

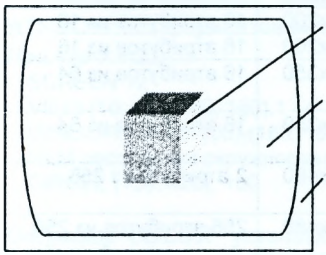


Рис. 9.4.14. Области экрана монитора доступные для управления цветом

Для хранения информации о каждой точке в графическом режиме требуется один байт, структура которого соответствует структуре второго байта в текстовом режиме. В современных мониторах для кодирования цвета точки используется три байта, что позволяет получить 256 x 256 x 256 цветов и оттенков.

Установочные операторы

К установочным операторам относятся операторы, которые управляют переводом экрана монитора в графический режим, устанавливают размеры окон, масштаб представления изображения и цвет экрана: SCREEN, VIEW, WINDOW, COLOR.

Оператор SCREEN устанавливает режим экрана. Формат оператора:

SCREEN режим [,[перекл_цвета] [,[актив_стр] [,видим_стр]]]

Здесь *режим* - целочисленная константа, устанавливает режим экрана. Может принимать значения от 0 до 13 (табл 9.4.4);

перекл._цвета - целочисленная константа, определяет цветность экрана. Если значение параметра равно 0, то монохромное изображение, если значение параметра больше нуля, то цветное изображение;

актив._стр - страница экрана, в которую записывается вывод текста или графики;

видим. стр - страница экрана, отображаемая на экране в данный момент.

При использовании оператора SCREEN ориентация осей координат соответствует 4-й координатной плоскости, то есть начало координат находится в левом верхнем углу экрана, ось X направлена вправо, ось Y направлена вниз.

Пример 9.4.20.

```
SCREEN 9
LINE (110, 70)-(190, 120), , BF
LINE (0, 0)-(320, 200), 3, , &HFF00
```

Таблица 9.4.4

Режимы работы монитора, адаптер VGA

Код режима	*Режим*	*Разрешающая способность*		*Цвет (число цветовых атрибутов из числа поддерживаемых цветов)*
		Текст	*Графика*	
0	текстовый	40x25, 9x16		2 атрибута из 16
1	графический	40x25	320x200	4 атрибута из 16
2	графический	80x25	720x400	2 атрибутов из 16
7	графический	40x25	320x200	16 атрибутов из 16
8	графический	80x25	640x200	16 атрибутов из 16
9	графический	80x25 или 80x43	640x350	16 атрибутов из 64
10	монохромный	80x25 или 80x43	640x350	16 атрибутов из 64
11	графический	80x30 или 80x60	640x480	2 атрибута из 256
12	графический	80x30 или 80x60	640x480	256 атрибутов из 256
13	графический	40x25	320x200	256 атрибутов из 256

Оператор VIEW определяет размер и положение области просмотра, где графика может быть выведена на экран, можно использовать для создания на экране нескольких окон просмотра. Формат оператора:

VIEW [[SCREEN] (x1,y1)-(x2,y2) [,[цвет] [,граница]]]

где опция

SCREEN указывает, что координаты задаются относительно экрана, а не области просмотра;

(x1,y1)-(x2,y2) - координаты диагонали противоположных углов области просмотра;

цвет - атрибут цвета, устанавливающий заполняющий цвет области просмотра (цвет фона);

граница - атрибут цвета, устанавливающий цвет границы области просмотра (бордюра).

Если все аргументы опущены, область просмотра - весь экран.

Допустимые атрибуты цвета зависят от используемого графического адаптера и режима экрана, установленного последним оператором SCREEN.

Пример 9.4.21:

```
SCREEN 9
VIEW (10, 10)-(300, 180), , 1
LOCATE 1, 11: PRINT " Первая область просмотра графики";
VIEW SCREEN (80, 80)-(200, 125),3, 5
LOCATE 11, 11: PRINT "Вторая область просмотра графики";
```

Оператор WINDOW переопределяет экранные координаты на математические. Формат оператора:

WINDOW [[SCREEN] (x1,y1)-(x2,y2)]

Данный оператор удобно использовать для установки требуемого *масштаба* изображения. Оператор изменяет ориентацию оси Y. Ось Y направлена вверх.

Опция *SCREEN* инвертирует обычное направление декартовых координат так, что ось Y направлена на экране вниз, то есть возвращает к координатам, установленным оператором SCREEN;

(x1,y1) - логические координаты, соответствующие координатам верхнего левого угла области просмотра;

(x2,y2) - логические координаты, соответствующие координатам нижнего правого угла области просмотра.

WINDOW без аргументов выключает логическую систему координат.

Пример 9.4.22.

```
SCREEN 12
VIEW (10, 10)-(300, 180),1 , 4
WINDOW (-160, -100)-(160, 100)
REM построение окружности в центре экрана радиусом 100
CIRCLE (150, 120), 100
```

Оператор COLOR устанавливает цвет изображения и других областей экрана. Действие оператора COLOR зависит от режима работы монитора установленного оператором SCREEN.

```
COLOR [передний план] [,фон] [, бордюр]   - для режима 0
COLOR [фон] [, палитра]                   - для режима 1
COLOR [передний план] [,фон]              - для режима 7,8,9,10
COLOR [передний план]                     - для режима 11-13
```

Здесь *палитра* – набор цветов.

Оператор *COLOR* без параметров устанавливает цвета областей экрана принятые по умолчанию.

Операторы для изображения рисунков

Операторы PSET и PRESET рисуют точку на экране.

```
PSET [STEP] (x,y) [,color]
PRESET [STEP] (x,y) [,color]
```

Опция *STEP* указывает, что x и y заданы относительно текущего графического положения курсора;

(x,y) - координаты точки устанавливаемой на экране;

color - атрибут цвета, устанавливаемый для точки. Если *color* опущен, то оператор *PSET* использует текущий цвет переднего плана, а *PRESET* использует текущий цвет фона (то есть точка не отображается на экране).

Доступные атрибуты цвета зависят от используемого графического видеоадаптера и режима экрана. Значения координат зависят от графического видеоадаптера, режима экрана и последних установок в операторах *VIEW* и *WINDOW*. При совместном использовании оператор *PSET* вычерчивает точку на экране, а оператор *PRESET* стирает точку.

Пример 9.4.23.

```
SCREEN 9
FOR i = 0 TO 320
    PSET (i, 100)
    FOR delay = 1 TO 100: NEXT delay     ' пауза
    PRESET (i, 100)
NEXT i
```

Оператор LINE рисует на экране линию или прямоугольник. Формат оператора:

```
LINE [[STEP](x1,y1)]-[STEP](x2,y2) [,[цвет] [,[B | BF] [,стиль]]]
```

Опция *STEP* - указывает, что координаты задаются относительно текущего графического положения курсора;

(x1,y1), (x2,y2) - координаты начала и конца линии на экране;

цвет - атрибут цвета, устанавливающий цвет линии или прямоугольника. Допустимые атрибуты цвета зависят от графического адаптера и режима экрана, установленного последним оператором *SCREEN*;

B - рисует прямоугольник вместо линии;

BF - рисует заполненный прямоугольник;

стиль - устанавливает стиль линии, представляет собой четырехразрядный шестнадцатеричный код. Если перевести шестнадцатеричный код в двоичный, то биты этого кода определяют, будут ли рисоваться точки на экране. Используется для изображения прерывистых и пунктирных линий. Например: 1110 1110 1110 1110 - двоичный код пунктирной линии. Код той же линии в шестнадцатеричном коде будет иметь вид &HEEEE; 1111 1111 1100 1100 – двоичный код штрих пунктирной линии. Код той же линии в шестнадцатеричном коде будет иметь вид &HFFCC. Символы & и H перед кодом числа являются признаком шестнадцатеричного кода.

Оператор CIRCLE рисует на экране окружности и эллипсы. Формат оператора.

```
CIRCLE [STEP] (x,y),радиус[,цвет] [,[-]старт] [,[-][конец] [,сжатие]
```

Опции

STEP - указывает, что координаты задаются по отношению к текущей графической позиции курсора;

(x,y) - координаты центра окружности или эллипса;

радиус - радиус окружности или эллипса в единицах текущей системы координат, определенной последними операторами *SCREEN, VIEW* и *WINDOW*;

цвет - атрибут цвета, устанавливающий цвет окружности.

старт - начальный угол дуги в радианах;

конец - конечный угол дуги в радианах. Знак "-" перед начальным или/и конечным значением угла позволяет соединить начало или конец дуги с центром окружности;

сжатие - отношение длины оси У к длине оси Х, используемое при изображении эллипсов.

Для перевода градусов в радианы используется формула:

(угол_в_градусах) * PI / 180

Пример 9.4.24:

```
SCREEN 12
PI=4*ATN(1) :      'Константа, вычисленная через арктангенс
REM окружность радиусом 200
CIRCLE (320, 100), 200
REM  сектор эллипса, начальный угол равен  30 градусам,
REM  конечный угол равен 120 градусам
CIRCLE STEP (200,100), 100,5,-30/180*PI,-120/180*PI,0.5
```

Оператор DRAW служит для вычерчивания фигур. Формат оператора:

DRAW строка_команд$

где *строка_команд$* - строковое выражение, содержащее одну или несколько команд оператора *DRAW*.

Команды оператора *DRAW* можно разделить на две группы: команды построения изображением и команды управления изображением.

Команды построения.

D[n] - перемещает курсор вниз на n единиц.

E[n] - перемещает курсор вправо-вверх на n единиц.

F[n] - перемещает курсор вправо-вниз на n единиц.

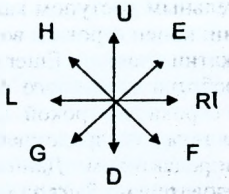
G[n] - перемещает курсор влево-вниз на n единиц.

H[n] - перемещает курсор влево-вверх на n единиц.

L[n] - перемещает курсор влево на n единиц.

R[n] - перемещает курсор вправо на n единиц.

U[n] - перемещает курсор вверх на n единиц.



[b] - префикс. Необязательная приставка, перемещение курсора без вычерчивания линии.

[n] - префикс. Необязательная приставка, нарисовать линию и возвратить курсор в первоначальную позицию.

M [{+|-}]x,y - перемещает курсор в точку x, y. Если перед x стоит знак + или - , то курсор перемещается относительно текущей точки.

Команды управления.

Команды управления должны всегда предшествовать командам построения.

An - поворачивает объект на n * 90 градусов (n может быть 0, 1, 2 или 3).

Cn - устанавливает рисующий цвет (n- атрибут цвета).

Pn1,n2 - устанавливает цвет заполнения и цвет границы объекта (n1 - атрибут цвета заполнения, n2 - атрибут цвета границы).

Sn - определяет масштаб рисунка, устанавливает единицу длины перемещения курсора. По умолчанию n равно 4, что эквивалентно 1 точке растра. При n> 4 уменьшение масштаба, при n <4 – увеличение масштаба, n может быть только целым числом.

TAn - поворачивает угол на n градусов (от -360 до 360). Угол задается в градусах.

Если в командах изображения линии и перемещения курсора опущен параметр *n*, то курсор перемещается на 1 единицу.

Для выполнения подстроки команд *DRAW* из строки команд *DRAW*, используйте команду "X":

```
DRAW "X"+ VARPTR$(строка_команд$)
```

Пример 9.4.25:

```
SCREEN 9
Triangle$ = "F60 L120 E60"
DRAW "C2 X" + VARPTR$(Triangle$)
DRAW "B030 P1, 2 C3 M-30,-30"
```

9.4.8. Файлы данных

Понятие о файлах данных

В процессе разработки программ часто возникает необходимость в хранении и обработке сохраненной информации. Эта информация может быть самой разнообразной: исходные данные для решения задач, результаты вычислений, списки и так далее. Для хранения такой информации могут использоваться файлы баз данных. Файлы данных текстовые. В зависимости от организации данных на дисках или других машинных носителях текстовые файлы делятся на файлы с *последовательным доступом*, *файлы с прямым доступом* и *двоичные файлы*.

Текстовые файлы с последовательным доступом (файлы последовательного доступа) не имеют какой-либо структуры. Структура этих файлов определяется самой считывающей программой. В текстовых файлах с последовательным доступом каждая строка заканчивается двумя специальными символами: конец строки и возврат каретки, которые вводятся в текст программы при нажатии клавиши Enter на клавиатуре. Поэтому один из самых легких способов обработки текстового файла с последовательным доступом состоит в чтении его строка за строкой. Создание текстовых файлов с последовательным доступом также не представляет большого труда. Его можно создать любым текстовым редактором. Данные в файл последовательного доступа записываются последовательно байт за байтом. Чтобы проанализировать и выбрать нужную информацию, файл должен быть полностью прочитан. Это повышает требования к объему оперативной памяти и снижает скорость выполнения программы.

Текстовые файлы с прямым доступом (файлы прямого доступа) предназначены для чтения и записи текста или структурированных двоичных файлов с записями фиксированной длины. Они позволяют записывать и извлекать данные из файла по номеру записи. Это сокращает время на поиск и извлечение данных. Однако при этом имеет место неэффективное использование дискового пространства, так как длина каждого поля в записи должна быть заранее оговорена.

Двоичные файлы (бинарные) используются для чтения и записи произвольно структурированных данных. Бинарные файлы - это, строго говоря, не новый тип файлов, а один из способов управления файлами любого типа. Методы работы с бинарными файлами позволяют считывать и изменять любой байт файла.

Для работы с файлами данных используются команды открытия файла, закрытия файла, записи и чтения данных из файла, а также ряд функций, облегчающих работу с файлами. Все эти команды традиционны для всех версий языка Basic.

Открытие файлов

Для открытия файлов служит команда ***Open***.

```
Open "спецификация_файла" For { тип файла} [Access{доступ}]
    [Lock{блокировка}] As [#] N [Len=длина]
```

Опция ***"Спецификация_файла"***, как известно, позволяет указать диск, маршрут, имя и расширение имени файла. Например: R:/Prognoz/Ucheb/prognoz1.dan. Чтобы файл мог использоваться на компьютерах и с операционной системой MS DOS, имя файла и его расширение должны формироваться по правилам операционной системы MS DOS. То есть в имени файла и расширении имени файла могут использоваться только латинские символы и цифры, имя файла должно начинаться с буквы, длина имени файла не должна превышать 8 символов, а расширение имени файла – четырех символов, включая точку. В имени файла не допускается использование точек и пробелов. Спецификация файла заключается в кавычки.

Опция ***For*** определяет тип файла. Тип файла указывает на его структуру и способ использования и может принимать следующие значения:

Input – файл последовательного доступа, открыт для чтения;

Output – файл последовательного доступа, открыт для записи;

Append - файл последовательного доступа, открыт для добавления данных;

Bynary – двоичный файл открыт для записи и чтения данных:

Random – файл прямого доступа открыт для записи и чтения данных.

Опция ***Access*** определяет права доступа к данным при работе в сетях ЭВМ. Она может иметь три значения:

Read – разрешено чтение данных из файла;

Write – разрешена запись данных в файл;

Read Write - разрешено чтение и запись данных. Этот режим доступа используется по умолчанию.

Опция ***Lock***. Так как режим чтения-записи, обычно, предназначен для работы с файлами, которые могут использоваться многими пользователями или приложениями, необходимо обеспечить целостность данных при коллективном использовании. Для этой цели используется параметр блокировка, который может принимать следующие значения:

Shared – файл может использоваться всеми процессами для считывания и записи данных;

LockRead – запрет чтения. Никакой другой процесс не может считывать данные из файла. Этот параметр можно установить, если в данный момент никакой другой процесс не выполняет операцию чтения.

LockWrite – запрет записи. Никакой другой процесс не может записывать данные в файл. Данный параметр можно установить, если в текущий момент никакой другой процесс не выполняет операцию записи.

LockReadWrite – запрет записи, чтения данных. Этот параметр можно установить, если в данный момент никакой другой процесс не выполняет операцию записи, чтения.

Опция ***As*** – определяет номер канала. Знак # можно опустить. Номер канала может принимать значения от 1 до 255. Число одновременно открытых

каналов определяется ограничениями операционной системы, указанными в файле Config.sys.

Опция ***Len*** – используется только в файлах прямого доступа. Она устанавливает длину записи в байтах.

При *открытии* или, иными словами, *инициализации* файлов выполняются следующие операции:

- устанавливается связь между спецификацией файла и его программным номером. Поэтому во всех последующих операциях с данным файлом дается ссылка на номер канала, а не на спецификацию файла;
- закрепляется системный или программный буфер, используемый для реализации операторов ввода-вывода. Использование буфера уменьшает число обращений программы к диску, а, следовательно, повышается скорость записи-чтения данных;
- формируются начальные значения параметров, расположенных в так называемом блоке управления файлом.

Закрытие файлов

Для закрытия файлов используется команда ***Close***. Синтаксис команды:

Close [# <номер канала>]

Команда Close с параметром номера канала закрывает указанный канал. Команда Close без параметров закрывает все открытые файлы. Команда Close очищает буфер и дает указание операционной системе обновить таблицу размещения файлов [FAT].

Команды записи данных в файл и чтения информации из файлов данных зависят от типа файла.

Файлы последовательного доступа

Файлы последовательного доступа отличаются не только простотой организации данных, но и простотой управления ими.

Файл последовательного доступа используется, обычно, для работы с текстовой информацией, хотя ничто не мешает использовать их для работы с числами.

Работа с файлами последовательного доступа состоит из двух самостоятельных операций: создания файла и использования файла.

Создание файла последовательного доступа

Создание файла последовательного доступа можно представить следующей схемой:

```
Открытие файла ' (команда OPEN или APPEND с опцией OUTPUT)
    Запись данных в файл.
Закрытие файла ' (команда CLOSE)
```

При необходимости, файл последовательного доступа может быть создан или отредактирован любым текстовым редактором.

Использование файла последовательного доступа

При использовании файла последовательного доступа также реализуется простая схема:

```
Открытие файла ' (команда OPEN с опцией INPUT)
    Чтение данных из файла.
Закрытие файла ' (команда CLOSE)
```

Запись данных в файл последовательного доступа

Для записи данных в файл последовательного доступа используются команды **PRINT # и WRITE #.**

Формат команды PRINT # полностью соответствует формату команды PRINT, используемой для вывода данных на экран монитора. Синтаксис оператора WRITE:

WRITE # <номер канала>[,список выражений]

Выражения в списке могут быть числового или строкового типа и должны отделяться запятыми или точкой с запятой.

В отличие от оператора PRINT # оператор WRITE # вставляет между элементами списка запятые и выделяет строки двойными кавычками. Перед положительными числами пробела под знак числа не вставляется. После последнего выведенного элемента списка Basic завершает строку символами возврата каретки и перевода строки.

```
Print #1, Анна, Минск, 17 , 3.75
Write #1, "Анна", "Минск", 17 , 3.75
```

При работе с числами предпочтительнее использовать оператор Write #.

Чтение данных из файла последовательного доступа

Чтение данных из файла последовательного доступа осуществляется операторами *Input #, Line Input #* и функцией *Input$*.

Оператор ***Line Input #*** считывает из файла строку данных. Разделителем данных в файле в этом случае должен быть символ возврата каретки (вводится в строку текста автоматически при нажатии клавиши Enter). Строка данных не должна превышать 255 символов.

Оператор ***Input #*** имеет следующий синтаксис:

```
Input #, "текстовое сообщение",<список переменных>
```

Переменные в списке разделяются запятыми.

Функция ***Input$*** служит для вывода из файла на экран указанного числа символов, не отображаемых на экранс. Синтаксис функции Input$:

```
<символьная_переменная>= Input$ (n, #N),
```

где n – число символов, выделяемых из файла, # N – номер открытого канала файла последовательного доступа.

Пример 9.4.26. Создание файла последовательного доступа.

```
Open "R:Test.dan" For Output As #1
    A$ = "Минск – столица Республики Беларусь"
    B%=13875
    C!=7.58
    Print#1, A$, B%, Str$ (C!)
Close #1
```

Пример 9.4.27. Использование файла последовательного доступа

```
Open "R:Test.dan" For Input As #1
    Line  Input  #1,A$
    Print A$
Close
```

На экране будет следующая строка:

```
Минск – столица Республики Беларусь, 13875, 7.58
```

Здесь 13875 и 7.58 текст.

```
Open "R:Test.dan" For Input As #1
   Input #1, A$, B%, C$
   Print A$, B%, Val (C$)
Close #1
```

На экране будет строка следующего вида:

```
Минск – столица Республики Беларусь 13875  7.58
```

В данном примере 13875 и 7.58 – числа

Оператор Input # целесообразно использовать в сочетании с оператором Write #.

Файлы прямого доступа

Порядок работы с файлами прямого доступа значительно отличается от порядка работы с файлами последовательного доступа.

Открытие файла прямого доступа

Файл прямого доступа открывается одной командой как для чтения, так и для записи.

```
OPEN "имя_файла"  FOR RANDOM AS #1
```

Описание буфера данных

```
FIELD #1, N1  As V1$ [, N2 As V2$]
```

где *N1, N2* – длина поля, *As* – служебное слово, *V1$, V2$* - имена полей буфера данных.

Создание файла прямого доступа:

а) запись данных в буфер:

```
LSET  V1$ =<переменная символьного типа>
LSET  V2$ =<переменная символьного типа>
...
```

б) запись данных из буфера в файл

```
PUT #1, nom
```

где nom – номер записи, LSET – функция, прижимает текст к левому краю поля буфера. Другая функция – RSET – прижимает текст к правому краю поля буфера.

Использование файла прямого доступа:

а) чтение данных из файла в буфер данных:

```
GET #1, nom
```

б) чтение данных из буфера

<переменная1>=V1$: <переменная2>=V2$ …

В файле прямого доступа могут храниться только символьные переменные. Поэтому числовые переменные при записи в буфер должны быть преобразованы в символьные эквиваленты с помощью следующих функций:

MKI$(N) – преобразует целое число одинарной длины в двухбайтовый символьный эквивалент;

MKL$(N) – преобразует целое число двойной длины в четырехбайтовый символьный эквивалент;

MKS$(N) – преобразует вещественное число одинарной точности в четырехбайтовый символьный эквивалент;

MKD$(N) – преобразует вещественное число двойной точности в восьмибайтовый символьный эквивалент.

Обратное преобразование переменных буфера в числовые переменные соответствующего типа осуществляется с помощью функций *CVI(C), CVL(C), CVS(C), CVD(C).*

9.4.9. Типовые схемы алгоритмов

Разработка паспорта и меню программы

Паспорт и меню программы представляют собой текстовую информацию. Для вывода ее на экран используются операторы *PRINT, LOCATE*, функции *TAB(n), SPC(n).* Для построения рамок проще всего использовать оператор *LINE* в графическом режиме.

Пример 9.4.28. Паспорт программы

```
REM Паспорт программы
CLS:
SCREEN 9
LINE(40,10)-(600,190),,B:
REM Для получения на экране и при печати линий равной
REM толщины проводится двойная вертикальная линия
LINE(41,10)-(601,190),,B:
LOCATE 3,20:?"Министерство образования Республики Беларусь"
LINE(100,25)-(540,25)
LOCATE 5,15:?"  Брестский государственный технический университет"
LOCATE 6,12:?"  Кафедра информатики  и прикладной математики"
LOCATE 10,20:?"          КУРСОВАЯ  РАБОТА"
LOCATE 11,20:?" тема: Моделирование изображения детали"
LOCATE13,12:?"Программа позволяет строить аксонометрическое изображение"
LOCATE 14,12:?"детали, и ее проекции, изменять  масштаб  изображения, пере-"
LOCATE 15,12:?"мещать деталь и поворачивать изображение на  произвольный"
LOCATE 16,12:?"угол в режиме  диалога  с  пользователем. Угол  вводится в"
LOCATE 17,12:?"градусах"
LOCATE 19,25:?"Исполнитель:  Баценко Д.Л.  гр.Т-39"
LOCATE 20,25:?"Дата сдачи работы: 1.06.97"
LOCATE 23,35:?"Брест 1997"
```

Пример 9.4.29. Меню программы

```
MENU:  REM Меню программы
DEFSTR c
REM  переменные, имя которых начинается с символа "c" -  символьные
SCREEN 9
LINE (115,20)-(515,140),,b : REM Построение двойной рамки
LINE (119,22)-(511,138),,b
LOCATE 5,22:?"***  Г Л А В Н О Е   М Е Н Ю   ***"
LOCATE 10,22:?"1.Паспорт программы."
LOCATE 12,22:?"2.Проверка наличия корня."
LOCATE 14,22:?"3.Демонстрация процедуры вычисления."
LOCATE 16,22:?"4.Выход."
RETURN
```

Обработка пунктов меню

Пример 9.4.30. Выбор пункта меню

```
ONK: ' подпрограмма выбора пунктов меню (простейший вариант)
LOCATE 23,1: "Ваш выбор ": INPUT " ",nkl
RETURN
```

Здесь *Mn* - метка подпрограммы выполнения соответствующего пункта меню.

Для анализа номера выбранного пункта меню можно использовать три оператора: оператор IF, оператора ON GOTO или ON GOSUB, оператор SELECT CASE. Наиболее простой из перечисленных операторов - оператор ON GOTO/GOSUB. Наиболее удобные, с точки зрения структурного программирования, оператор IF, многострочный оператор IF и оператор SELECT CASE.

Для обработки пунктов меню можно использовать простой оператор IF:

IF <условие> THEN <оператор перехода>

IF <условие> THEN <оператор вызова подпрограммы>

Пример 9.4.31. Обработка пунктов меню оператором IF

```
OPM1:  REM подпрограмма обработки пунктов меню
```

```
GOSUB ONK : REM вызов подпрограммы выбора пунктов меню. ONK -метка
      1-й вариант                          2-й вариант
IF nkl=1 THEN  GOTO M1          |  IF nkl=1 THEN  GOSUB M1: GOSUB MENU
IF nkl=2 THEN  GOTO M2          |  IF nkl=2 THEN  GOSUB M2: GOSUB MENU
...                             |  ...
IF nkl=n THEN  GOTO Mn          |  IF nkl=n THEN  GOSUB Mn: GOSUB MENU

LOCATE 1,1: PRINT "Неправильный ввод. Введите пункт меню"
BEEP 1:    REM Выдача звукового сигнала
SLEEP 3:   REM Останов на 3 секунды и очистка строки сообщения
LOCATE 1,1: PRINT "                                                    "
GOTO OPM1: REM Возврат для ожидания нажатия клавиши"
END
```

При использовании многострочного IF получается хорошо читаемая структура программы:

Пример 9.4.32. Обработка пунктов меню многострочным IF

```
OPM1:  REM подпрограмма обработки пунктов меню
GOSUB ONK : REM вызов подпрограммы выбора пунктов меню
    IF nkl=1 THEN
        GOSUB M11
        ...
        GOSUB MENU
    ELSEIF nkl=2
        GOSUB M21
        ...
        GOSUB MENU
        ...
    ELSE
        LOCATE 1,1: PRINT "Неправильный ввод. Введите пункт меню"
        BEEP 1:    REM Выдача звукового сигнала
        SLEEP 3:   REM Останов на 3 секунды и очистка строки сообщения
        LOCATE 1,1: PRINT "                                                    "
        GOTO OPM1: REM Возврат для ожидания ввода пункта меню"
    END IF
```

Пример 9.4.33. Обработка пунктов меню оператором ON GOTO/GOSUB

```
OPM2:  REM подпрограмма обработки пунктов меню
GOSUB ONK : REM вызов подпрограммы выбора пунктов меню
          Вариант 1                          Вариант 2
ON nkl GOTO  M1,M2,...,Mn       |  ON nkl GOSUB M1,M2,...,Mn

LOCATE 1,1: PRINT "Неправильный ввод. Введите пункт меню"
BEEP 1:        REM Выдача звукового сигнала
SLEEP 3:       REM Останов на 3 секунды и очистка строки сообщения
LOCATE 1,1: PRINT "                                                    "
GOTO OPM2: REM Возврат для ожидания ввода пункта меню"
```

Пример 9.4.34. Обработка пунктов меню оператором SELECT CASE

Оператор SELECT CASE, так же как и многострочный IF позволяет сформировать хорошо читаемую программу.

```
    OPM3:  REM подпрограмма обработки пунктов меню
    GOSUB ONK : REM вызов подпрограммы выбора пунктов меню
    SELECT CASE nkl :  REM nkl - переключающее выражение
          CASE 1   :  REM 1 - условие выборки
                GOSUB M1   :  REM M1, M11 ..- подпрограммы, вызываемые
```

```
            GOSUB M11  :  REM при выполнении первого пункта меню
            GOSUB MENU
        CASE 2
            GOSUB M2
            GOSUB M21
              ...
        CASE  ELSE
            LOCATE 1,1: PRINT "Неправильный ввод. Введите пункт меню"
            BEEP 1:   REM Выдача звукового сигнала
            SLEEP 3:  REM Останов на 3 секунды и очистка строки сообщения
            LOCATE  1,1:  PRINT"                                          "
            GOTO OPM1: REM Возврат для ожидания ввода пунктов меню"
    END  SELECT
```

Пример 9.4.35. Подпрограмма выбора пунктов меню

```
ONK1:  REM Подпрограмма обработки нажатия клавиши
msg1=" Введите правильно номер требуемого пункта меню "
LOCATE 23,1: REM Перемещение курсора в 23 строку
c$=SPACE$(80): PRINT c$: REM Очистка 23 строки
a=(40-LEN(msg1))/2:       REM центрирование сообщения
LOCATE 23,10:?TAB(a);msg1 :REM Вывод сообщения в 23 строку
1000 IF INKEY$="" THEN GOTO 1000: REM Пустой цикл до нажатия
                    REM любой клавиши
GOSUB ONK
RETURN
```

Остановки в программе

В ряде случаев при выполнении программы необходимо задержать ее выполнение на некоторое время, например, чтобы просмотреть результаты промежуточных вычислений, вывести сообщение программы, необходимое пользователю, и так далее. Такие остановки можно реализовать с помощью подпрограмм.

Пример 9.4.36.

```
OSTANOWKA1: REM остановка до нажатия любой клавиши
LOCATE 23,1: PRINT "Для продолжения нажмите любую клавишу"
1000 IF INKEY$="" THEN GOTO 1000 : REM пустой цикл
RETURN
```

Пример 9.4.37.

```
OSTANOWKA2: REM остановка до нажатия любой клавиши
LOCATE 23,1: PRINT "Для продолжения нажмите любую клавишу"
A$=INPUT$(1) : REM ожидание ввода одного символа
RETURN
```

Пример 9.4.38.

```
OSTANOWKA3: REM остановка на заданное время
LOCATE 23,1 : REM комментарий отсутствует
SLEEP 5  :      REM остановка на 5 секунд
RETURN
```

Пример 9.4.39.

```
OSTANOWKA4: REM остановка на заданное время
LOCATE 23,1: PRINT " Ждите. Идут вычисления"
```

```
REM функция TIMER хранит количество секунд, прошедших после полуночи
T=TIMER: REM переменной t присвоено текущее время
1000 IF TIMER-T<5 THEN GOTO 1000: REM остановка на 5 секунд
RETURN
```

9.4.10. Обработка символьных переменных

Понятие о символьных переменных

Символьные переменные представляют собой цепочки любых символов заключенных в кавычки. Максимальная длина строки – 32767 символов, минимальная – ноль, пустая строка.

Для объявления символьных переменных используется суффикс – знак доллара "$". Тип символьных переменных может быть объявлен также с помощью оператора DEFSTR:

A$, C1$ - символьные переменные;

DEFSTR C, S - все переменные, имена которых начинаются с символов C и S, считаются переменными символьного типа.

Тип переменной объявленный оператором DEFSTR можно изменить с помощью суффикса $. Символьные переменные можно объединять, используя операцию конкатенации "&" или "+": C2$ = C$ & C1$ или C2$ = C$+C1$.

Ввод символьных переменных

Для ввода символьных переменных используются операторы *LET, DATA, INPUT, LINE INPUT*, функция *INPUT$*.

При использовании оператора *LET* символьная переменная заключается в кавычки:

```
LET A$ = "Брестская крепость – герой"
```

При присваивании символьных переменных с помощью оператора *DATA* или *INPUT* символьные переменные записываются без кавычек, если они не содержат пробелов или символов-разделителей, в противном случае они заключаются в кавычки:

```
DATA Сталинград, "Сталинградская битва"
INPUT "Введите имя летчика-космонавта", S$
```

При вводе команды RUN на экране появится запрос, в ответ на который вводится ответ без кавычек, если символьная переменная не содержит пробелов, и в кавычках, если имеются пробелы:

```
Введите имя летчика-космонавта? Гагарин
Введите имя летчика-космонавта? "Юрий Гагарин"
```

Оператор *LINE INPUT* позволяет ввести строку символов без кавычек. В строке символов допускаются пробелы и символы-разделители. Для окончания ввода необходимо нажать клавишу *Enter*.

Синтаксис оператора LINE INPUT аналогичен синтаксису оператора INPUT.

Функция *INPUT$* позволяет ввести указанное количество символов с клавиатуры или из файла без отображения на экране.

Синтаксис функции: B$ = INPUT$(n [[,#] номер файла])

Данную функцию можно использовать для ввода пароля. Эту функцию можно использовать также для организации остановок в программе: B$=INPUT$(1). В последнем случае, когда программа встречает функцию

INPUT$(1), она приостанавливает работу и ожидает ввода символа, для продолжения выполнения программы достаточно нажать любую клавишу.

Вывод символьных переменных

Для вывода символьных переменных на экран используется оператор PRINT, PRINT USING, а на печать - LPRINT, LPRINT USING.

Синтаксис оператора PRINT при выводе символьных переменных ничем не отличается от его синтаксиса при выводе числовых данных. При выводе символьных констант они заключаются в кавычки:

```
PRINT "Чем отличается герой нашего времени от Лермонтовского " "Героя нашего времени""?"
```

Оператор LPRINT предназначен для вывода информации на печать и имеет тот же синтаксис, что и оператор PRINT.

Операторы PRINT USING и LPRINT USING служат для вывода информации на экран с форматированием. Синтаксис операторов PRINT USING и LPRINT USING одинаков:

```
PRINT USING "шаблон, <список переменных>"
```

В качестве шаблона используются следующие маски:

\ \ - (символы пробела, заключенные между символами обратный слеш) – резервирует место для *n* + два символов, где *n* – число пробелов;

! – выводит один первый символ из символьной переменной.

Функции для обработки символьных переменных

QBASIC имеет значительное число функций для работы с символьными переменными.

Примечание. В данном разделе все символьные переменные будут обозначены символом C, а числовые переменные символом N.

По назначению функции для работы с символьными переменными можно разделить на ряд групп.

Обмен данными: SWAP. Синтаксис функции SWAP C1, C2

Перевод символов таблицы ASCII в число и чисел в код ASCII: ASC, CHR$.

Синтаксис функции ASC: ASC(C).

Функция ASC выделяет из строки символов первый символ и возвращает ASCII код числа. Например, ASC(Азбука) – результат число 128 – код русской буквы А.

Синтаксис функции CHR$: CHR$(N)

Функция CHR$ возвращает символ, соответствующий коду числа в таблице ASCII. Например, CHR$(129) – результат символ Б.

Перевод чисел из машинного представления в строку символов и из символьного представления в числовое: STR$ и VAL.

Синтаксис функции STR$: STR$(N).

Функция STR$ переводит число в строку символов.

Синтаксис функции VAL: VAL(C).

Функция VAL переводит строку символов в число.

Перевод числового аргумента любого типа в строки символов: BIN$, OST$, HEX$:

– функция BIN$(N) переводит число в двоичный код;

– функция OST$(N) переводит число в восьмеричный код;
– функция HEX$(N) переводит число в шестнадцатеричный код.

Замена числовых данных их символьными эквивалентами и преобразование символьных эквивалентов в числовые данные: MKI$, MKL$, MKS$, MKD$, CVI, CVL, CVS, CVD. При переводе чисел в символьные эквиваленты программа добавляет необходимое число байтов и заключает содержание переменной в кавычки.

Функция MKI$(N) переводит числовую переменную целого типа одинарной длины в символьный двухбайтовый эквивалент.

Функция MKL$(N) переводит числовую переменную целого типа удвоенной длины в символьный четырехбайтовый эквивалент.

Функция MKS$(N) переводит вещественную переменную одинарной точности в символьный четырехбайтовый эквивалент.

Функция MKD$(N) переводит вещественную переменную удвоенной точности в символьный восьмибайтовый эквивалент.

Функции CVI(C), CVL(C), CVS(C) и CVD(C) выполняют действия, обратные действию функций MKI$, MKL$, MKS$, MKD$, то есть переводят символьные эквиваленты числовых переменных в числовые переменные соответствующего типа.

Функции для обработки строк: LEN, LEFT$, RIGHT$, MID$, LTRIM$, RTRIM$, INSTR; LCASE$, UCASE$.

Функция *LEN(C)* возвращает длину строки.

Функции *LEFT$(C,N), RIGHT$(C,N)* – выделяют из строки символов N символов слева и справа, соответственно.

Функция *MID$(C,N1,N2)* – выделяет из строки символов N2 символов с позиции N1. Например:

```
C1 = MID$("Шумит, гудит зеленый лес",8,5)
PRINT C1
RUN
гудит
```

Функция MID$ выделяет из строки "Шумит, гудит зеленый лес" 5 символов с восьмой позиции - слово "гудит".

Функция MID$ может использоваться и для замены части текста. Синтаксис функции при замене текста:

```
MID$(C, N1, N2) = C1
```

В данном случае в строке С заменяется N2 символов строкой C1, начиная с позиции N1. Если длина строки C1 больше N2, то лишние символы отбрасываются. Если длина строки C1 меньше N2, то недостающие символы заполняются пробелами. Длина строки С при этом не изменяется. Результат замены сохраняется в строке С. Например:

```
C$="Отражается месяц в пруду"
MID$(C$, 12, 5) = "башня"
PRINT C$
RUN
Отражается башня в пруду
```

Функции *LTRIM$(C) и RTRIM$(C)* отбрасывают лидирующие и хвостовые пробелы, соответственно.

Функция *LCASE$(C)* переводит прописные буквы в строчные.

Функция *UCASE$(C)* переводит строчные буквы в прописные.

Функция *INSTR([I,]C1, C2)* осуществляет поиск первого вхождения строки символов С2 в строку символов С1 и возвращает номер позиции, с которой строка символов С2 входит в строку символов С1. Если вхождение не найдено, то возвра-

щается значение 0. Поиск может осуществляться с начала строки или с позиции I, если I отсутствует, то поиск начинается с первой позиции.

```
C$ = "Светит месяц, светит ясный"
C1$ = "месяц"
n = INSTR(1,C$, C1$)
PRINT n
RUN
 8
```

Функция INSTR(I,C,C1) осуществляет поиск первого вхождения искомой строки (C2), в строку C1:

```
C$ = "светит месяц, светит ясный"
C1$ = "светит"
n = INSTR(C$, C1$)
PRINT n
RUN
 1
```

Для поиска других вхождений необходимо организовать цикл.

Функции генерирования строк символов: SPACE$, STRING$

Функция *SPACE$(n)* генерирует строку из *n* пробелов.

Функция имеет два формата.

В первом формате: *STRING$(n,C)*, - функция выделяет из строки С первый символ и генерирует строку, содержащую n этих символов.

Во втором формате: *STRING$(n,Cod)*, - функция генерирует строку из n символов, определяемых кодом Cod. Здесь Cod – код кодовой таблицы ASCII. Cod может принимать значения от 31 до 255.

Типовые процедуры обработки символьных переменных

Строка символов рассматривается программой как одномерный массив. Символы можно сортировать и сравнивать так же как и числа. Операции отношения над текстовыми операндами базируются на посимвольном сравнении их числовых кодов. В кодовой таблице ASCII, например, цифры имеют меньшие номера, чем буквы. Латинские буквы имеет меньшие номера, чем символы русского алфавита. Прописные буквы имеют меньшие номера, чем строчные буквы соответствующего алфавита.

При сравнении двух строк символов программа сравнивает их посимвольно. Если строки разной длины, то строка с большим числом символов считается большей, если остальные символы совпадают. Если символы в строках разные, то при сравнении учитываются результаты сравнения по первым n символам. Где n число символов в короткой строке.

Рассмотрим некоторые типовые процедуры обработки символьных переменных.

Задача 9.4.8. Выделить i –й символ из текста.

Задача решается с помощью функции MID$:

```
C1$ = MID$(C, i, 1)
```

Задача 9.4.9. Найти позицию, в которой располагается заданный символ или цепочка символов.

Задача решается с помощью функции INSTR:

```
n = INSTR(C1, C2)
```

Задача 9.4.10. Удалить символ или цепочку символов.

Решение.

- найти позицию, с которой заданный символ или цепочка символов входит в строку;
- выделить часть строки слева от символа;
- выделить часть строки справа от символа;
- объединить полученные строки.

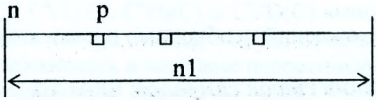
Задача 9.4.11. Вставить текст между i – м и i+1 – м символами.

- разделить строку на две подстроки в месте нахождения заданного символа;
- прибавить к левой части строки вставляемый текст и пробел;
- прибавить к полученной строке правую часть строки.

Задача 9.4.12. Разделить текст на строки, если ограничитель спецсимвол.

Решение.

1. Определить длину текста;
2. Найти первый спецсимвол (p).



3. Если p=0, то напечатать текст от n до n1 и завершить программу; иначе напечатать текст от n до p.
4. Запомнить позицию пробела: n = p + 1 и повторить операции по п. 2- 4

```
CLS
c$ = "во поле береза стояла"
c1$ = " "
n = 1  ' текущее значение переменной цикла
n1 = LEN(c$)
m:
IF n < n1 THEN
    p = INSTR(n, c$, c1$)        ' p - позиция спецсимвола (пробела)
    IF p = 0 THEN PRINT MID$(c$, n, n1 - (n - 1)): END
    PRINT MID$(c$, n, p - n)
    n = p + 1
    GOTO m
END IF
END
```

Задача 9.4.13. Разделить текст на строки по пробелам при заданной длине строки.

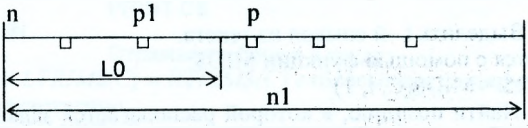
Решение. Данная задача является дальнейшим развитием задачи 9.4.12.

Введем переменные:

n – начало строки (подстроки);

n1 – длина строки символов;

p - текущая позиция спецсимвола (пробела);



p1 – предыдущая позиция спецсимвола;

L0 - заданная длина строки

Если длина выделяемой подстроки p-n больше L0, тогда на печать будет выведена строка p1-n.

```
CLS
```

```
DEFSTR C
c1 = "у лукоморья дуб  зеленый, златая цепь на дубе том"
c2 = " "
lc = 14
'Line input "Введите строку текста";c1
'input "Введите символ-разделитель"c2
'Input "Введите длину строки",lc
n = 1: REM начальное положение указателя
p = 1: REM текущее значение указателя
p1 = 0: REM предыдущее положение указателя
IF LEN(c1) < lc THEN PRINT c1: GOTO m2
m1:
IF p - n < lc THEN
     p1 = p
     p = INSTR(p1 + 1, c1, c2)
    GOTO m1
ELSE
     p2 = p1 - n
     c4 = MID$(c1, n, p2)
     PRINT c4
     n = p1 + 1
     p = p1
     c3 = LEFT$(c1, p1)
     '?c3
     p3 = (LEN(c1) - LEN(c3))
     IF (LEN(c1) - LEN(c3)) < lc THEN  PRINT RIGHT$(c1, p3): GOTO m2
     GOTO m1
 END IF
m2:
END
```

Задача 9.4.14. Сравнить символы или цепочки символов

Примеры: "a"<"b"; "ab"="ab"; "ac">"ab"; "abc"<"abd", но "abf">"abd"

"abc"<"abcd", но "fbc">"abcd"

Задача 9.4.15. Выделить слово из текста.

Решение.

- вычислить длину слова (функция LEN);
- найти позицию вхождения слова в строку символов (функция INSTR);
- выделить слово (функция MID$).

Задача 9.4.16. Определить, является буква гласной или согласной.

- ввести строку текста;
- ввести строку символов, содержащую только гласные Cg$;
- выделить символ из текста (функция MID$(C,n,1));
- проверить принадлежит ли символ строке гласных Cg$ (функция INSTR(C,C1)). Если принадлежит, значит символ -- гласная.

Контрольные вопросы и задания

1. Провести классификацию прикладного программного обеспечения.
2. В чем отличие библиотеки прикладных программ от пакетов прикладных программ?
3. В чем особенности интегрированных пакетов?

4. Что такое табулирование функций? Как оно осуществляется?

5. Поясните порядок вычисления многочлена по схеме Горнера.

6. В чем различие между аналитическими и численными методами решения уравнений?

7. Что такое отделение корней уравнения?

8. Что понимается под уточнением корней уравнения?

9. Отделите корни уравнения $2x^3 + 5x^2 - 3=0$.

10. Отделите корни уравнения $3x^3 - 2x^2 + 5=0$.

11. Назовите методы уточнения корней уравнения на отрезке отделения.

12. В чем сущность метода дихотомии?

13. В чем отличие метода "золотого сечения" от метода дихотомии?

14. В чем сущность метода простых итераций?

15. Назовите условия наличия экстремума внутри отрезка.

16. Сформулируйте условия выпуклости и вогнутости функции.

17. Опишите порядок решения системы линейных алгебраических уравнений методом Гаусса.

18. Что такое интеграл? Дайте геометрическую интерпретацию определенного интеграла.

19. Поясните способы вычисления определенного интеграла.

20. Что такое интерполирование функций?

21. Вычислить площадь земельного участка, который в прямоугольной системе координат описывается уравнениями $y = \varnothing$; $x_1 = 2$; $x_2 = 4$; $y = 1 / x$.

22. Вычислить работу, необходимую для откачки воды из цистерны, радиус основания которой R равен 0,5 м, высота H равна 2 м; $f(h) = 9807 \cdot S_{осн} \cdot h$.

23. Вычислить силу давления воды на вертикальную плотину, имеющую форму равнобедренной трапеции с основаниями c, d и высотой h; $f(x) = d \cdot x + ((c - d) / h) x^2$.

24. Найти путь, пройденный автомобилем за 10 с, если скорость его описывается уравнением: $v = 3t^2 + 2t + 1$.

25. Вычислить работу A силы F при сжатии пружины на 0,04 м, если для сжатия ее на 0,01 м (x_0) требуется сила 10Н (F_0). Сжатие винтовой пружины пропорционально приложенной силе; $f(x) = k_0 x_2$; $k_0 = F_0 / x_0$.

26. Найти площадь поперечного сечения канала для орошения, имеющего форму параболического сегмента, если ширина канала B равна 1 м, глубина H - 0,6 м; $f(x) = a x^2$; $a = H/(B/2)^2$.

27. Рассчитать величину управляющего сигнала разгона двигателя при $t = 3$ с, если закон управления описывается выражением $y = 3x^3 + x^2 + 5$. Решить методом Горнера.

28. Решить систему линейных уравнений:

$$\begin{cases} 7,5\,x_1 + 3,87x_2 - 1,74x_3 = 5,68; \\ 1,85x_1 - 2,34x_2 + 6,48x_3 = 8,56; \\ 3,15x_1 + 0,85x_2 + 3,79x_3 = 4,59. \end{cases}$$

29. Что такое символьная переменная?

30. Приведите классификацию функций для обработки символьных переменных.

31. Что лежит в основе сравнения символьных переменных?

32. На чем основан принцип обработки символьных переменных?

Литература

1. Алексеев А. П., Информатика, - М.: "Солон-Р", 2002. – 400 с., ил.

2. Гурин Н. И. Работа на персональном компьютере. - Мн.: Выш. школа, 1994.

3. Дьяконов В. П. Справочник по алгоритмам и программам на языке БЕЙСИК для персональных ЭВМ: Справочник. - М.: Наука. Гл. ред. физ.-мат. лит., 1987. - 240 с.

4. Использование Microsoft Windows XP Home Edition. Специальное издание.: - М.: Издательский дом "Вильямс", 2002. – 896 с.: ил.

5. Кетков GW-, Turbo-, QBasic для IBM PC

6. Коуров Л. В., Словарь-Справочник по информатике, - Мн.: Амалфея, 2000. – 176 с.

7. Кучура Н. А., Ходош М. В., Цагельский В. И. Персональные ЭВМ единой системы. - М.: Финансы и статистика, 1988.

8. Кушниренко А. Г., Лебедев Г. В., Сворень Р. А. Основы информатики и вычислительной техники. – М.: Просвещение, 1993. – 224 с.: ил.

9. Леонтьев В. П., Новейшая энциклопедия персонального компьютера 2000. – 2-ое евроиздание, перераб. и доп. – М.: ОЛМА-ПРЕСС, 2000. – 847 с.:ил.

10. Ляхович В. Ф. Основы информатики. - Ростов н/Д.: Изд-во "Феникс", 1996. –640 с.

11.Михайлов В. Ю., Степанников В. М. Современный бейсик для IBM PC. – М.: Изд-во МАИ, 1993. – 288 с.

12.Новейший самоучитель работы на компьютере. - Москва: издательство "ДЕСС КОМ", 2000. - 654 с.

13.Основы информатики: Учебное пособие; под ред. А. Н. Морозевича. – Мн.: Новое знание, 2001. – 544 с.: ил.

14.Острейковский В. А. Информатика. – М.: "Высшая школа", 2000.-511 с.: ил.

15.Поснова М. Ф., Стрикелева Л. В., Поснов Н. Н. ЭВМ для всех. – Мн.: Университетское, 1990. – 208 с.: ил.

16.Рыжиков Ю. И. Решение научно-технических задач на персональном компьютере. – СПб.: КОРОНА принт, 2000. – 272 с.: ил.

17.Светозарова Г. И. и др. Практикум по программированию на языке БЕЙСИК. - М.: Наука, 1988.

18. Турчак Л. И. Основы численных методов. -- М.: Наука. Гл. ред. физ.-мат. лит., 1987. – 320 с.

19.Фигурнов В. Ф. IBM PC для пользователя. - М.: Финансы и статистика, 2003.

20.Экономическая информатика / под ред. П. В. Конюховского и Д. Н. Колесова. – СПб.: Питер, 2001. – 560 с.: ил.

Учебное издание

Составитель: Вячеслав Леонидович Быков

Основы информатики

(конспект лекций)

ISBN 985-6584-61-2

9 789856 584612

Ответственный за выпуск: В.Л. Быков
Редактор: Т.В. Строкач
Корректор: Е.В. Никитчик
Компьютерный набор: В.Л. Быков, Е.К. Августинович
Компьютерная вёрстка: Е.А. Боровикова

ЛВ № 382 от 1.09 2002 г. ЛП № 178 от 14.01. 2003 г.
Подписано в печать 22.09.2003 г. Формат 60x84/16. Бумага «Писчая». Усл. п. л. 14,8. Уч. изд. л. 15,9. Тираж 200 экз. Заказ № 671. Отпечатано на ризографе Учреждения образования «Брестский государственный технический университет». 224017, г. Брест, ул. Московская, 267.